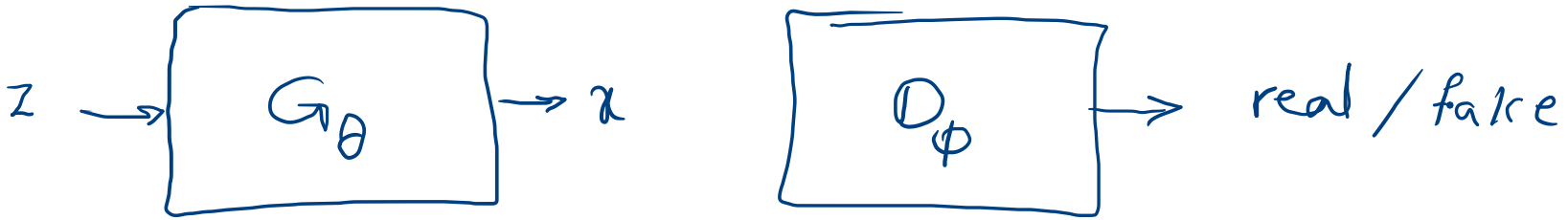
GAN

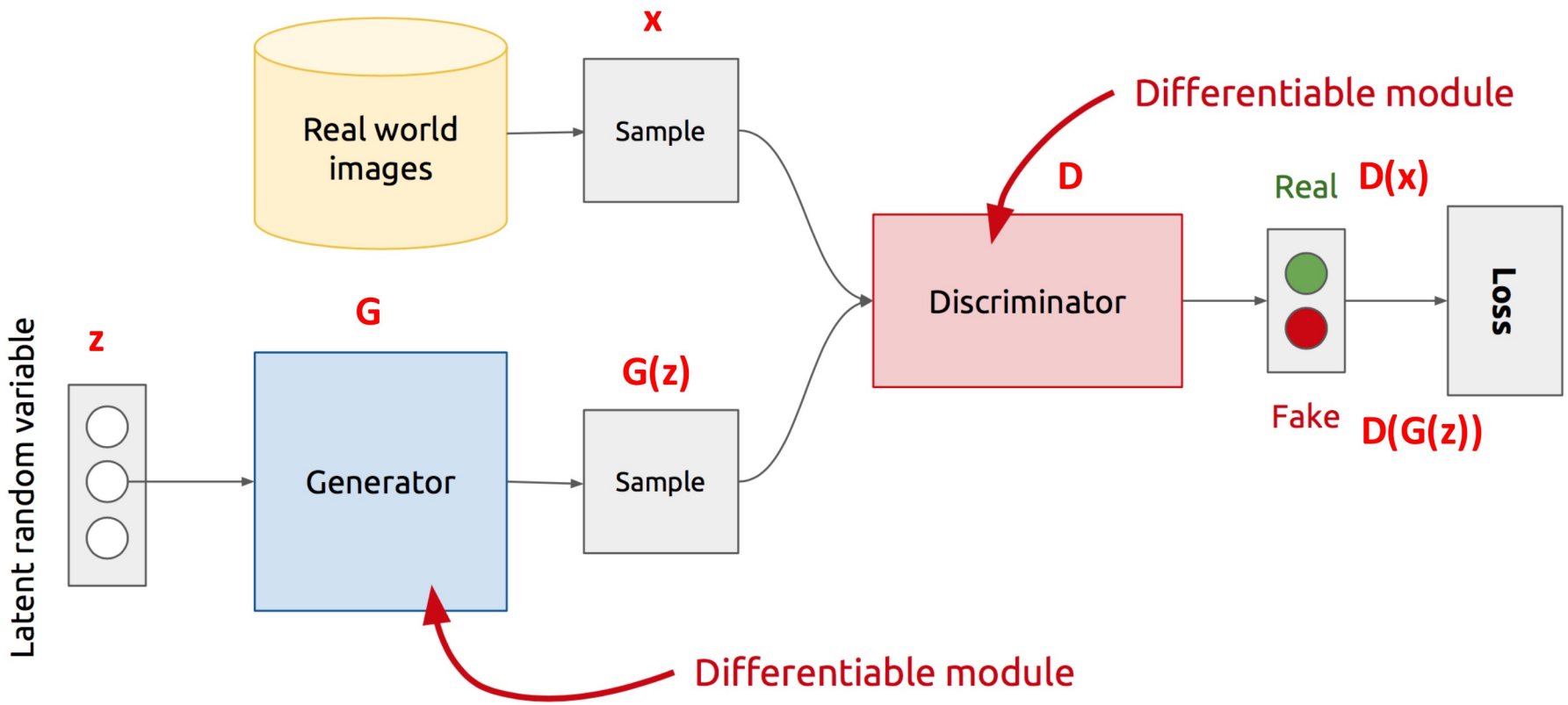
$D = \{x_1, \ldots, x_n\}$



$z \sim N(0, I)$

$$\max_{G_\theta} \min_{D_\phi} V(\phi, \theta) = \mathbb{E}_{x \sim p_{data}}\left[\log D(x)\right] + \mathbb{E}_{x \sim p_\theta}\left[\log\left(1 - D(x)\right)\right]$$

