Deep Generative Models

Thomas Lucas & Jakob Verbeek

Content of this talk

Focus of this talk: Latent variable deep generative models

- Part I: Generative adversarial networks (GANs)
- Part II: Variational auto-encoders (VAEs)

Latent variable generative models

Reminder:

- A latent variable generative model defines $p_\theta(x) = \int_z p(z)p_\theta(x|z)dz$
- A VAE maximises a lower bound of $\mathbb{E}_{x\sim p^*}(log(p_\theta(x)))$ called the ELBO
- A GAN avoids intractable integrals using a discriminator

Training support VS sample quality

GANs and VAEs have complementary strengths and weaknesses

Training support VS sample quality

GANs and VAEs have complementary strengths and weaknesses

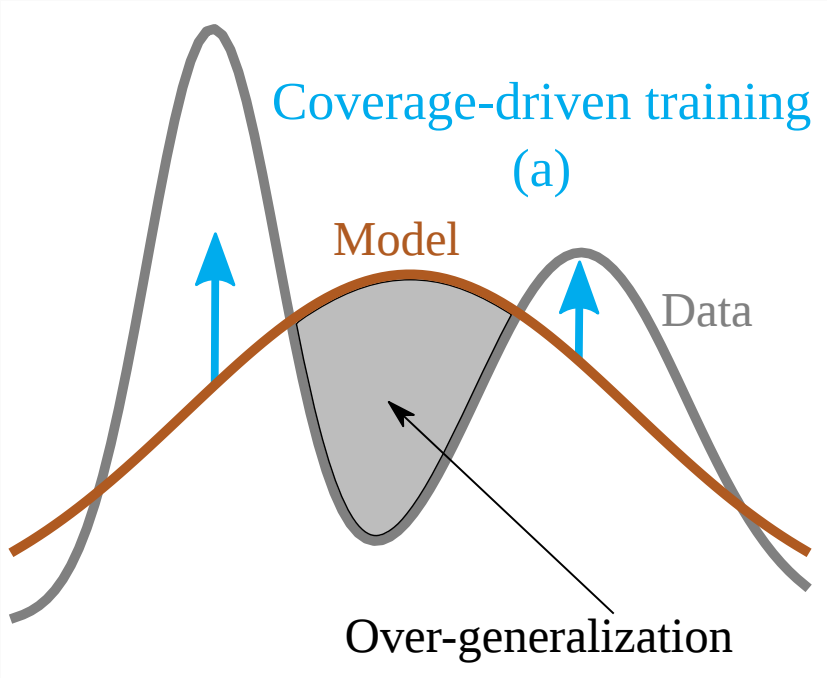
- Maximum-likelihood: sample from p^*, evaluate probability under p_θ.
    - Explicitely optimize coverage of training points
    - implicitely optimize sample quality (density is **normalized**)

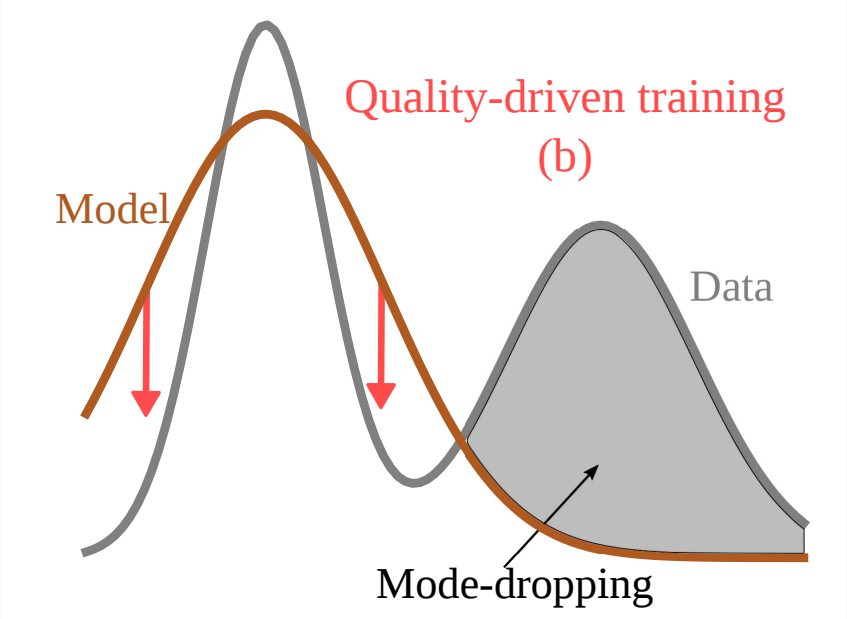
Training support VS sample quality

GANs and VAEs have complementary strengths and weaknesses

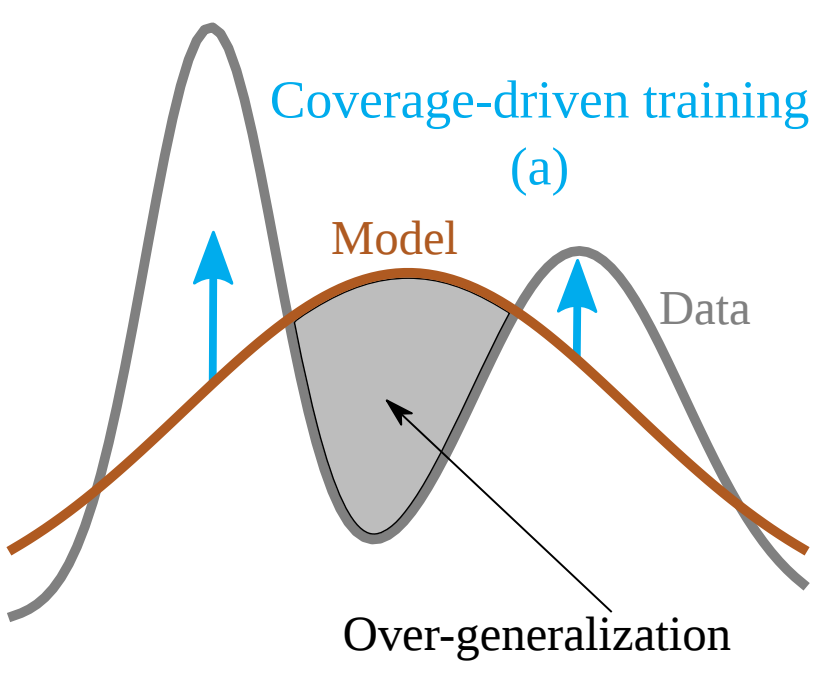
- Maximum-likelihood: sample from p^*, evaluate probability under p_θ.
  - Explicitely optimize coverage of training points
  - implicitely optimize sample quality (density is **normalized**)
- Adversarial training: sample from p_θ, "evaluate" probability under p^*.
  - implicitely optimize coverage of training points (**normalise**)
  - Explicitely optimize sample quality

