Generative Adversarial Networks

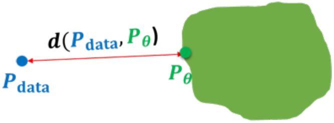
Stefano Ermon, Aditya Grover

Stanford University

Lecture 9

Recap

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



$\theta \in M$

Model family

- Model families
    - Autoregressive Models: $p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i|\mathbf{x}_{<i})$
    - Variational Autoencoders: $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$
    - Normalizing Flow Models: $p_X(\mathbf{x}; \theta) = p_Z\left(\mathbf{f}_\theta^{-1}(\mathbf{x})\right) \left|\det\left(\frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right)\right|$
- All the above families are based on maximizing likelihoods (or approximations)
- Is the likelihood a good indicator of the quality of samples generated by the model?

Towards likelihood-free learning

- **Case 1:** Optimal generative model will give best **sample quality** and highest test **log-likelihood**
- For imperfect models, achieving high log-likelihoods might not always imply good sample quality, and vice-versa (Theis et al., 2016)

Towards likelihood-free learning

- **Case 2:** Great test log-likelihoods, poor samples. E.g., For a discrete noise mixture model $p_\theta(\mathbf{x}) = 0.01 p_{\text{data}}(\mathbf{x}) + 0.99 p_{\text{noise}}(\mathbf{x})$
  - 99% of the samples are just noise
  - Taking logs, we get a lower bound

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \log[0.01 p_{\text{data}}(\mathbf{x}) + 0.99 p_{\text{noise}}(\mathbf{x})] \\ &\geq \log 0.01 p_{\text{data}}(\mathbf{x}) = \log p_{\text{data}}(\mathbf{x}) - \log 100\end{aligned}$$

  - For expected likelihoods, we know that
    - Lower bound

$$E_{p_{\text{data}}}[\log p_\theta(\mathbf{x})] \geq E_{p_{\text{data}}}[\log p_{\text{data}}(\mathbf{x})] - \log 100$$

    - Upper bound (via non-negativity of KL)

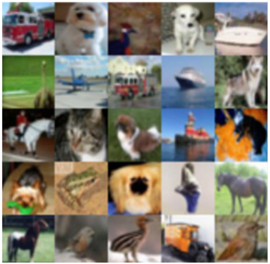
$$E_{p_{\text{data}}}[\log p_{\text{data}}(\mathbf{x}))] \geq E_{p_{\text{data}}}[\log p_\theta(\mathbf{x})]$$

  - As we increase the dimension of $\mathbf{x}$, absolute value of $\log p_{\text{data}}(\mathbf{x})$ increases proportionally but $\log 100$ remains constant. Hence, $E_{p_{\text{data}}}[\log p_\theta(\mathbf{x})] \approx E_{p_{\text{data}}}[\log p_{\text{data}}(\mathbf{x})]$ in very high dimensions

Towards likelihood-free learning

- **Case 3:** Great samples, poor test log-likelihoods. E.g., Memorizing training set
  - Samples look exactly like the training set (cannot do better!)
  - Test set will have zero probability assigned (cannot do worse!)
- The above cases suggest that it might be useful to disentangle likelihoods and samples
- **Likelihood-free learning** consider objectives that do not depend directly on a likelihood function

Comparing distributions via samples



vs.

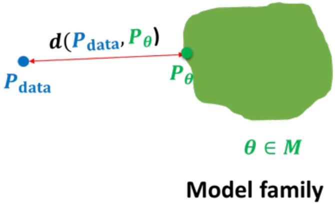
$S_1 = \{\mathbf{x} \sim P\}$ $\qquad$ $S_2 = \{\mathbf{x} \sim Q\}$

Given a finite set of samples from two distributions $S_1 = \{\mathbf{x} \sim P\}$ and $S_2 = \{\mathbf{x} \sim Q\}$, how can we tell if these samples are from the same distribution? (i.e., $P = Q$?)