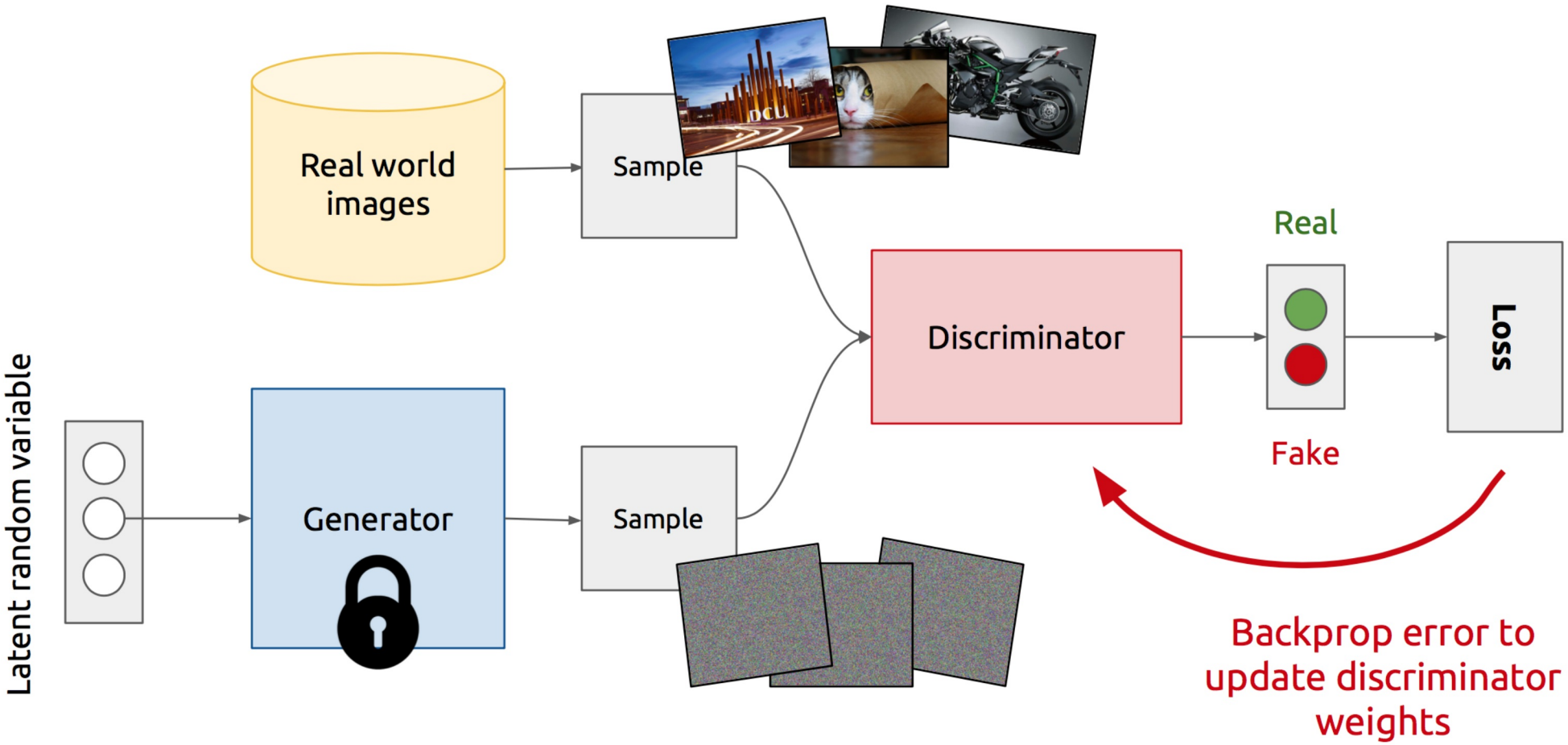
GAN's Architecture



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

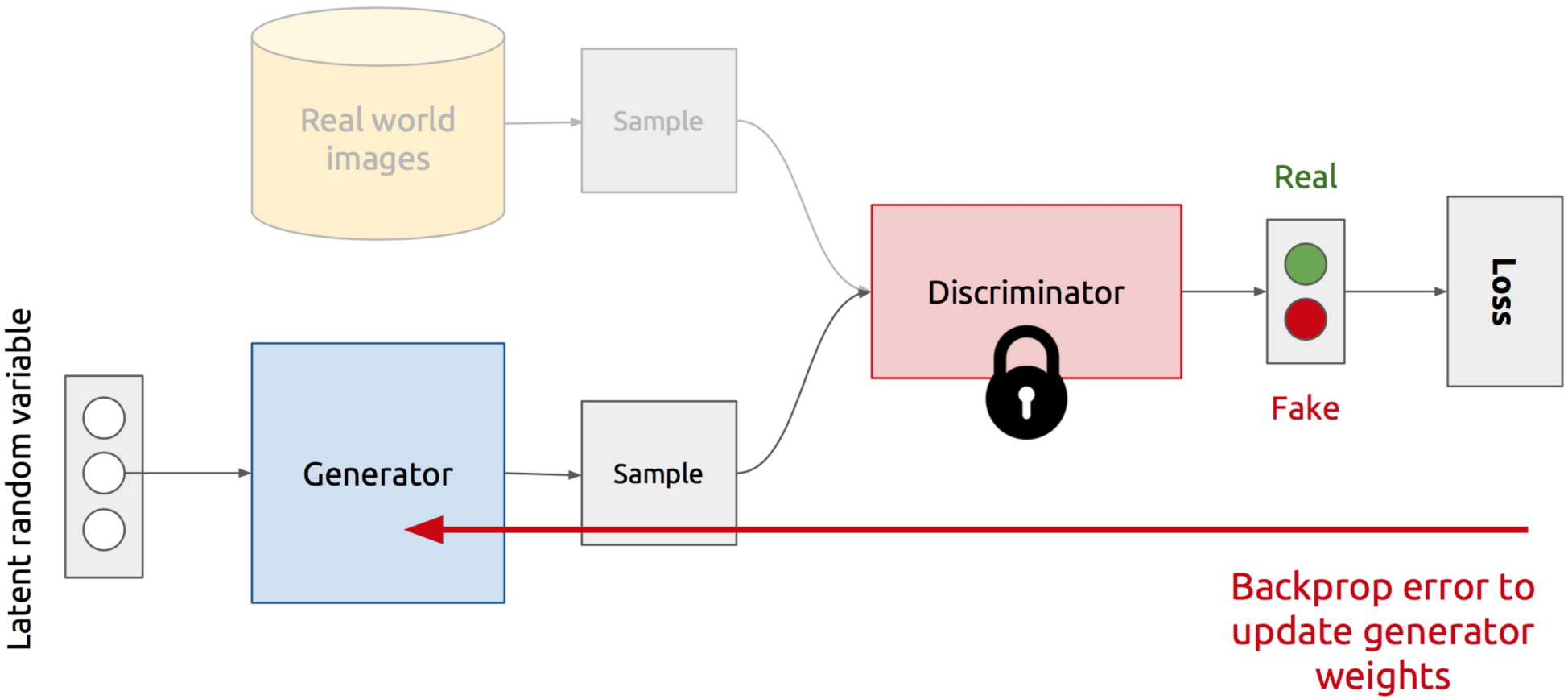
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Discriminator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Generator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Generative Adversarial Networks

Stefano Ermon

Stanford University

Lecture 10

Recap.

- Likelihood-free training
- Training objective for GANs

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

- With the optimal discriminator D^*_G, we see GAN minimizes a scaled and shifted Jensen-Shannon divergence

$$\min_G 2D_{JSD}[p_{\text{data}}, p_G] - \log 4$$

- Parameterize D by ϕ and G by θ. Prior distribution $p(\mathbf{z})$.

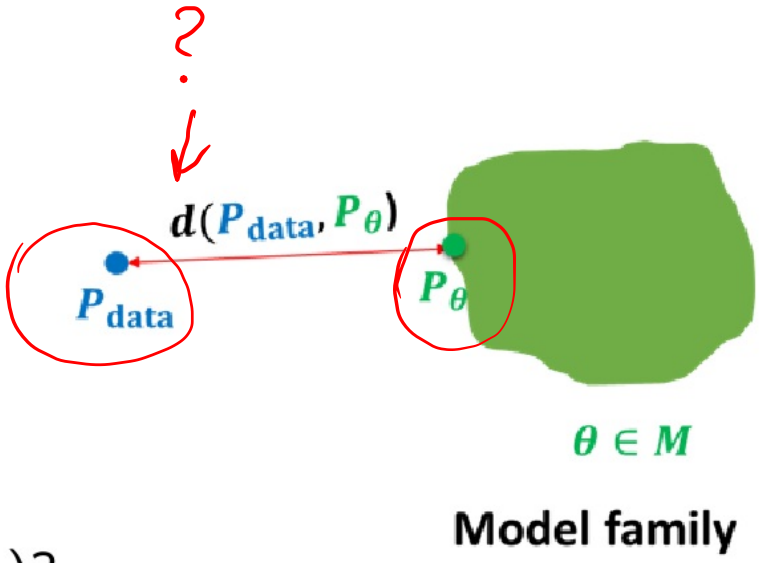
$$\min_\theta \max_\phi E_{\mathbf{x} \sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

Selected GANs

- `https://github.com/hindupuravinash/the-gan-zoo`
  The GAN Zoo: List of all named GANs
- Today
  - Rich class of likelihood-free objectives via f-GANs
  - Wasserstein GAN
  - Inferring latent representations via BiGAN
  - Application: Image-to-image translation via CycleGANs

Beyond KL and Jenson-Shannon Divergence

$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



What choices do we have for $d(\cdot)$?

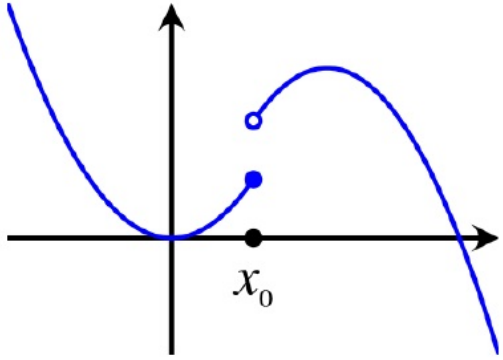
- KL divergence: Autoregressive Models, Flow models
- (scaled and shifted) Jensen-Shannon divergence (approximately): original GAN objective

f divergences

- Given two densities p and q, the f-divergence is given by

$$D_f(p, q) = E_{\mathbf{x} \sim q}\left[f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right]$$

  where f is any convex, lower-semicontinuous function with $f(1) = 0$.
- Convex: Line joining any two points lies above the function
- Lower-semicontinuous: function value at any point $\mathbf{x}_0$ is close to $f(\mathbf{x}_0)$ or greater than $f(\mathbf{x}_0)$



- Jensen's inequality: $E_{\mathbf{x} \sim q}[f(p(\mathbf{x})/q(\mathbf{x}))] \geq f(E_{\mathbf{x} \sim q}[p(\mathbf{x})/q(\mathbf{x})]) = f(\int q(x)p(\mathbf{x})/q(\mathbf{x})) = f(\int p(x)) = f(1) = 0$
- Example: KL divergence with $f(u) = u \log u$

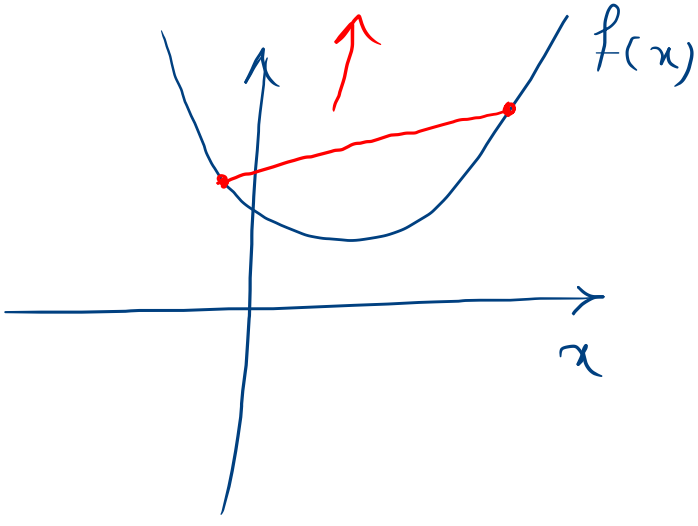
$$D(P, q) = E_q\left[f\left(\frac{P(x)}{q(x)}\right)\right] \geq 0$$

① $D(P, q) \geq 0$

② $D(P, q) = 0 \iff P = q$

Jensen inequality,

$$E[f(x)] \geq f(E[x])$$



$$D(P, q) = E_q\left[f\left(\frac{P(x)}{q(x)}\right)\right] \geq f\left(E_q\left[\frac{P(x)}{q(x)}\right]\right)$$

$$= f\left(\int \cancel{q(x)} \frac{P(x)}{\cancel{q(x)}} dx\right) = 0$$

$$D(p,q) = E_q\left[f\left(\frac{p(x)}{q(x)}\right)\right]$$

$$f(u) = u \log u$$

$$D(p,q) = E_q\left[\frac{p(x)}{q(x)} \log \frac{p(x)}{q(x)}\right] = \int \cancel{q(x)} \frac{p(x)}{\cancel{q(x)}} \log \frac{p(x)}{q(x)} dx$$

$$= \int p(x) \log \frac{p(x)}{q(x)} dx = KL(p \| q)$$

$$f(u) = -\log u \longrightarrow \frac{-1}{u} \longrightarrow \frac{1}{u^2}$$

$$D(p,q) = KL(q \| p)$$

f divergences

Many more f-divergences!