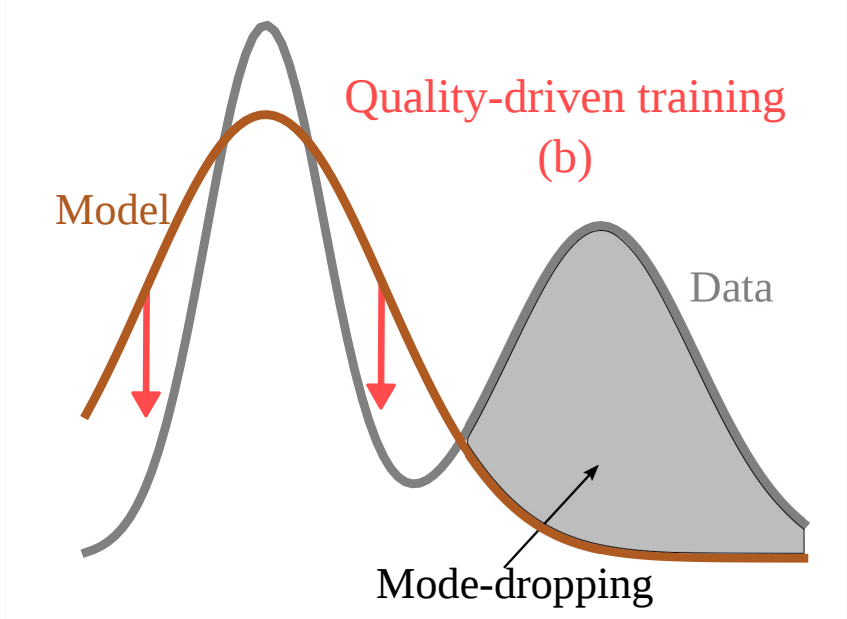
Different paradigms





Different paradigms

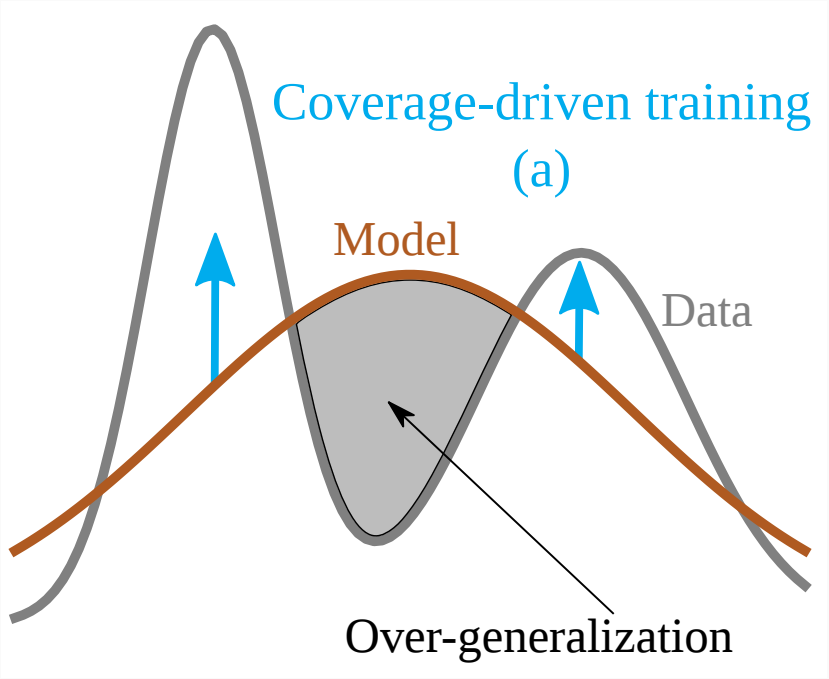


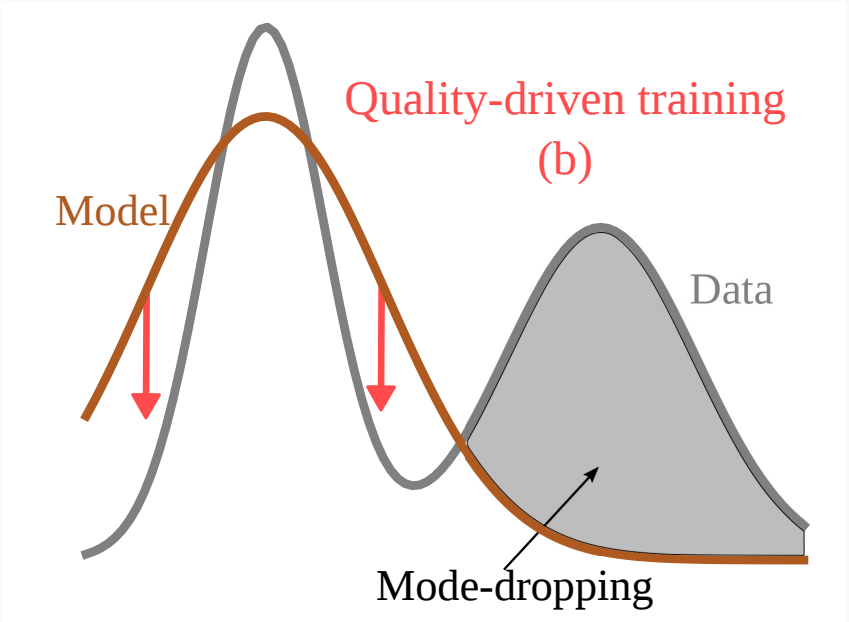


If p_θ is less flexible than p^*:

- Maximum-likelihood: p_θ goes through regions of low density: Over-generalization
- Adversarial: p_θ drops some modes to avoid low density: mode-dropping

Different paradigms





From this perspective:

- You can expect GANs to produce better looking images
- You can expect VAEs to better compress the training data

Some research directions

Improving on VAEs and GANs:

- Better losses for GANs: encourage full support, improve stability.
- Avoiding poor assumptions in VAEs to reduce over-generalization.