Two-sample tests

- Given $S_1 = \{\mathbf{x} \sim P\}$ and $S_2 = \{\mathbf{x} \sim Q\}$, a **two-sample test** considers the following hypotheses
  - Null hypothesis H_0: $P = Q$
  - Alternate hypothesis H_1: $P \neq Q$
- Test statistic T compares S_1 and S_2 e.g., difference in means, variances of the two sets of samples
- If T is less than a threshold α, then accept H_0 else reject it
- **Key observation:** Test statistic is **likelihood-free** since it does not involve the densities P or Q (only samples)

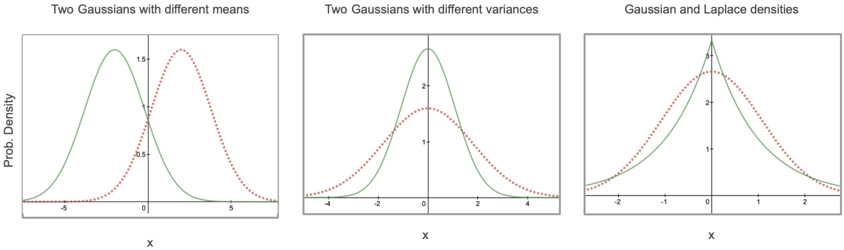
Generative modeling and two-sample tests



- Apriori we assume direct access to $S_1 = \mathcal{D} = \{\mathbf{x} \sim p_{\text{data}}\}$
- In addition, we have a model distribution p_θ
- Assume that the model distribution permits efficient sampling (e.g., directed models). Let $S_2 = \{\mathbf{x} \sim p_\theta\}$
- **Alternate notion of distance between distributions:** Train the generative model to minimize a two-sample test objective between S_1 and S_2

Two-Sample Test via a Discriminator

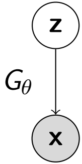
- Finding a two-sample test objective in high dimensions is hard



- In the generative model setup, we know that S_1 and S_2 come from different distributions p_{data} and p_θ respectively
- **Key idea: Learn** a statistic that **maximizes** a suitable notion of distance between the two sets of samples S_1 and S_2

Generative Adversarial Networks

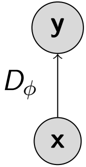
- A two player minimax game between a **generator** and a **discriminator**



- **Generator**
  - Directed, latent variable model with a deterministic mapping between $\mathbf{z}$ and $\mathbf{x}$ given by G_θ
  - Minimizes a two-sample test objective (in support of the null hypothesis $p_{\mathrm{data}} = p_\theta$)

Generative Adversarial Networks

- A two player minimax game between a generator and a discriminator



- **Discriminator**
    - Any function (e.g., neural network) which tries to distinguish "real" samples from the dataset and "fake" samples generated from the model
    - Maximizes the two-sample test objective (in support of the alternate hypothesis $p_{\mathrm{data}} \neq p_\theta$)

Example of GAN objective

- **Training objective for discriminator**:

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

- For a fixed generator G, the discriminator is performing binary classification with the cross entropy objective
  - Assign probability 1 to true data points $\mathbf{x} \sim p_{\mathrm{data}}$
  - Assing probability 0 to fake samples $\mathbf{x} \sim p_G$
- Optimal discriminator

$$D_G^*(\mathbf{x}) = \frac{p_{\mathrm{data}}(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x}) + p_G(\mathbf{x})}$$

Example of GAN objective

- **Training objective for generator**:

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

- For the optimal discriminator $D^*_G(\cdot)$, we have

$$
\begin{aligned}
& V(G, D^*_G(\mathbf{x})) \\
&= E_{\mathbf{x} \sim p_{\mathrm{data}}}\left[\log \frac{p_{\mathrm{data}}(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x}) + p_G(\mathbf{x})}\right] + E_{\mathbf{x} \sim p_G}\left[\log \frac{p_G(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x}) + p_G(\mathbf{x})}\right] \\
&= E_{\mathbf{x} \sim p_{\mathrm{data}}}\left[\log \frac{p_{\mathrm{data}}(\mathbf{x})}{\frac{p_{\mathrm{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}}\right] + E_{\mathbf{x} \sim p_G}\left[\log \frac{p_G(\mathbf{x})}{\frac{p_{\mathrm{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}}\right] - \log 4 \\
&= \underbrace{D_{KL}\left[p_{\mathrm{data}}, \frac{p_{\mathrm{data}} + p_G}{2}\right] + D_{KL}\left[p_G, \frac{p_{\mathrm{data}} + p_G}{2}\right]}_{2 \times \text{Jenson-Shannon Divergence (JSD)}} - \log 4 \\
&= 2 D_{JSD}[p_{\mathrm{data}}, p_G] - \log 4
\end{aligned}
$$

Jenson-Shannon Divergence

- Also called as the symmetric KL divergence

$$D_{JSD}[p,q] = \frac{1}{2}\left(D_{KL}\left[p, \frac{p+q}{2}\right] + D_{KL}\left[q, \frac{p+q}{2}\right]\right)$$

- Properties
  - $D_{JSD}[p,q] \geq 0$
  - $D_{JSD}[p,q] = 0$ iff $p = q$
  - $D_{JSD}[p,q] = D_{JSD}[q,p]$
  - $\sqrt{D_{JSD}[p,q]}$ satisfies triangle inequality $\rightarrow$ Jenson-Shannon Distance
- Optimal generator for the JSD/Negative Cross Entropy GAN

$$p_G = p_{\mathrm{data}}$$

- For the optimal discriminator $D^*_{G^*}(\cdot)$ and generator $G^*(\cdot)$, we have

$$V(G^*, D^*_{G^*}(\mathbf{x})) = -\log 4$$

The GAN training algorithm

- Sample minibatch of m training points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$ from $\mathcal{D}$
- Sample minibatch of m noise vectors $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)}$ from p_z
- Update the generator parameters θ by stochastic gradient **descent**

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m}\nabla_\theta \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))$$

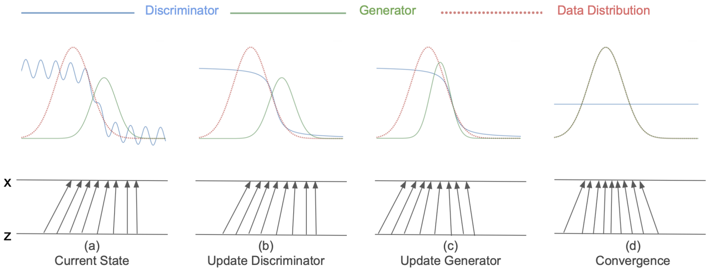
- Update the discriminator parameters ϕ by stochastic gradient **ascent**

$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m}\nabla_\phi \sum_{i=1}^{m} [\log D_\phi(\mathbf{x}^{(i)}) + \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))]$$

- Repeat for fixed number of epochs

Alternating optimization in GANs

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



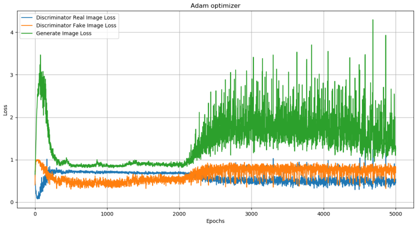
Frontiers in GAN research

- GANs have been successfully applied to several domains and tasks
- However, working with GANs can be very challenging in practice
  - Unstable optimization
  - Mode collapse
  - Evaluation
- Many bag of tricks applied to train GANs successfully

Image Source: Ian Goodfellow. Samples from Goodfellow et al., 2014, Radford et al., 2015, Liu et al., 2016, Karras et al., 2017, Karras et al., 2018