Name	$D_f(P\|Q)$	Generator $f(u)$
Total variation	$\frac{1}{2}\int \lvert p(x)-q(x)\rvert \,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$
Reverse Kullback-Leibler	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2 \mathrm{d}x$	$(\sqrt{u}-1)^2$
Jeffrey	$\int (p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x$	$(u-1)\log u$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)} + q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2} + u\log u$
Jensen-Shannon-weighted	$\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)} + (1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$	$\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)} + q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x - \log(4)$	$u\log u-(u+1)\log(u+1)$
α-divergence ($\alpha\notin\{0,1\}$)	$\frac{1}{\alpha(\alpha-1)}\int\left(p(x)\left[\left(\frac{q(x)}{p(x)}\right)^{\alpha}-1\right]-\alpha(q(x)-p(x))\right)\mathrm{d}x$	$\frac{1}{\alpha(\alpha-1)}\left(u^{\alpha}-1-\alpha(u-1)\right)$

Source: Nowozin et al., 2016

Training with f-divergences

- Given p_{data} and p_θ, we could minimize $D_f(p_\theta, p_{data})$ or $D_f(p_{data}, p_\theta)$ as learning objectives. Non-negative, and zero if $p_\theta = p_{data}$
- However, it depends on the density ratio which is unknown

$$D_f(p_\theta, p_{data}) = \underbrace{E_{\mathbf{x}\sim p_{data}}}_{\text{approx w. samples}} \left[\underbrace{f\left(\frac{p_\theta(\mathbf{x})}{p_{data}(\mathbf{x})}\right)}_{\text{uknown ratio}} \right]$$

$$D_f(p_{data}, p_\theta) = \underbrace{E_{\mathbf{x}\sim p_\theta}}_{\text{approx w. samples}} \left[\underbrace{f\left(\frac{p_{data}(\mathbf{x})}{p_\theta(\mathbf{x})}\right)}_{\text{uknown ratio}} \right]$$

- To use f-divergences as a two-sample test objective for likelihood-free learning, we need to be able to estimate the objective using only samples (e.g., training data and samples from the model)

f-GAN

Towards Variational Divergence Minimization

- Fenchel conjugate: For any function $f(\cdot)$, its convex conjugate is

$$f^*(t) = \sup_{u \in \mathrm{dom}_f} (ut - f(u))$$

  where dom_f is the domain of the function f
- f^* is convex (pointwise supremum of convex functions is convex) and lower semi-continuous.
- Let f^{**} be the Fenchel conjugate of f^*

$$f^{**}(u) = \sup_{t \in \mathrm{dom}_{f^*}} (tu - f^*(t))$$

- $f^{**} \leq f$. Proof: By definition, for all t, u

$$f^*(t) \geq ut - f(u) \quad \text{or equivalently} \quad f(u) \geq ut - f^*(t)$$

$$f(u) \geq \sup_t (ut - f^*(t)) = f^{**}(u)$$

- Strong Duality: $f^{**}{=}f$ when $f(\cdot)$ is convex, lower semicontinous.

f-GAN: Variational Divergence Minimization

- We obtain a lower bound to an f-divergence via Fenchel conjugate

$$
\begin{aligned}
D_f(p,q) &= E_{\mathbf{x}\sim q}\left[f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right] = E_{\mathbf{x}\sim q}\left[f^{**}\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right] \\
&\overset{f^{**}\ \text{def}}{=} E_{\mathbf{x}\sim q}\left[\sup_{t\in \mathrm{dom}_{f^*}}\left(t\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(t)\right)\right] \\
&= E_{\mathbf{x}\sim q}\left[T^*(x)\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(T^*(x))\right] \\
&= \int_{\mathcal{X}} q(x)\left[T^*(x)\frac{p(\mathbf{x})}{q(\mathbf{x})} - f^*(T^*(x))\right]\mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{X}} \left[T^*(x)p(\mathbf{x}) - f^*(T^*(x))q(\mathbf{x})\right]\mathrm{d}\mathbf{x} \\
&= \sup_T \int_{\mathcal{X}} \left[T(x)p(\mathbf{x}) - f^*(T(x))q(\mathbf{x})\right]\mathrm{d}\mathbf{x} \\
&\geq \sup_{T\in\mathcal{T}} \int_{\mathcal{X}} (T(\mathbf{x})p(\mathbf{x}) - f^*(T(\mathbf{x}))q(\mathbf{x}))\mathrm{d}\mathbf{x} \\
&= \sup_{T\in\mathcal{T}} \left(E_{\mathbf{x}\sim p}\left[T(\mathbf{x})\right] - E_{\mathbf{x}\sim q}\left[f^*(T(\mathbf{x}))\right]\right)
\end{aligned}
$$

  where $\mathcal{T} : \mathcal{X} \mapsto \mathbb{R}$ is an arbitrary class of functions

- **Note:** Lower bound is likelihood-free w.r.t. p and q

f-GAN: Variational Divergence Minimization

- Variational lower bound

$$D_f(p, q) \geq \sup_{T \in \mathcal{T}} \left(E_{\mathbf{x} \sim p}[T(\mathbf{x})] - E_{\mathbf{x} \sim q}[f^*(T(\mathbf{x}))] \right)$$

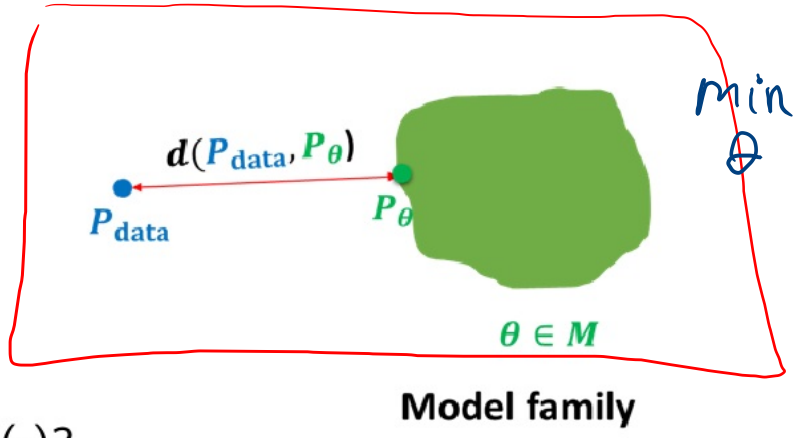
- Choose any f-divergence
- Let $p = p_{\text{data}}$ and $q = p_G$
- Parameterize T by ϕ and G by θ
- Consider the following f-GAN objective

$$\min_{\theta} \max_{\phi} F(\theta, \phi) = E_{\mathbf{x} \sim p_{\text{data}}}[T_\phi(\mathbf{x})] - E_{\mathbf{x} \sim p_{G_\theta}}[f^*(T_\phi(\mathbf{x}))]$$

- Generator G_θ tries to minimize the divergence estimate and discriminator T_ϕ tries to tighten the lower bound
- Substitute any f-divergence and optimize the f-GAN objective

Beyond KL and Jenson-Shannon Divergence

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



Model family

$\min_\theta KL(P_{data} \| P_\theta)$

$= \max_\theta$ likelihood

What choices do we have for $d(\cdot)$?

- KL divergence: Autoregressive Models, Flow models
- (scaled and shifted) Jensen-Shannon divergence (approximately): via the original GAN objective
- Any other f-divergence (approximately): via the f-GAN objective

Problems with GANs

- **Probability Distribution is Implicit**
  - Not straightforward to compute P(X).
  - Thus **Vanilla GANs** are only good for Sampling/Generation.

- **Training is Hard**
  - Non-Convergence
  - Mode-Collapse

Goodfellow, Ian. **"NIPS 2016 Tutorial: Generative Adversarial Networks."** arXiv preprint arXiv:1701.00160 (2016).

Training Problems

- **Non-Convergence**
- Mode-Collapse

GAN's formulation

$$\min_G \max_D V(D,G)$$

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D,G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

- The Nash equilibrium of this particular game is achieved at:
  - $P_{data}(x) = P_{gen}(x) \;\; \forall x$
  - $\mathrm{D}(x) = \frac{1}{2} \;\; \forall x$

- **Deep Learning models (in general) involve a single player**
  - The player tries to maximize its reward (minimize its loss).
  - Use SGD (with Backpropagation) to find the optimal parameters.
  - SGD has convergence guarantees (under certain conditions).
  - **Problem:** With non-convexity, we might converge to local optima.

$$\min_G L(G)$$

$$\min_\theta \max_\phi V(\theta, \phi)$$

- **GANs instead involve two (or more) players**
  - Discriminator is trying to maximize its reward.
  - Generator is trying to minimize Discriminator's reward.

$$\min_G \max_D V(D, G)$$

  - SGD was not designed to find the Nash equilibrium of a game.
  - **Problem:** We might not converge to the Nash equilibrium at all.

Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

Non-Convergence

$$\min_{x} \max_{y} V(x, y)$$

Let $V(x, y) = xy$

• State 1:	x > 0	y > 0	V > 0	Increase y / Decrease x
• State 2:	x < 0	y > 0	V < 0	Decrease y / Decrease x
• State 3:	x < 0	y < 0	V > 0	Decrease y / Increase x
• State 4 :	x > 0	y < 0	V < 0	Increase y / Increase x
• State 5:	x > 0	y > 0	V > 0	== State 1 / Increase y / Decrease x

Vanishing gradient strikes back again…

$\sigma(x) \rightarrow \sigma'(x) = \sigma(x)(1-\sigma(x))$

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1-D(G(z)))]$$

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1-D\left(G(z)\right)\right)\right]$$

- $\nabla_a \log(1-\sigma(a)) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)(1-\sigma(a))}{1-\sigma(a)} = -\sigma(a) = -D(G(z))$
- Gradient goes to 0 if D is confident, i.e. $D(G(z)) \rightarrow 0$

- Minimize $-\mathbb{E}_{z\sim q(z)}[\log D(G(z))]$ for **Generator** instead (keep Discriminator as it is)