Part I

Stabilizing GAN training

A discussion on the GAN training loss

A discussion on the GAN training loss

- Recall divergence measures between distributions

A discussion on the GAN training loss

- Recall divergence measures between distributions
- Kullback-Leibler divergence: maximum likelihood training
  - Infinite if q (model) has a zero in the support of p (data)

$$D_{KL}(p||q) = \int_x p(x)\big[\ln q(x) - \ln p(x)\big] \tag{1}$$

A discussion on the GAN training loss

- Recall divergence measures between distributions
- Kullback-Leibler divergence: maximum likelihood training
  - Infinite if q (model) has a zero in the support of p (data)

$$D_{KL}(p||q) = \int_x p(x)\big[\ln q(x) - \ln p(x)\big] \tag{1}$$

- Jensen-Shannon divergence: idealized loss approximated by the discriminator
  - Symmetric KL to mixture of p and q

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}\left(p\middle|\middle|\frac{p+q}{2}\right) + \frac{1}{2}D_{KL}\left(q\middle|\middle|\frac{p+q}{2}\right) \tag{2}$$

A discussion on the GAN training loss

- Training loss for the Discriminator:

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))] \quad (3)$$

A discussion on the GAN training loss

- Training loss for the Discriminator:

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))] \quad (3)$$

A discussion on the GAN training loss

- Training loss for the Discriminator:

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))] \quad (3)$$

- Approximates the ideal loss:

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}\left(p \middle|\middle| \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \middle|\middle| \frac{p+q}{2}\right) \quad (4)$$

A discussion on the GAN training loss

- Training loss for the Discriminator:

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))] \quad (3)$$

- Approximates the ideal loss:

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}\left(p \middle|\middle| \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \middle|\middle| \frac{p+q}{2}\right) \quad (4)$$

A discussion on the GAN training loss

- Training loss for the Discriminator:

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))] \quad (3)$$

- Approximates the ideal loss:

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}\left(p \middle|\middle| \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \middle|\middle| \frac{p+q}{2}\right) \quad (4)$$

- The blue term is independant from the model p_θ, disapears when differentiating

A discussion on the GAN training loss

- Training loss for the Discriminator:

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))] \quad (3)$$

- Approximates the ideal loss:

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}\left(p \middle|\middle| \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \middle|\middle| \frac{p+q}{2}\right) \quad (4)$$

- The blue term is independant from the model p_θ, disapears when differentiating
- The generator is trained on the red term.

Quality driven training

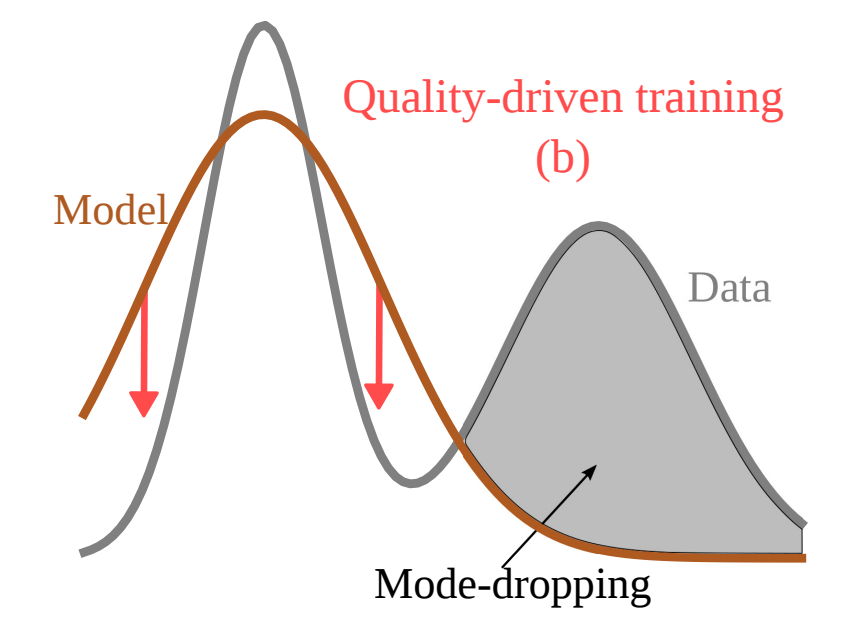
- Training loss for the generator: $D_{KL}\left(q \middle|\middle| \frac{p+q}{2}\right)$

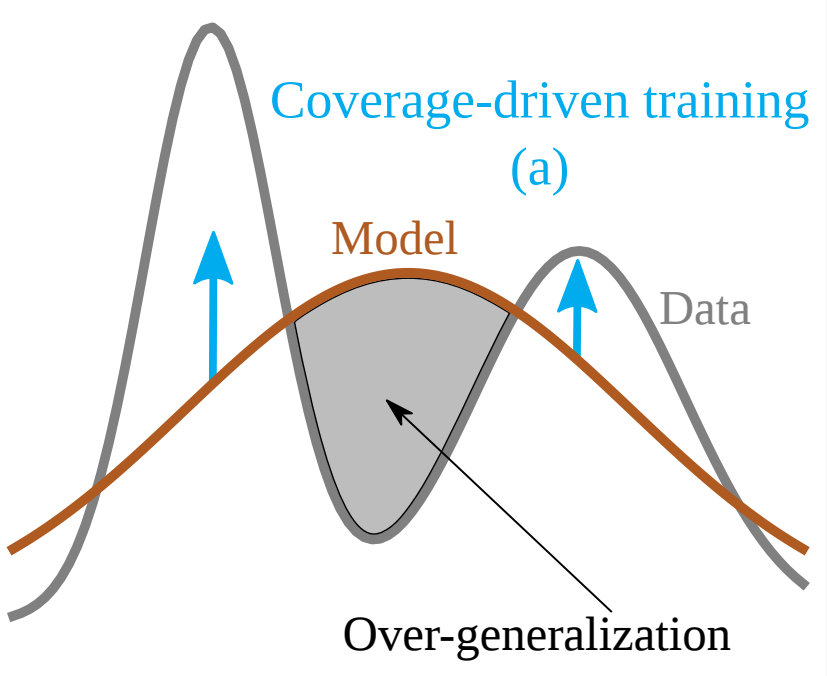
Quality driven training

- Training loss for the generator: $D_{KL}\left(q \middle|\middle| \frac{p+q}{2}\right)$
- It is an integral on q, 'symmetric' to maximum-likelihood estimation

Quality driven training

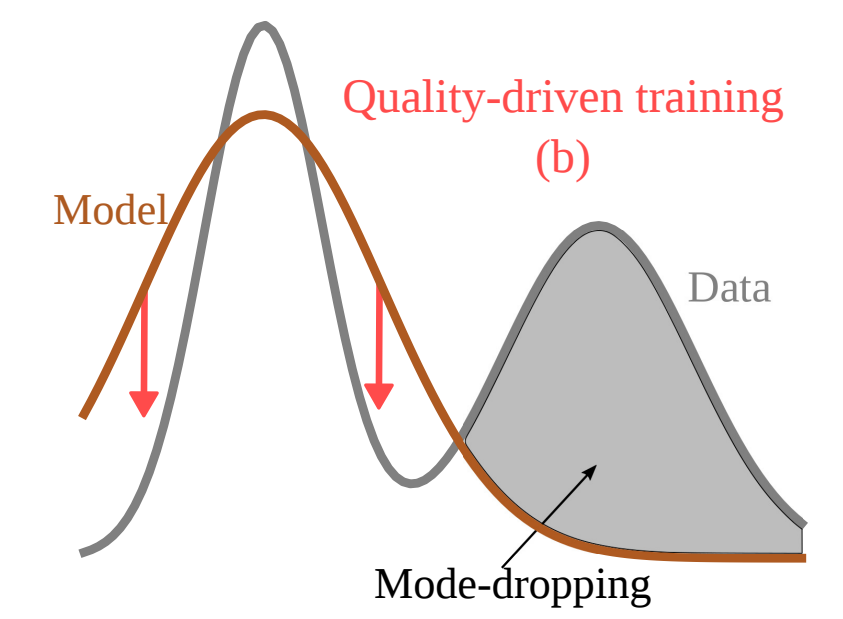
- Training loss for the generator: $D_{KL}\left(q \middle|\middle| \frac{p+q}{2}\right)$
- It is an integral on q, 'symmetric' to maximum-likelihood estimation