Optimization challenges

- **Theorem (informal):** If the generator updates are made in function space and discriminator is optimal at every step, then the generator is guaranteed to converge to the data distribution
- **Unrealistic assumptions**!
- In practice, the generator and discriminator loss keeps oscillating during GAN training

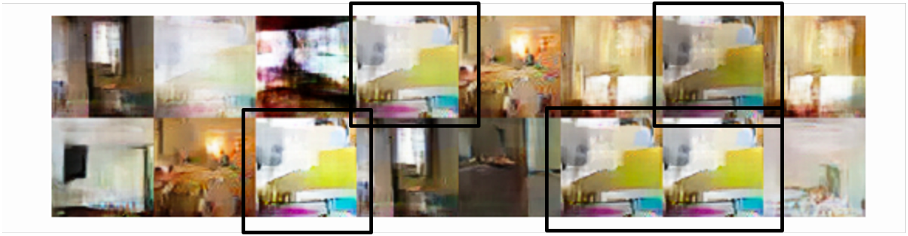


Source: Mirantha Jayathilaka

- No robust stopping criteria in practice (unlike MLE)

Mode Collapse

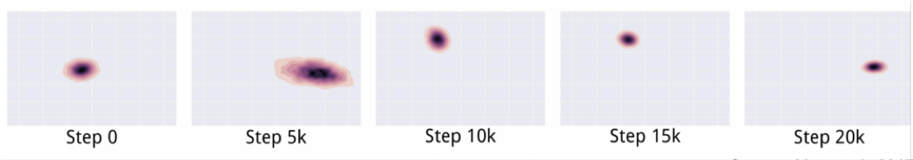
- GANs are notorious for suffering from **mode collapse**
- Intuitively, this refers to the phenomena where the generator of a GAN collapses to one or few samples (dubbed as "modes")



Arjovsky et al., 2017

Mode Collapse

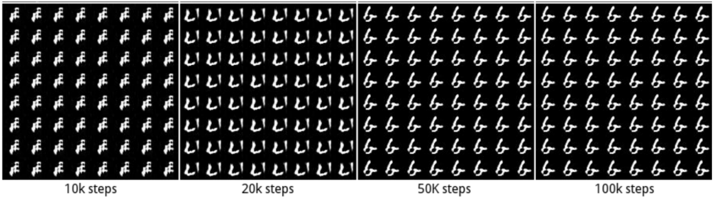
- True distribution is a mixture of Gaussians



Source: Metz et al., 2017

- The generator distribution keeps oscillating between different modes

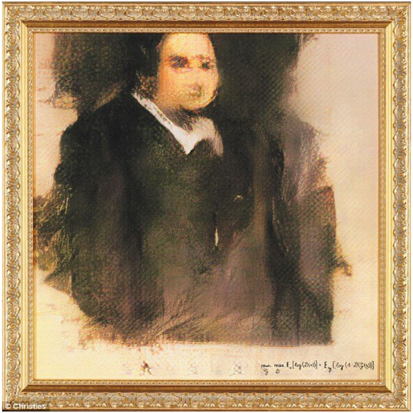
Mode Collapse



Source: Metz et al., 2017

- Fixes to mode collapse are mostly empirically driven: alternate architectures, adding regularization terms, injecting small noise perturbations etc.
- `https://github.com/soumith/ganhacks`
  How to Train a GAN? Tips and tricks to make GANs work by Soumith Chintala

Beauty lies in the eyes of the discriminator



Source: Robbie Barrat, Obvious

GAN generated art auctioned at Christie's.
Expected Price: $\$7,000 - \$10,000$
True Price: $\$432,500$

Generative Adversarial Networks

Stefano Ermon, Aditya Grover

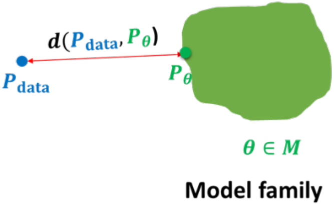
Stanford University

Lecture 10

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
  - Inferring latent representations via BiGAN
  - Application: Image-to-image translation via CycleGANs