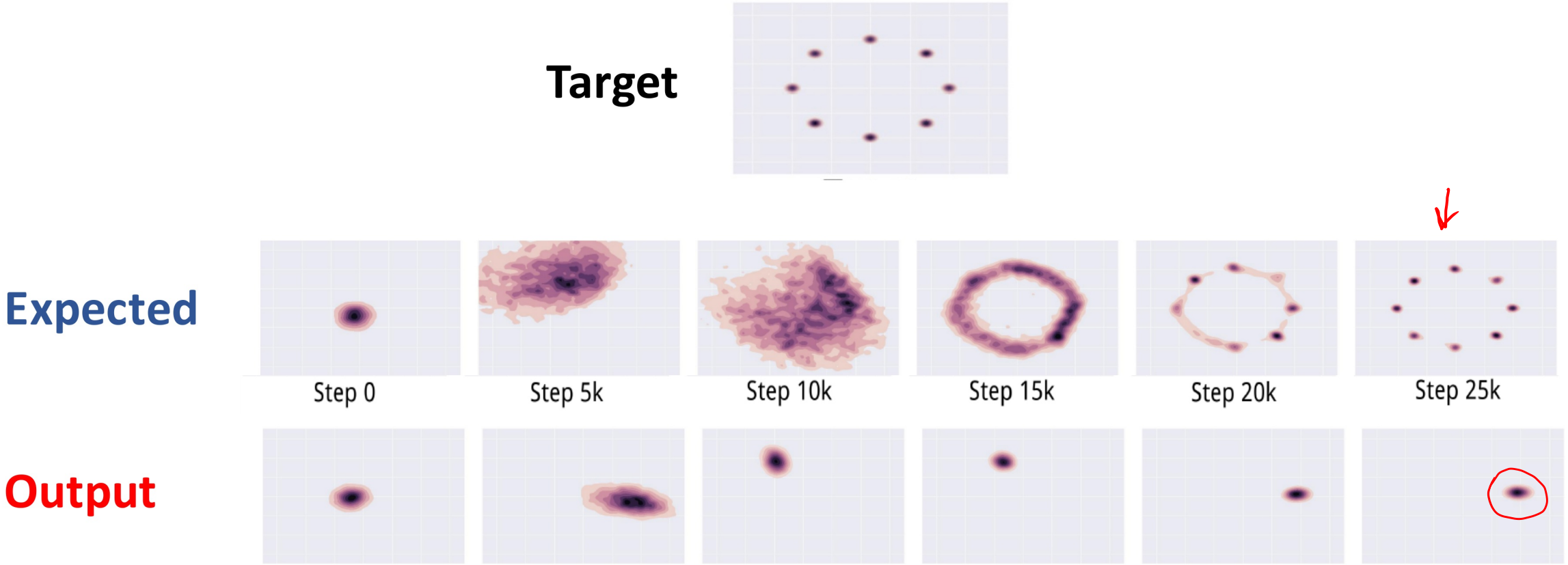
min

Problems with GANs

- Non-Convergence
- **Mode-Collapse**

Mode-Collapse

- **Generator fails to output diverse samples**



Metz, Luke, et al. **"Unrolled Generative Adversarial Networks."** arXiv preprint arXiv:1611.02163 (2016).

Some real examples

Reed, S., et al. ***Generating interpretable images with controllable structure.*** Technical report, 2016. 2, 2016.

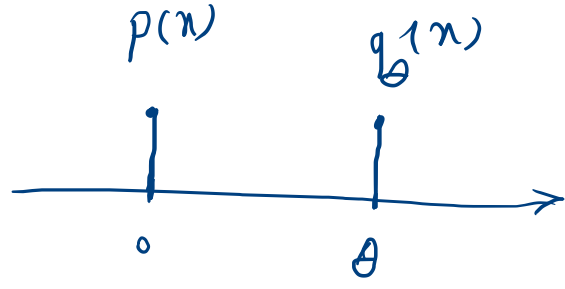
Wasserstein GAN: beyond f-divergences

The f-divergence is defined as

$$D_f(p, q) = E_{\mathbf{x} \sim q}\left[f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right]$$

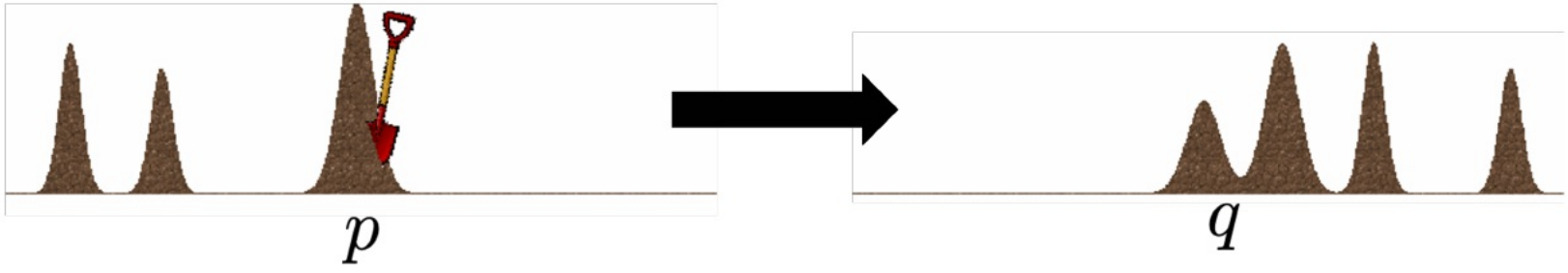
$\min_\theta KL(P \| q_\theta)$

- The support of q has to cover the support of p. Otherwise discontinuity arises in f-divergences.
  - Let $p(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} = 0 \\ 0, & \mathbf{x} \neq 0 \end{cases}$, and $q_\theta(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} = \theta \\ 0, & \mathbf{x} \neq \theta \end{cases}$.
  - $D_{KL}(p, q_\theta) = \begin{cases} 0, & \theta = 0 \\ \infty, & \theta \neq 0 \end{cases}$.
  - $D_{JS}(p, q_\theta) = \begin{cases} 0, & \theta = 0 \\ \log 2, & \theta \neq 0 \end{cases}$.



- We need a "smoother" distance $D(p, q)$ that is defined when p and q have disjoint supports.

Wasserstein (Earth-Mover) distance

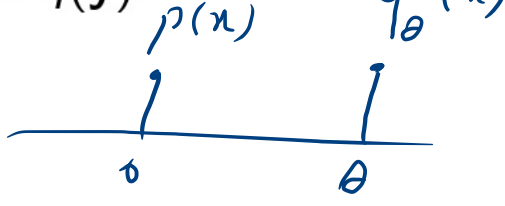


- Wasserstein distance

$$D_w(p, q) = \inf_{\gamma \in \Pi(p,q)} E_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x} - \mathbf{y}\|_1],$$

where $\Pi(p, q)$ contains all joint distributions of $(\mathbf{x}, \mathbf{y})$ where the marginal of $\mathbf{x}$ is $p(\mathbf{x}) = \int \gamma(\mathbf{x}, \mathbf{y})d\mathbf{y}$, and the marginal of $\mathbf{y}$ is $q(\mathbf{y})$.

- $\gamma(\mathbf{y} \mid \mathbf{x})$: a probabilistic earth moving plan that warps $p(\mathbf{x})$ to $q(\mathbf{y})$.
- Let $p(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} = 0 \\ 0, & \mathbf{x} \neq 0 \end{cases}$, and $q_\theta(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} = \theta \\ 0, & \mathbf{x} \neq \theta \end{cases}$.
- $D_w(p, q_\theta) = |\theta|$.



Wasserstein GAN (WGAN)

- Kantorovich-Rubinstein duality

$$D_w(p, q) = \sup_{\|f\|_L \leq 1} E_{\mathbf{x}\sim p}[f(\mathbf{x})] - E_{\mathbf{x}\sim q}[f(\mathbf{x})]$$

$\|f\|_L \leq 1$ means the Lipschitz constant of $f(\mathbf{x})$ is 1. Technically,

$$\forall \mathbf{x}, \mathbf{y}: \quad |f(\mathbf{x}) - f(\mathbf{y})| \leq \|\mathbf{x} - \mathbf{y}\|_1$$

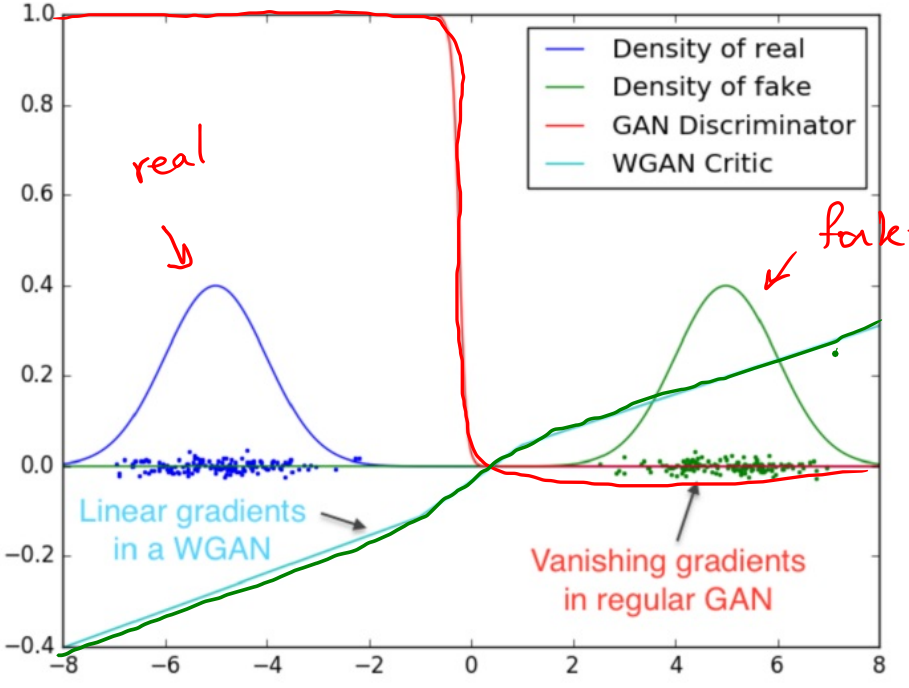
Intuitively, f cannot change too rapidly.

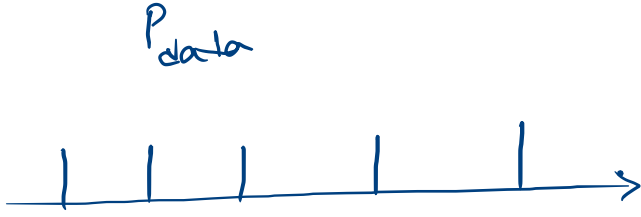
- Wasserstein GAN with discriminator $D_\phi(\mathbf{x})$ and generator $G_\theta(\mathbf{z})$:

$$\min_\theta \max_\phi E_{\mathbf{x}\sim p_{\text{data}}}[D_\phi(\mathbf{x})] - E_{\mathbf{z}\sim p(\mathbf{z})}[D_\phi(G_\theta(\mathbf{z}))]$$

Lipschitzness of $D_\phi(\mathbf{x})$ is enforced through weight clipping or gradient penalty on $\nabla_{\mathbf{x}} D_\phi(\mathbf{x})$.