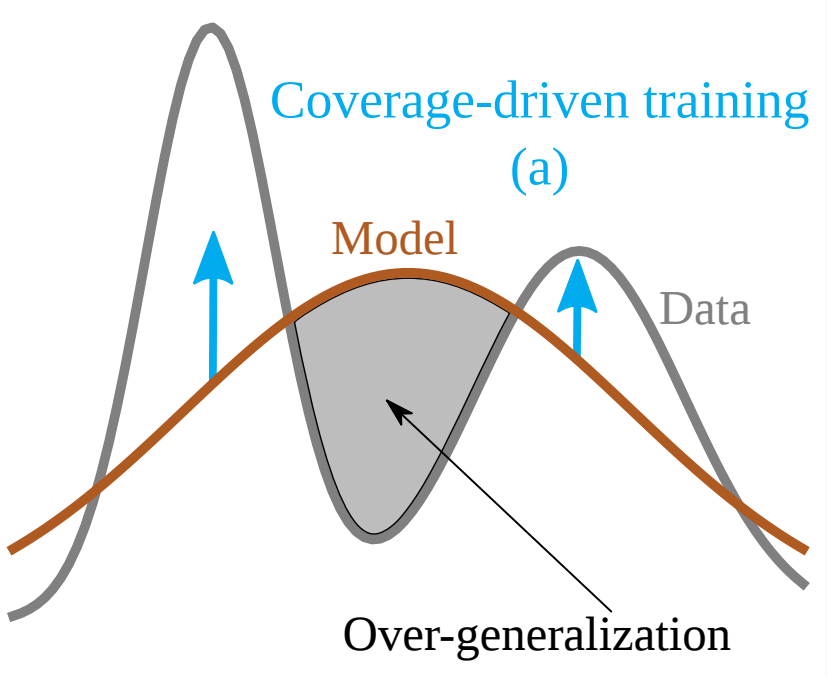


Quality driven training

- Training loss for the generator: $D_{KL}\left(q\middle\|\frac{p+q}{2}\right)$
- It is an integral on q, 'symmetric' to maximum-likelihood estimation



$\frac{1}{2}D_{KL}\left(q\middle\|\frac{p+q}{2}\right)$



$\frac{1}{2}D_{KL}\left(p\middle\|\frac{p+q}{2}\right)$

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - (see drawing on the board)

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - (see drawing on the board)
   - Happens early in training with poor generator

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - (see drawing on the board)
   - Happens early in training with poor generator
   - Tuning of capacity and training regime of discriminator

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - (see drawing on the board)
   - Happens early in training with poor generator
   - Tuning of capacity and training regime of discriminator

2. Minimizing $-\mathbb{E}_{p_z}[\ln(D(G(z)))]$ instead to boost gradient

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - (see drawing on the board)
   - Happens early in training with poor generator
   - Tuning of capacity and training regime of discriminator

2. Minimizing $-\mathbb{E}_{p_z}[\ln(D(G(z)))]$ instead to boost gradient
   - Optimizes $KL(p_G||p_{\text{data}}) - 2JS(p_G||p_{\text{data}})$

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - (see drawing on the board)
   - Happens early in training with poor generator
   - Tuning of capacity and training regime of discriminator

2. Minimizing $-\mathbb{E}_{p_z}[\ln(D(G(z)))]$ instead to boost gradient
   - Optimizes $KL(p_G||p_{\text{data}}) - 2JS(p_G||p_{\text{data}})$
   - Wrong sign in the JS divergence

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - (see drawing on the board)
   - Happens early in training with poor generator
   - Tuning of capacity and training regime of discriminator

2. Minimizing $-\mathbb{E}_{p_z}[\ln(D(G(z)))]$ instead to boost gradient
   - Optimizes $KL(p_G||p_{\text{data}}) - 2JS(p_G||p_{\text{data}})$
   - Wrong sign in the JS divergence
   - Same stable points in the minimax optimization

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - (see drawing on the board)
   - Happens early in training with poor generator
   - Tuning of capacity and training regime of discriminator

2. Minimizing $-\mathbb{E}_{p_z}[\ln(D(G(z)))]$ instead to boost gradient
   - Optimizes $KL(p_G||p_{\text{data}}) - 2JS(p_G||p_{\text{data}})$
   - Wrong sign in the JS divergence
   - Same stable points in the minimax optimization
   - Helps early, but problem remains: if D_ϕ becomes too good, gradients vanish

Question:

Can we think of a better 'ideal loss'?

Wasserstein or "earth-mover" distance

- Consider joint distribution $\gamma(x, y)$
  with marginals $p(x) = \gamma(x)$ and $q(y) = \gamma(y)$

Wasserstein or "earth-mover" distance

- Consider joint distribution $\gamma(x, y)$
  with marginals $p(x) = \gamma(x)$ and $q(y) = \gamma(y)$
- Conditional $\gamma(y|x)$ "moves mass" to transform $p(\cdot)$ into $q(\cdot)$

Wasserstein or "earth-mover" distance

- Consider joint distribution $\gamma(x, y)$
  with marginals $p(x) = \gamma(x)$ and $q(y) = \gamma(y)$
- Conditional $\gamma(y|x)$ "moves mass" to transform $p(\cdot)$ into $q(\cdot)$
- Cost associated with a given transformation

$$T(\gamma) = \int_{x,y} \gamma(x, y) \, ||x - y|| = \int_x p(x) \int_y \gamma(y|x) \, ||x - y||$$

Wasserstein or "earth-mover" distance

- Consider joint distribution $\gamma(x, y)$
  with marginals $p(x) = \gamma(x)$ and $q(y) = \gamma(y)$
- Conditional $\gamma(y|x)$ "moves mass" to transform $p(\cdot)$ into $q(\cdot)$
- Cost associated with a given transformation

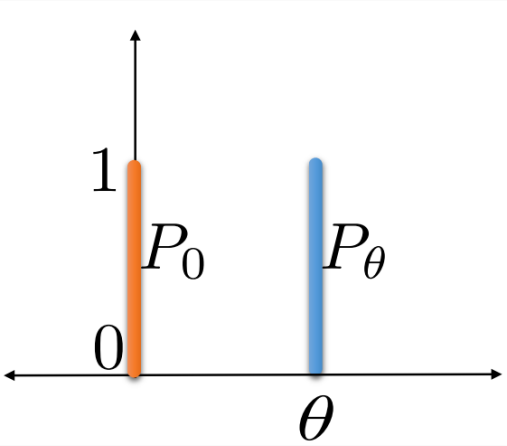
$$T(\gamma) = \int_{x,y} \gamma(x, y) \, ||x - y|| = \int_x p(x) \int_y \gamma(y|x) \, ||x - y||$$