Beyond KL and Jenson-Shannon Divergence



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

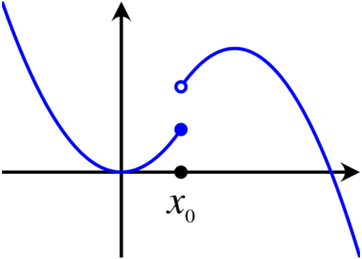
- KL divergence: Autoregressive Models, Flow models
- (scaled and shifted) Jenson-Shannon divergence: original GAN objective

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 $f(\mathbf{x}_0)$



- Example: KL divergence with $f(u) = u \log u$

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 ($\alpha\notin\{0,1\}$)	$\frac{1}{\alpha(\alpha-1)}\int\left(p(x)\left[\left(\frac{q(x)}{p(x)}\right)^\alpha-1\right]-\alpha(q(x)-p(x))\right)\,\mathrm{d}x$	$\frac{1}{\alpha(\alpha-1)}(u^\alpha-1-\alpha(u-1))$

Source: Nowozin et al., 2016

f-GAN: Variational Divergence Minimization

- To use f-divergences as a two-sample test objective for likelihood-free learning, we need to be able to estimate it only via samples
- Fenchel conjugate: For any function $f(\cdot)$, its convex conjugate is defined as
$$f^*(t) = \sup_{u \in \mathrm{dom}_f} (ut - f(u))$$
- Duality: $f^{**} = f$. When $f(\cdot)$ is convex, lower semicontinous, so is $f^*(\cdot)$
$$f(u) = \sup_{t \in \mathrm{dom}_{f^*}} (tu - f^*(t))$$

f-GAN: Variational Divergence Minimization

- We can obtain a lower bound to any f-divergence via its Fenchel conjugate

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

  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

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

- 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

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

- Generator G_θ tries to minimize the divergence estimate and discriminator T_ϕ tries to tighten the lower bound

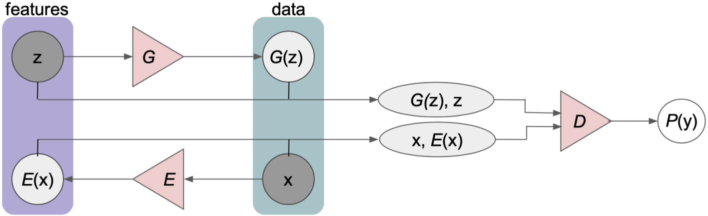
Inferring latent representations in GANs

- 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

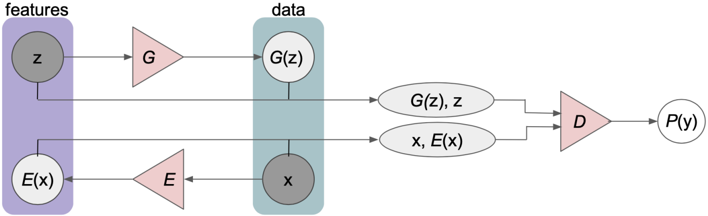
- If we want to directly infer the latent variables $\mathbf{z}$ of the generator, we need a different learning algorithm
- A regular GAN optimizes a two-sample test objective that compares samples of $\mathbf{x}$ from the generator and the data distribution
- **Solution 2**: To infer latent representations, 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)

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 $\mathbf{x} \sim p_{\mathrm{data}}(\mathbf{x})$ 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 $G : \mathbf{z} \mapsto \mathbf{x}$

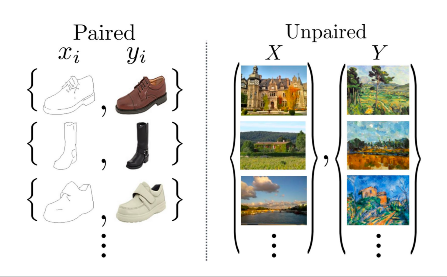
Bidirectional Generative Adversarial Networks (BiGAN)