Wasserstein GAN (WGAN)





- More stable training, and less mode collapse.

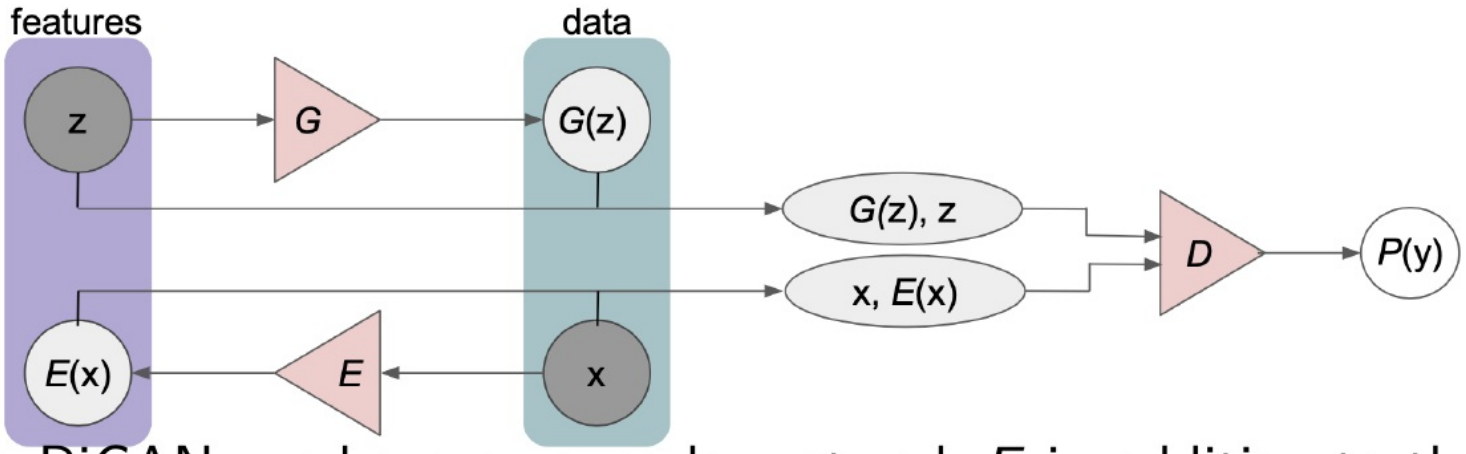
Inferring latent representations in GANs

- The generator of a GAN is typically a directed, latent variable model with latent variables $\mathbf{z}$ and observed variables $\mathbf{x}$. How can we infer the latent feature representations in a GAN?
- Unlike a normalizing flow model, the mapping $G : \mathbf{z} \mapsto \mathbf{x}$ need not be invertible
- Unlike a variational autoencoder, there is no inference network $q(\cdot)$ which can learn a variational posterior over latent variables
- **Solution 1**: For any point $\mathbf{x}$, use the activations of the prefinal layer of a discriminator as a feature representation
- Intuition: Similar to supervised deep neural networks, the discriminator would have learned useful representations for $\mathbf{x}$ while distinguishing real and fake $\mathbf{x}$

Inferring latent representations in GANs

- If we want to directly infer the latent variables $\mathbf{z}$ of the generator, we need a different learning algorithm
- A regular GAN optimizes a two-sample test objective that compares samples of $\mathbf{x}$ from the generator and the data distribution
- **Solution 2**: To infer latent representations, we will compare samples of $\mathbf{x}, \mathbf{z}$ from the joint distributions of observed and latent variables as per the model and the data distribution
- For any $\mathbf{x}$ generated via the model, we have access to $\mathbf{z}$ (sampled from a simple prior $p(\mathbf{z})$)
- For any $\mathbf{x}$ from the data distribution, the $\mathbf{z}$ is however unobserved (latent). Need an encoder!

Bidirectional Generative Adversarial Networks (BiGAN)

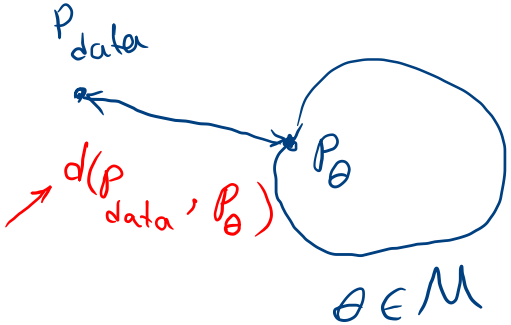


- In a BiGAN, we have an encoder network E in addition to the generator network G
- The encoder network only observes $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ during training to learn a mapping $E : \mathbf{x} \mapsto \mathbf{z}$
- As before, the generator network only observes the samples from the prior $\mathbf{z} \sim p(\mathbf{z})$ during training to learn a mapping $G : \mathbf{z} \mapsto \mathbf{x}$

f-GAN

f-Divergence $\quad D(p,q) = E_q\left[f\left(\frac{p(x)}{q(x)}\right)\right]$

1. convex
2. lower-semicontinuous
3. $f(1) = 0$



$$\hat{\theta} = \arg\max_\theta KL(p_{data} \,\|\, p_\theta)$$

$$= \arg\max_\theta E_{x\sim p_{data}}\left[\log p_\theta(x)\right]$$

NF

$$\simeq \arg\max_\theta \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i)$$

$$\hat{\theta} = \arg\max_\theta \text{ ELBO}$$

← VAE

GAN: $d(P_{data}, P_\theta) = 2\,JSD(P_{data}, P_\theta) - \log 4$

Wasserstein GAN (WGAN)

$$W(p, q) = \inf_{\gamma \in \Pi(p,q)} E\left[\|x - y\|_1\right]$$

$p(x)$

$q(y)$

$$q(y) = \int \gamma(x, y)\, dx$$

$$p(x) = \int \gamma(x, y)\, dy$$

$$W(p,q) = \sup_{\|f\|_L \le 1} E_p[f(x)] - E_q[f(x)]$$

$$W(P_{data}, P_\theta) \simeq \sup_f \frac{1}{n}\sum_{i=1}^{n} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

$x_i \sim P_{data}$ $\qquad$ $x_i \sim P_\theta$

$$\min_\theta W(P_{data}, P_\theta)$$

$$W(P_{data}, P_\theta) = \max_\phi \frac{1}{n}\sum_{i=1}^{n} D_\phi(x_i) - \frac{1}{n}\sum_{i=1}^{n} D_\phi(G_\theta(z_i))$$

$$\min_\theta \max_\phi \frac{1}{n}\sum_{i=1}^{n} D_\phi(x_i) - \frac{1}{n}\sum_{i=1}^{n} D_\phi(G_\theta(z_i)) + \lambda \|\nabla_x D_\phi(x)\|$$

$x_i \sim P_{data}$

$x \rightarrow [D_\phi] \rightarrow$ real/fake

$$\frac{\partial D}{\partial x} = \frac{\partial D}{\partial \phi}\frac{\partial \phi}{\partial x}$$

$1 > \phi(z) > 0$

$\phi'(z) = \phi(z)(1-\phi(z))$

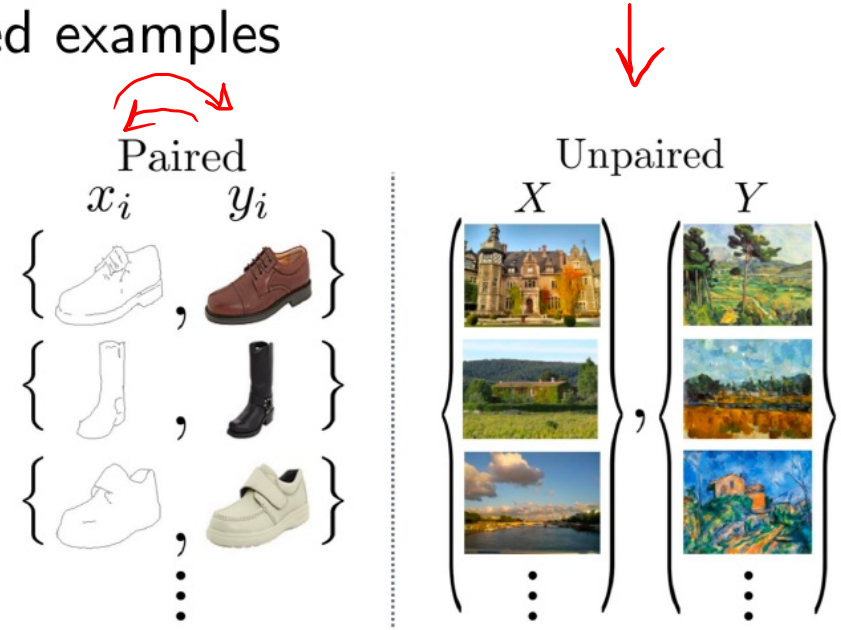
L- Lipschitz :

$$\| f(x) - f(x') \| \leq L \| x - x' \|$$

$$\frac{\| f(x) - f(x+\Delta x) \|}{\| \Delta x \|} \leq L$$

Translating across domains

- Image-to-image translation: We are given images from two domains, $\mathcal{X}$ and $\mathcal{Y}$
- Paired vs. unpaired examples

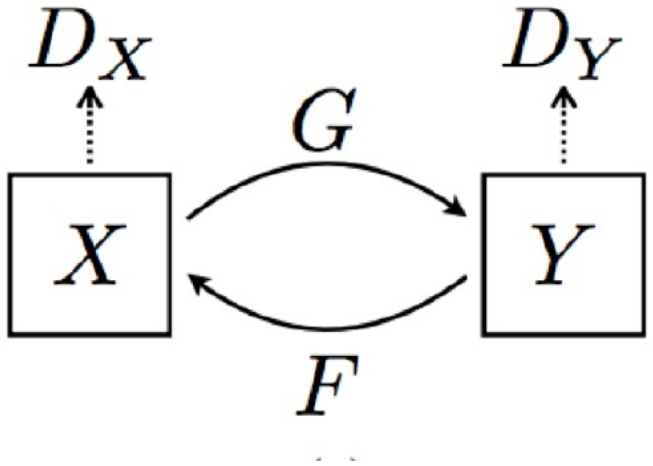


Source: Zhu et al., 2016

- Paired examples can be expensive to obtain. Can we translate from $\mathcal{X} \leftrightarrow \mathcal{Y}$ in an unsupervised manner?