- Wasserstein distance is the cost of optimal transformation

$$D_{WS}(p||q) = \inf_{\gamma \in \Gamma(p,q)} T(\gamma) \tag{5}$$

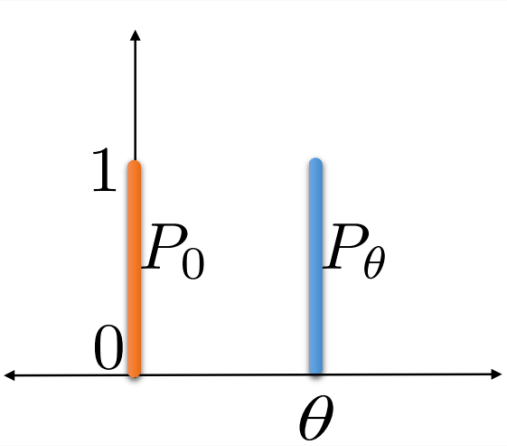
Distributions with low dimensional support

- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$



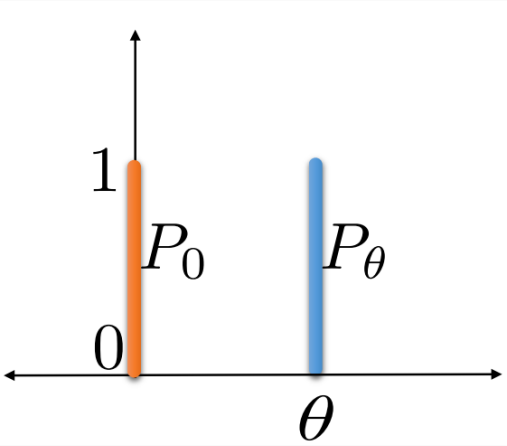
Distributions with low dimensional support

- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$



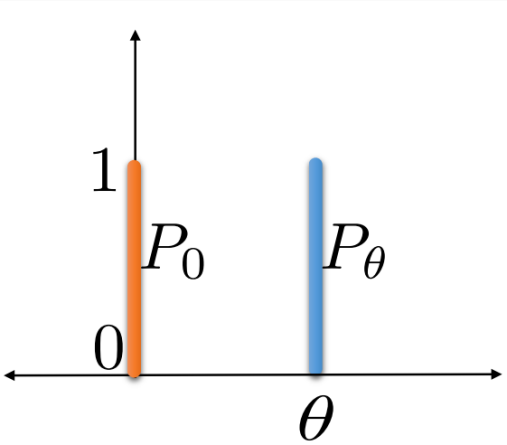
Distributions with low dimensional support

- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$
  - $D_{KL}(p_0 || p_\theta) = \infty$



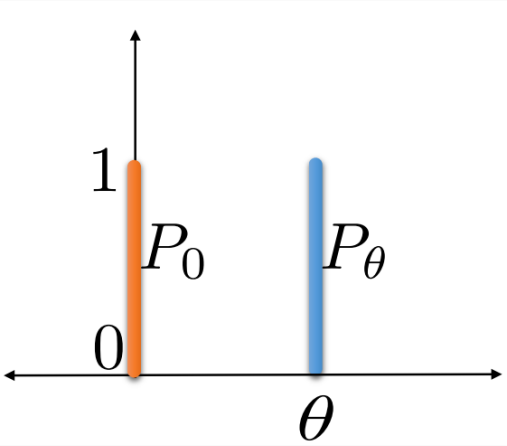
Distributions with low dimensional support

- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$
  - $D_{KL}(p_0||p_\theta) = \infty$
  - $D_{JS}(p_0||p_\theta) = \ln 2$



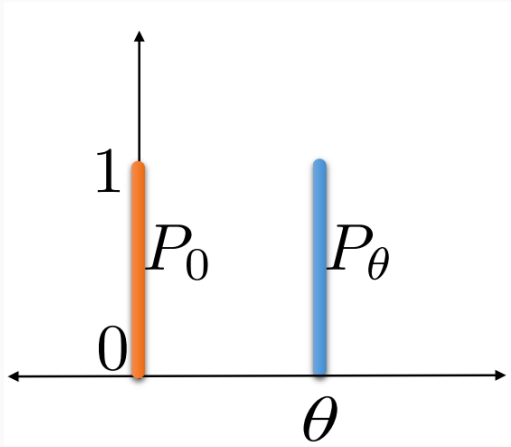
Distributions with low dimensional support

- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$
  - $D_{KL}(p_0||p_\theta) = \infty$
  - $D_{JS}(p_0||p_\theta) = \ln 2$
  - $D_{WS}(p_0||p_\theta) = |\theta|$



Distributions with low dimensional support

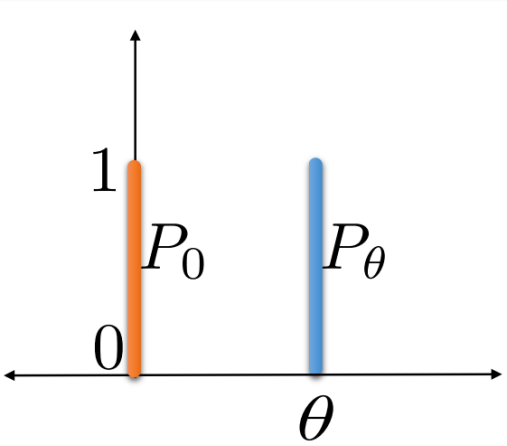
- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$
  - $D_{KL}(p_0||p_\theta) = \infty$
  - $D_{JS}(p_0||p_\theta) = \ln 2$
  - $D_{WS}(p_0||p_\theta) = |\theta|$



- Wasserstein based on proximity of support

Distributions with low dimensional support

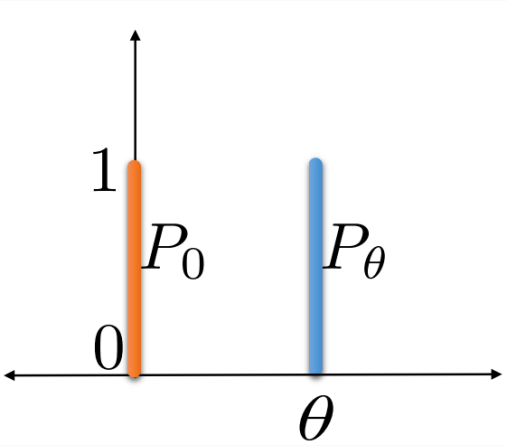
- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$
  - $D_{KL}(p_0||p_\theta) = \infty$
  - $D_{JS}(p_0||p_\theta) = \ln 2$
  - $D_{WS}(p_0||p_\theta) = |\theta|$