- The discriminator D observes samples from the generative model $\mathbf{z}, G(\mathbf{z})$ and the encoding distribution $E(\mathbf{x}), \mathbf{x}$
- The goal of the discriminator is to maximize the two-sample test objective between $\mathbf{z}, G(\mathbf{z})$ and $E(\mathbf{x}), \mathbf{x}$
- After training is complete, new samples are generated via G and latent representations are inferred via E

Translating across domains

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

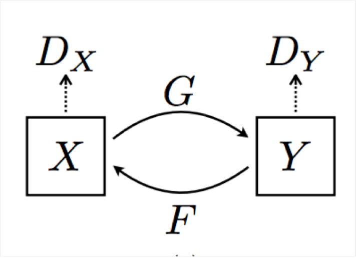


Source: Zhu et al., 2016

- 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

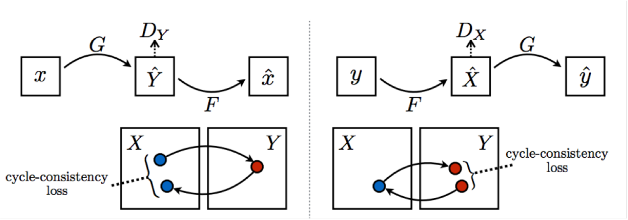
- To match the two distributions, we learn two parameterized conditional generative models $G : \mathcal{X} \leftrightarrow \mathcal{Y}$ and $F : \mathcal{Y} \leftrightarrow \mathcal{X}$
- G maps an element of $\mathcal{X}$ to an element of $\mathcal{Y}$. A discriminator $D_{\mathcal{Y}}$ compares the 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: $G(F(Y)) \approx Y$

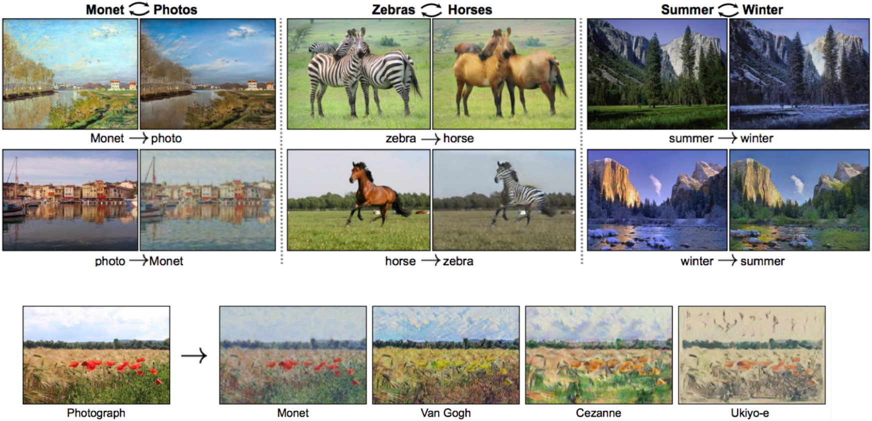


Source: Zhu et al., 2016

- Overall loss function

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

CycleGAN in practice



Source: Zhu et al., 2016

AlignFlow

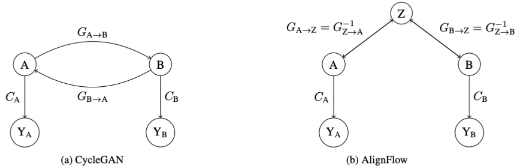


Figure 1: CycleGAN v.s. AlignFlow for unpaired cross-domain translation. Unlike CycleGAN, AlignFlow specifies a single invertible mapping $G_{A\to Z} \circ G^{-1}_{B\to Z}$ that is exactly cycle-consistent, represents a shared latent space Z between the two domains, and can be trained via both adversarial 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

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 more)
- Latent representations can be inferred via BiGAN
- Cycle-consistent domain translations via CycleGAN and AlignFlow