CycleGAN: Adversarial training across two domains

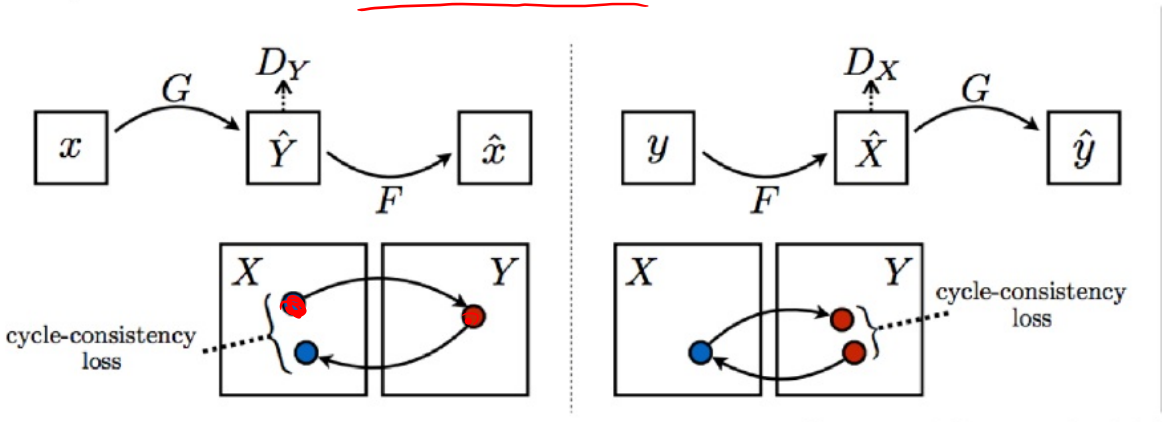
- To match the two distributions, we learn two parameterized conditional generative models $G : \mathcal{X} \mapsto \mathcal{Y}$ and $F : \mathcal{Y} \mapsto \mathcal{X}$
- G maps an element of $\mathcal{X}$ to an element of $\mathcal{Y}$. A discriminator $D_{\mathcal{Y}}$ compares the observed dataset Y and the generated samples $\hat{Y} = G(X)$
- Similarly, F maps an element of $\mathcal{Y}$ to an element of $\mathcal{X}$. A discriminator $D_{\mathcal{X}}$ compares the observed dataset X and the generated samples $\hat{X} = F(Y)$



Source: Zhu et al., 2016

CycleGAN: Cycle consistency across domains

- **Cycle consistency:** If we can go from X to $\hat{Y}$ via G, then it should also be possible to go from $\hat{Y}$ back to X via F
  - $F(G(X)) \approx X$
  - Similarly, vice versa: $G(F(Y)) \approx Y$

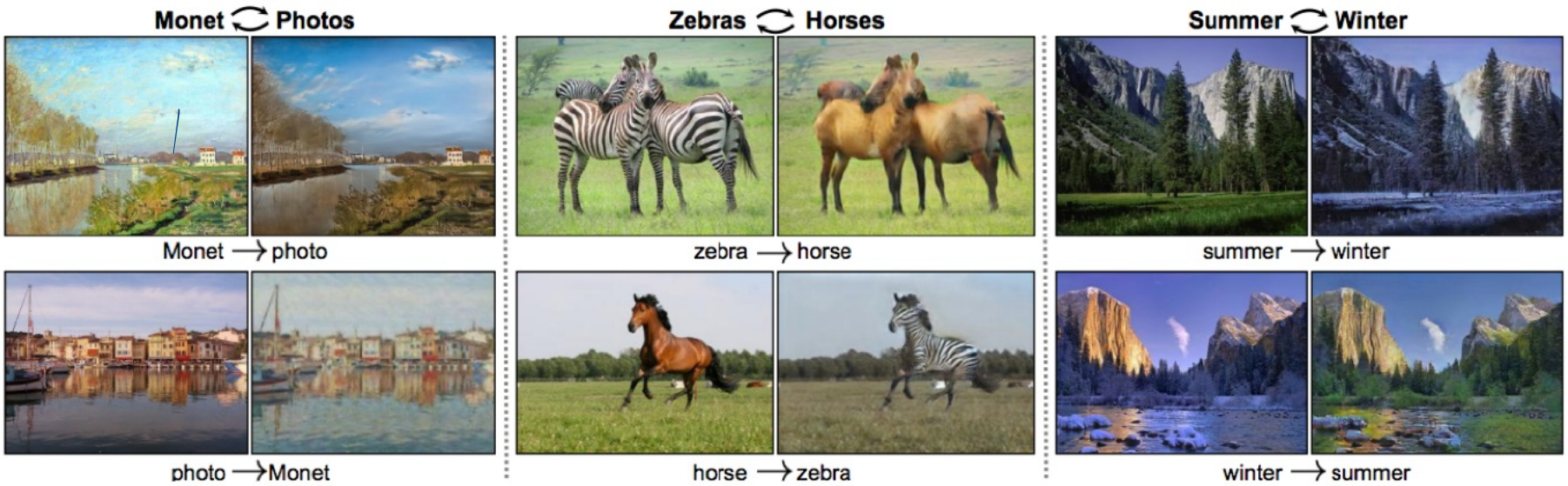


Source: Zhu et al., 2016

- Overall loss function

$$\min_{F,G,D_X,D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, X, Y)$$
$$+\lambda \underbrace{\left(E_X[\|F(G(X)) - X\|_1] + E_Y[\|G(F(Y)) - Y\|_1]\right)}_{\text{cycle consistency}}$$

CycleGAN in practice



Source: Zhu et al., 2016

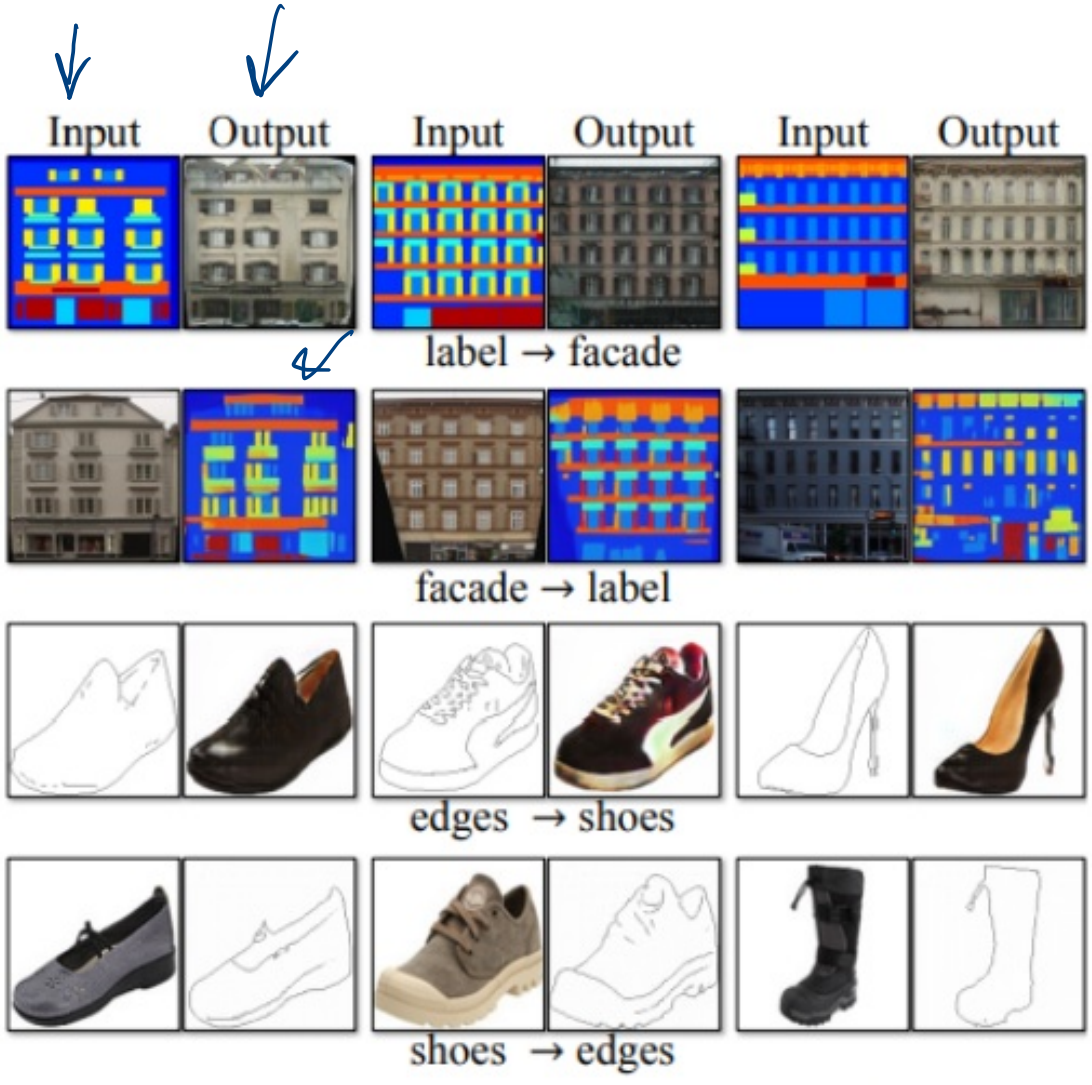


Figure 8: Example results of CycleGAN on paired datasets used in "pix2pix" [22] such as architectural labels↔photos and edges↔shoes.

AlignFlow (Grover et al.)