- Wasserstein based on proximity of support
- JS and KL based on overlap of support

Distributions with low dimensional support

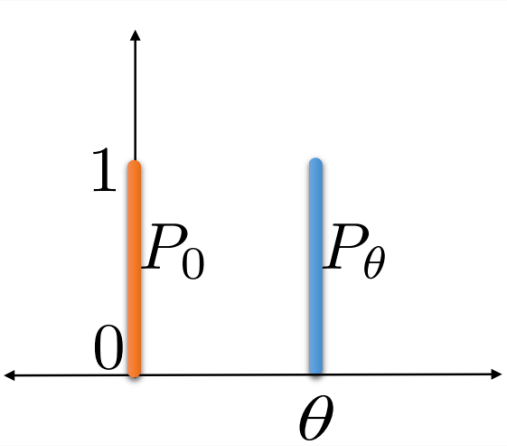
- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$
  - $D_{KL}(p_0||p_\theta) = \infty$
  - $D_{JS}(p_0||p_\theta) = \ln 2$
  - $D_{WS}(p_0||p_\theta) = |\theta|$



- Wasserstein based on proximity of support
- JS and KL based on overlap of support
  - In general measure zero overlap with low dim. supports

Distributions with low dimensional support

- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$
  - $D_{KL}(p_0||p_\theta) = \infty$
  - $D_{JS}(p_0||p_\theta) = \ln 2$
  - $D_{WS}(p_0||p_\theta) = |\theta|$



- Wasserstein based on proximity of support
- JS and KL based on overlap of support
  - In general measure zero overlap with low dim. supports
  - GAN has support with dimension of latent variable **z**

Wasserstein GAN

- Dual formulation of Wasserstein distance

$$D_{WS}(p_data||q) = \inf_{\gamma \in \Gamma(p,q)} T(\gamma) \tag{6}$$

$$= \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))] \tag{7}$$

Wasserstein GAN

- Dual formulation of Wasserstein distance

$$D_{WS}(p_data||q) = \inf_{\gamma \in \Gamma(p,q)} T(\gamma) \tag{6}$$

$$= \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))] \tag{7}$$

1. $||.||_L$ is the lipschitz norm

Wasserstein GAN

- Dual formulation of Wasserstein distance

$$D_{WS}(p_data||q) = \inf_{\gamma \in \Gamma(p,q)} T(\gamma) \tag{6}$$

$$= \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))] \tag{7}$$

1. $||.||_L$ is the lipschitz norm
2. In practice: restrict D to some deep net architecture

Wasserstein GAN

- Dual formulation of Wasserstein distance

$$D_{WS}(p_data||q) = \inf_{\gamma \in \Gamma(p,q)} T(\gamma) \tag{6}$$

$$= \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))] \tag{7}$$

1. $||.||_L$ is the lipschitz norm
2. In practice: restrict D to some deep net architecture
3. Enforce Lipschitz constraint by clipping discriminator weights or penalty on gradient magnitude [Gulrajani et al., 2017a]

Wasserstein GAN

- Dual formulation of Wasserstein distance

$$D_{WS}(p_data||q) = \inf_{\gamma \in \Gamma(p,q)} T(\gamma) \tag{6}$$

$$= \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))] \tag{7}$$

  1. $||.||_L$ is the lipschitz norm
  2. In practice: restrict D to some deep net architecture
  3. Enforce Lipschitz constraint by clipping discriminator weights or penalty on gradient magnitude [Gulrajani et al., 2017a]

- Removes log-sigmoid transformation w.r.t. normal GAN

Wasserstein GAN

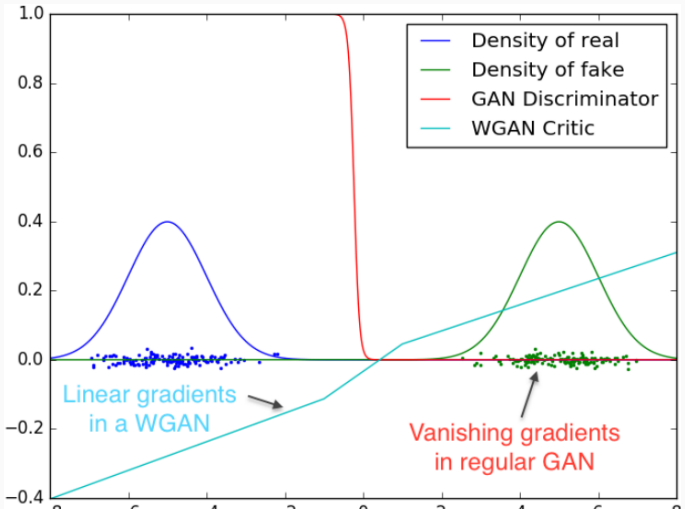
- Dual formulation of Wasserstein distance

$$D_{WS}(p_data||q) = \inf_{\gamma \in \Gamma(p,q)} T(\gamma) \quad (6)$$

$$= \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))] \quad (7)$$

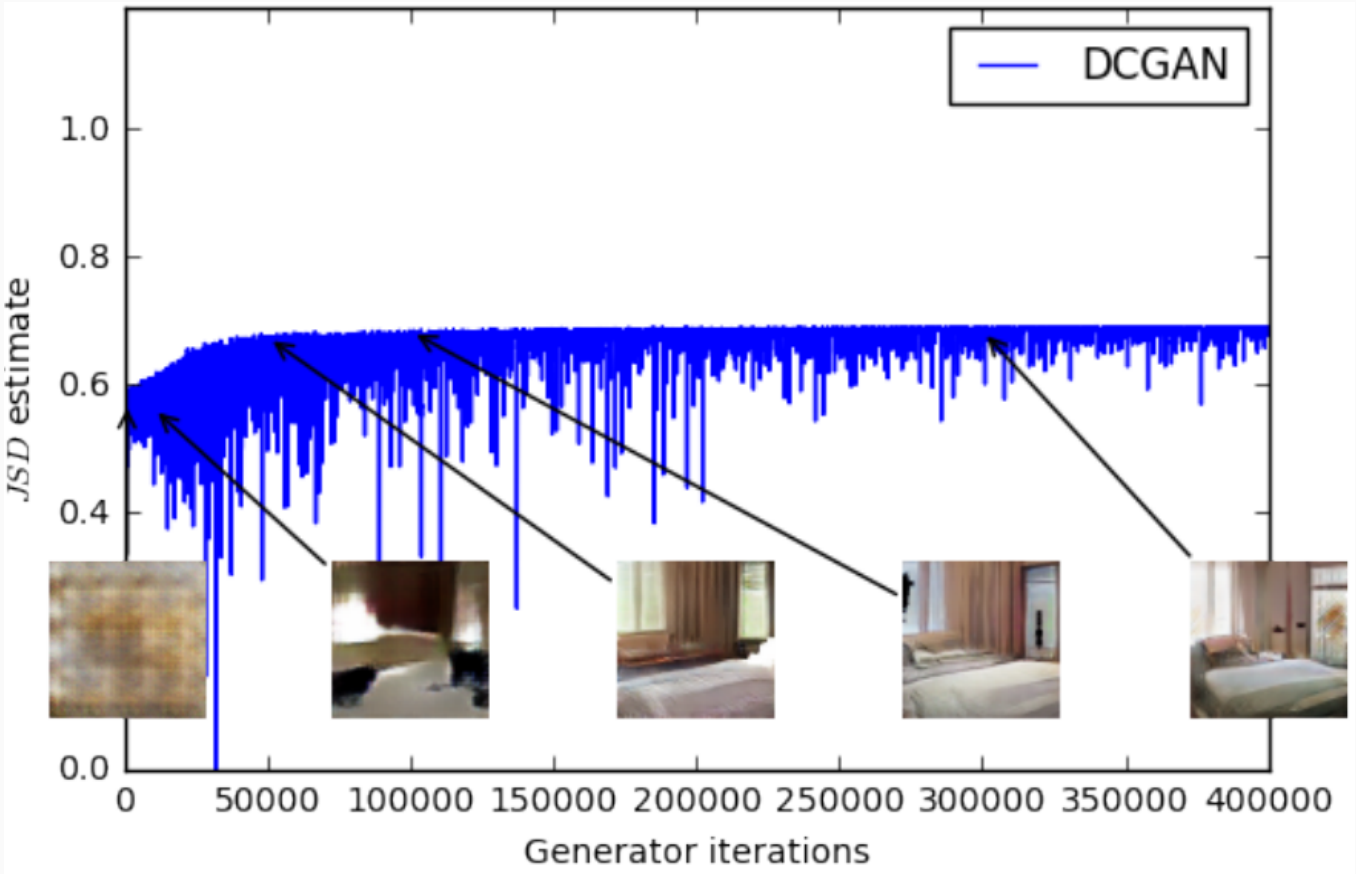
1. $||.||_L$ is the lipschitz norm
2. In practice: restrict D to some deep net architecture
3. Enforce Lipschitz constraint by clipping discriminator weights or penalty on gradient magnitude [Gulrajani et al., 2017a]

- Removes log-sigmoid transformation w.r.t. normal GAN



Experimental comparison GAN and WGAN

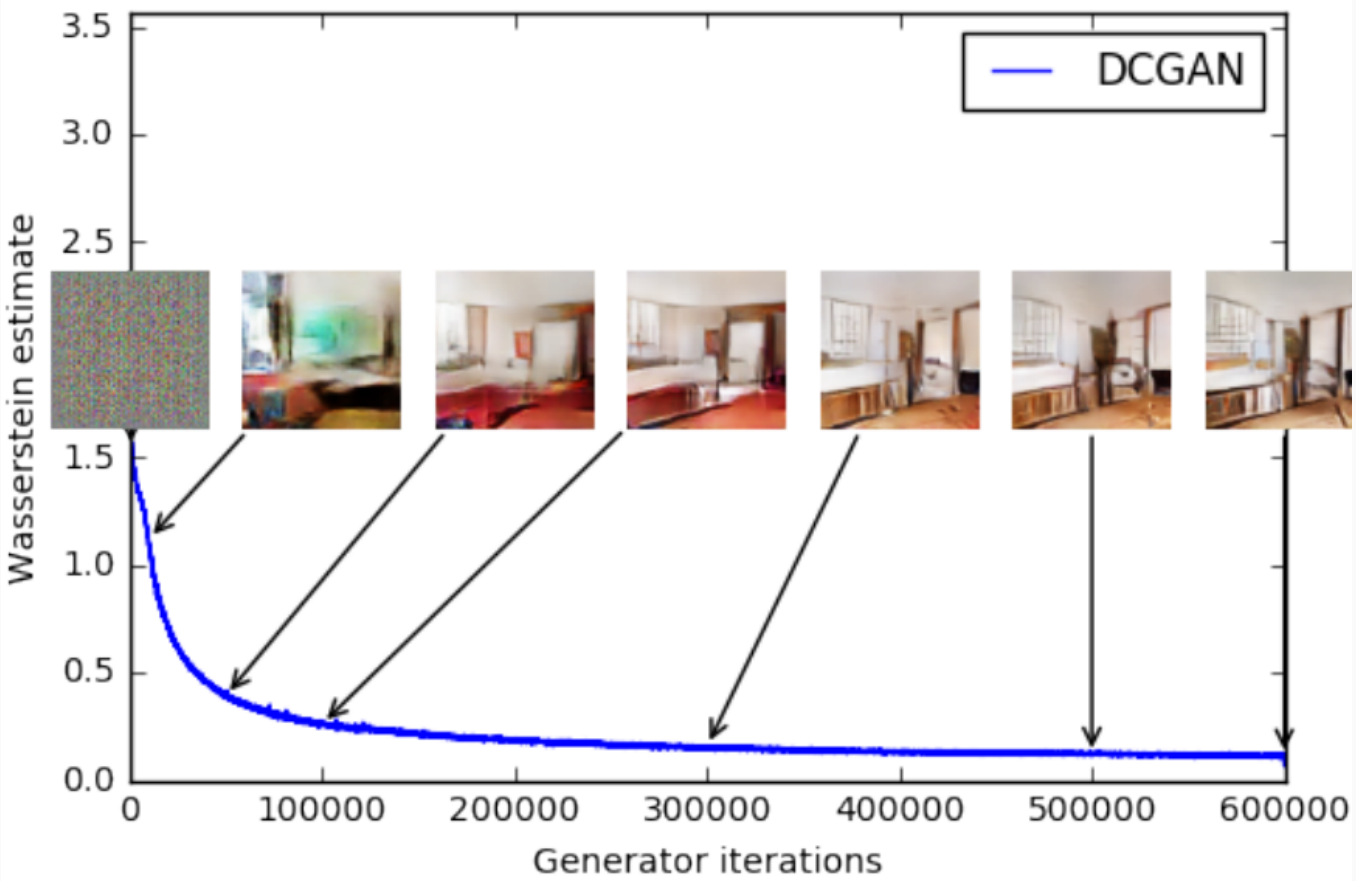
- WGAN loss may decrease in a more stable manner
- WGAN loss may give better correlation between loss and sample quality



GAN

Experimental comparison GAN and WGAN

- WGAN loss may decrease in a more stable manner
- WGAN loss may give better correlation between loss and sample quality



WGAN

In practice

Is this analysis relevant in practice?

In practice

Is this analysis relevant in practice?