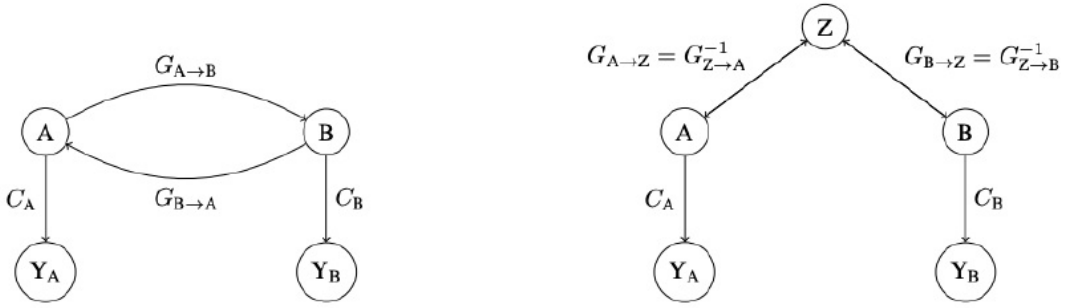


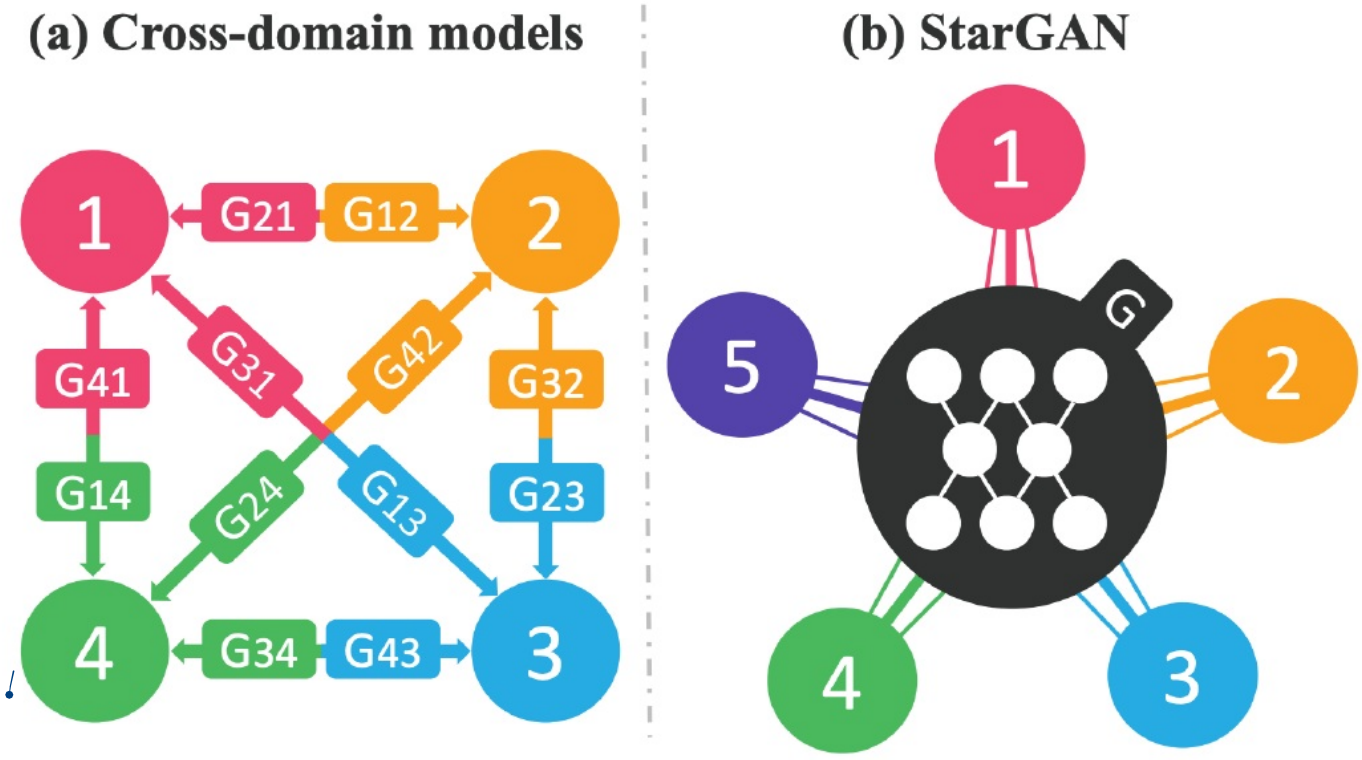
Figure 1: CycleGAN v.s. AlignFlow for unpaired cross-domain translation. Unlike CycleGAN, AlignFlow specifies a single invertible mapping $G_{A\to Z} \circ G^{-1}_{B\to Z}$ that is exactly cycle-consistent, represents a shared latent space Z between the two domains, and can be trained via both adversarial training and exact maximum likelihood estimation. Double-headed arrows denote invertible mappings. Y_A and Y_B are random variables denoting the output of the critics used for adversarial training.

- What if G is a flow model?
- No need to parameterize F separately! $F = G^{-1}$
- Can train via MLE and/or adversarial learning!
- Exactly cycle-consistent

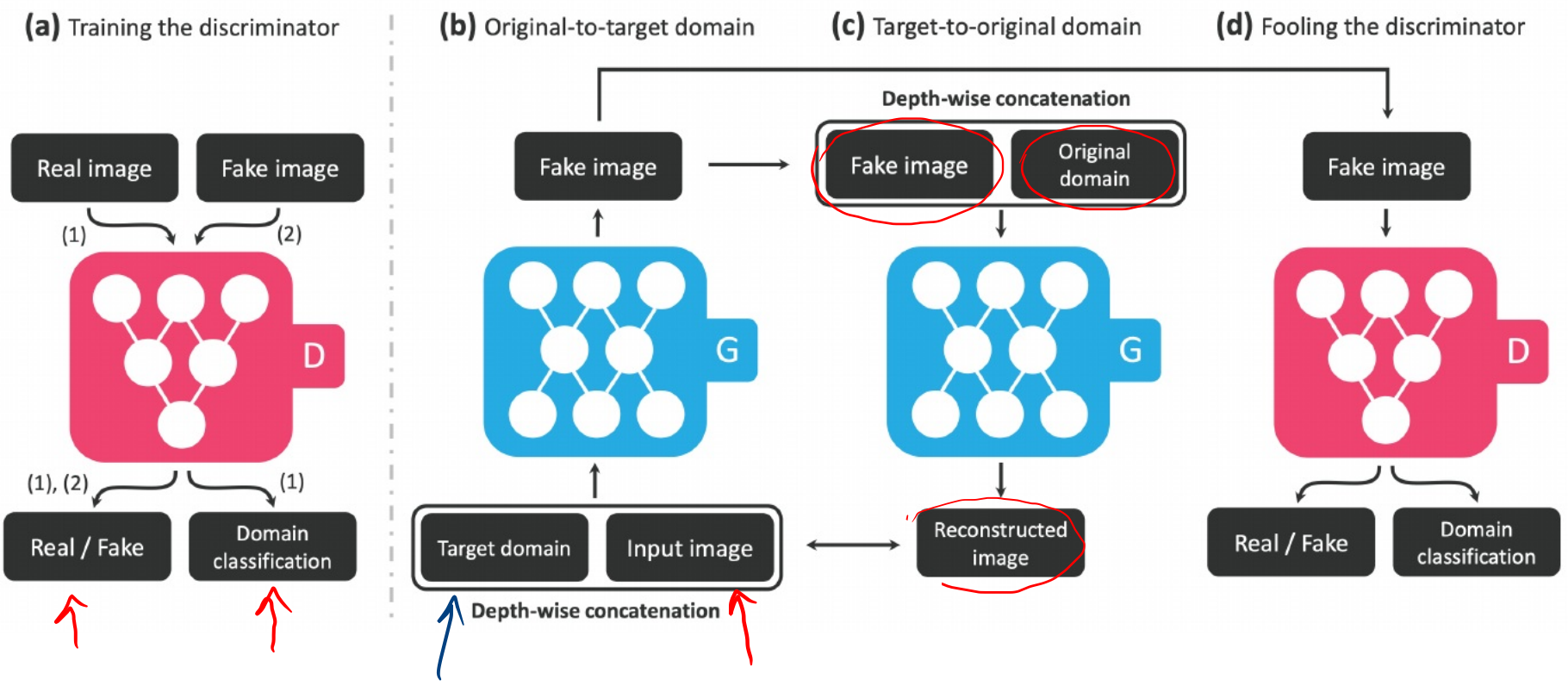
$$F(G(X)) = X$$
$$G(F(Y)) = Y$$

StarGAN (Choi et al.)

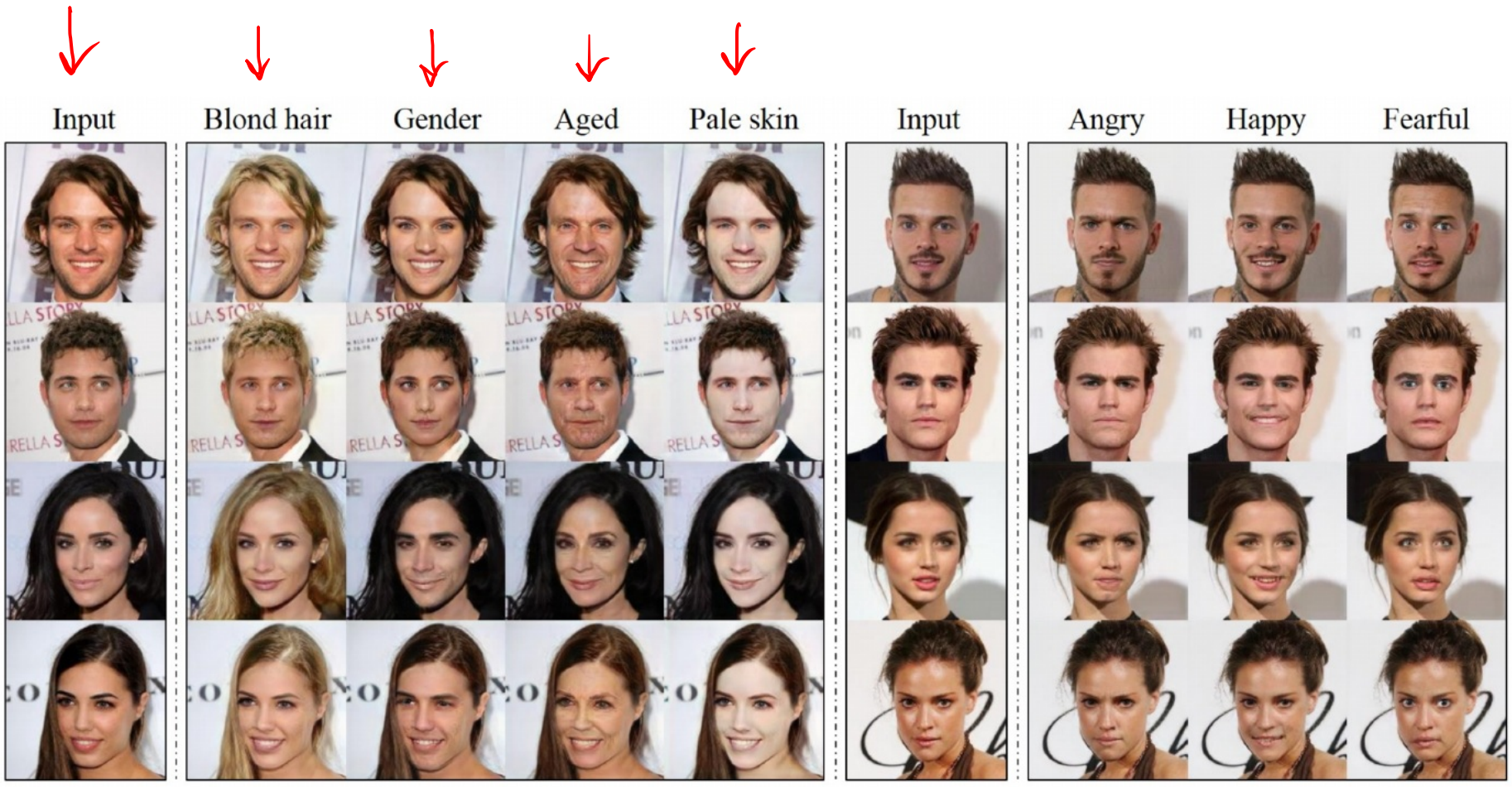
- What if there are multiple domains?



StarGAN (Choi et al.)



StarGAN (Choi et al.)



Summary of Generative Adversarial Networks

- Key observation: Samples and likelihoods are not correlated in practice
- Two-sample test objectives allow for learning generative models only via samples (likelihood-free)
- Wide range of two-sample test objectives covering f-divergences and Wasserstein distances (and more)
- Latent representations can be inferred via BiGAN
- Cycle-consistent domain translations via CycleGAN, AlignFlow and StarGAN.

Progressive GAN

style GAN 1,2,3

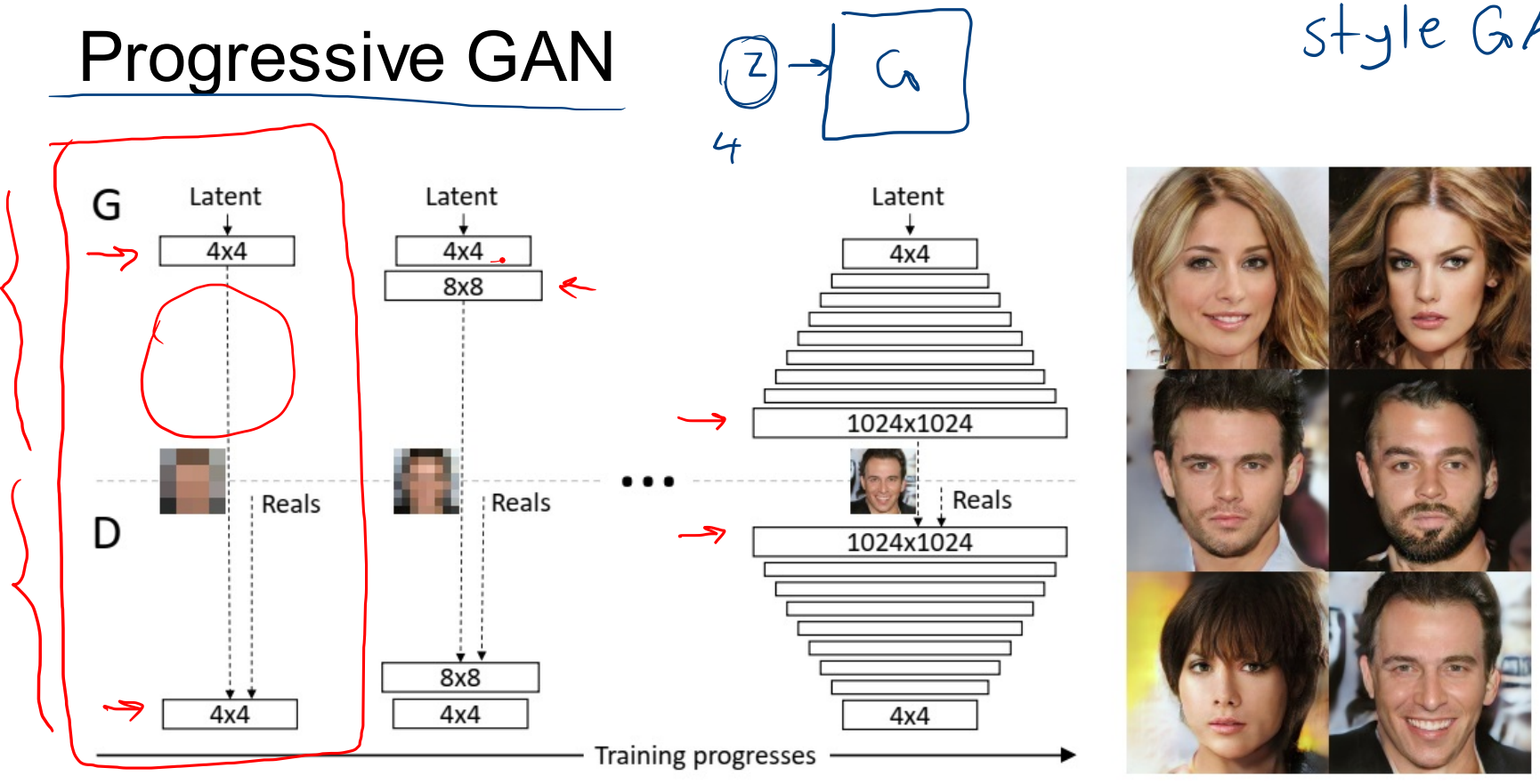


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

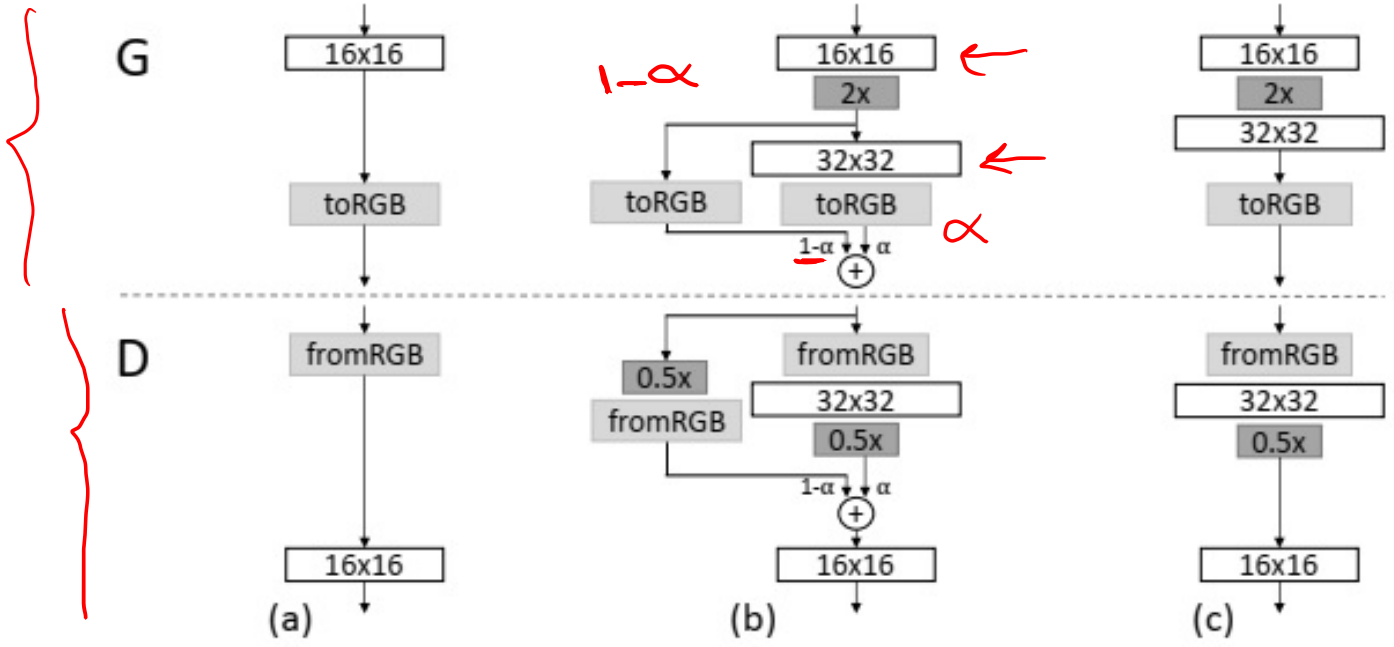


Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16×16 images (a) to 32×32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. Here $2\times$ and $0.5\times$ refer to doubling and halving the image resolution using nearest neighbor filtering and average pooling, respectively. The toRGB represents a layer that projects feature vectors to RGB colors and fromRGB does the reverse; both use 1×1 convolutions. When training the discriminator, we feed in real images that are downscaled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions.

StyleGAN (1, 2, 3)