- This analysis regards the **ideal losses** (D_{KL} VS. D_{WS})

In practice

Is this analysis relevant in practice?

- This analysis regards the **ideal losses** (D_{KL} VS. D_{WS})
- In practice, both are approximated by **similar discriminators**
  - $L_{WGAN} = \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))]$
  - $L_{GAN} = \frac{1}{k} \max_D \mathbb{E}_{p_{\text{data}}}[log(D(\mathbf{x}))] - \mathbb{E}_{p_z}[log(1 - D(G(\mathbf{z})))]$

In practice

Is this analysis relevant in practice?

- This analysis regards the **ideal losses** (D_{KL} VS. D_{WS})
- In practice, both are approximated by **similar discriminators**
  - $L_{WGAN} = \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))]$
  - $L_{GAN} = \frac{1}{k} \max_D \mathbb{E}_{p_{\text{data}}}[log(D(\mathbf{x}))] - \mathbb{E}_{p_z}[log(1 - D(G(\mathbf{z})))]$
- In practice, non-overlapping support does not break the discriminator

In practice

Is this analysis relevant in practice?

- This analysis regards the **ideal losses** (D_{KL} VS. D_{WS})
- In practice, both are approximated by **similar discriminators**
  - $L_{WGAN} = \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))]$
  - $L_{GAN} = \frac{1}{k} \max_D \mathbb{E}_{p_{\text{data}}}[log(D(\mathbf{x}))] - \mathbb{E}_{p_z}[log(1 - D(G(\mathbf{z})))]$
- In practice, non-overlapping support does not break the discriminator
- Constraining the Lipshitz constant is a good regularizer

In practice

Is this analysis relevant in practice?

- This analysis regards the **ideal losses** (D_{KL} VS. D_{WS})
- In practice, both are approximated by **similar discriminators**
  - $L_{WGAN} = \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))]$
  - $L_{GAN} = \frac{1}{k} \max_D \mathbb{E}_{p_{\text{data}}}[log(D(\mathbf{x}))] - \mathbb{E}_{p_z}[log(1 - D(G(\mathbf{z})))]$
- In practice, non-overlapping support does not break the discriminator
- Constraining the Lipshitz constant is a good regularizer
- Removing the log avoids vanishing gradients

Lipschitz continuity as a regularizer

- **Reminder:** k-Lispschitz means $|f(x) - f(y)| \leq |x - y|$
- **Reminder:** For linear functions, seek the largest eigenvalue

Lipschitz continuity as a regularizer

- Lispschitz continuity now widely used, but avoid clipping

Lipschitz continuity as a regularizer

- Lispschitz continuity now widely used, but avoid clipping
- Spectral Normalization [Miyato et. al, 2018]:
  - Approximate the spectral norm using the power iteration method
  - Divide each weight matrices by it's spectral norm
  - The spectral norm of the full network is bounded by the product of norms

Lipschitz continuity as a regularizer

- Lispschitz continuity now widely used, but avoid clipping
- Spectral Normalization [Miyato et. al, 2018]:
  - Approximate the spectral norm using the power iteration method
  - Divide each weight matrices by it's spectral norm
  - The spectral norm of the full network is bounded by the product of norms
- Gradient penalty [Gulrajani et. al, 2017]
  - Add a penalty to the loss:

$$G_{\text{pen}} = \lambda \mathbb{E}_x[||\nabla_x D(x)||_2 - 1)^2]$$

Wrap up

- A **lot** of other losses have been develloped
- The lipschitz regularization is a widely adopted regularization
- The log is usually avoided to improve gradients when Discriminator is good.

Part II

Adversarial Inference

Latent variable inference in GANs [Donahue et al., 2017]

- Vanilla GAN lacks a mechanism to infer $\mathbf{z}$ from $\mathbf{x}$

Latent variable inference in GANs [Donahue et al., 2017]

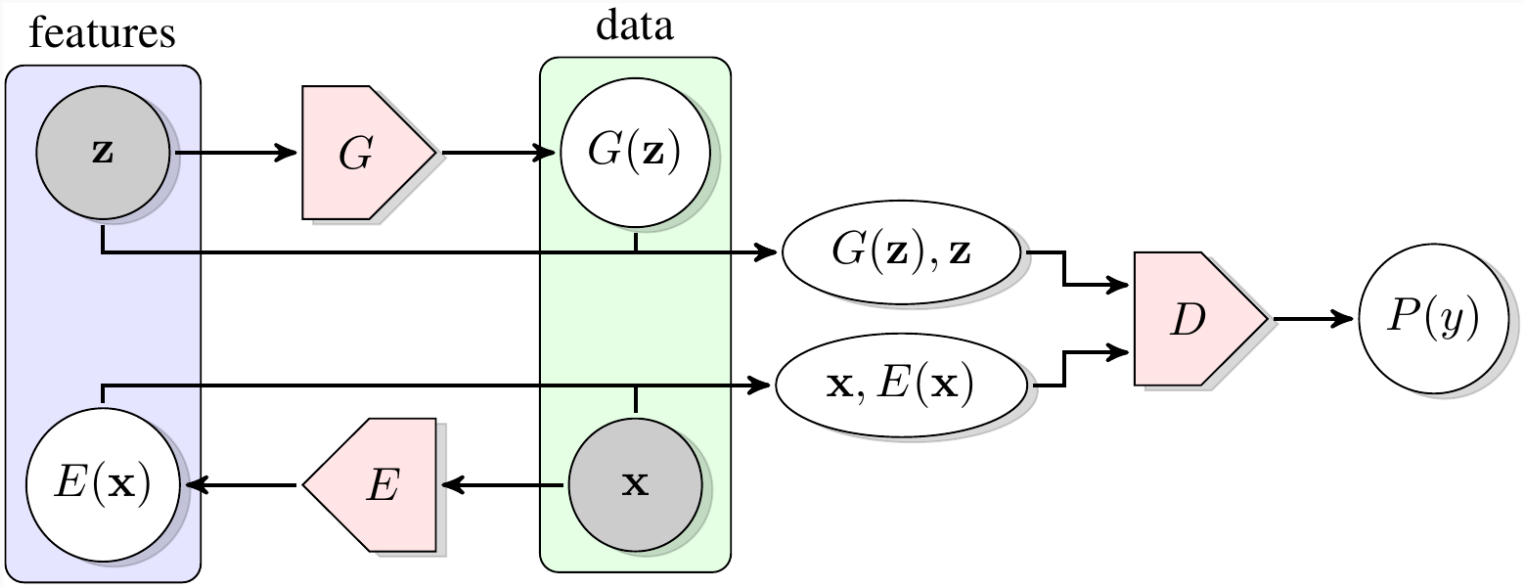
- Vanilla GAN lacks a mechanism to infer **z** from **x**
- Generator: maps latent variable **z** to data point **x**

Latent variable inference in GANs [Donahue et al., 2017]

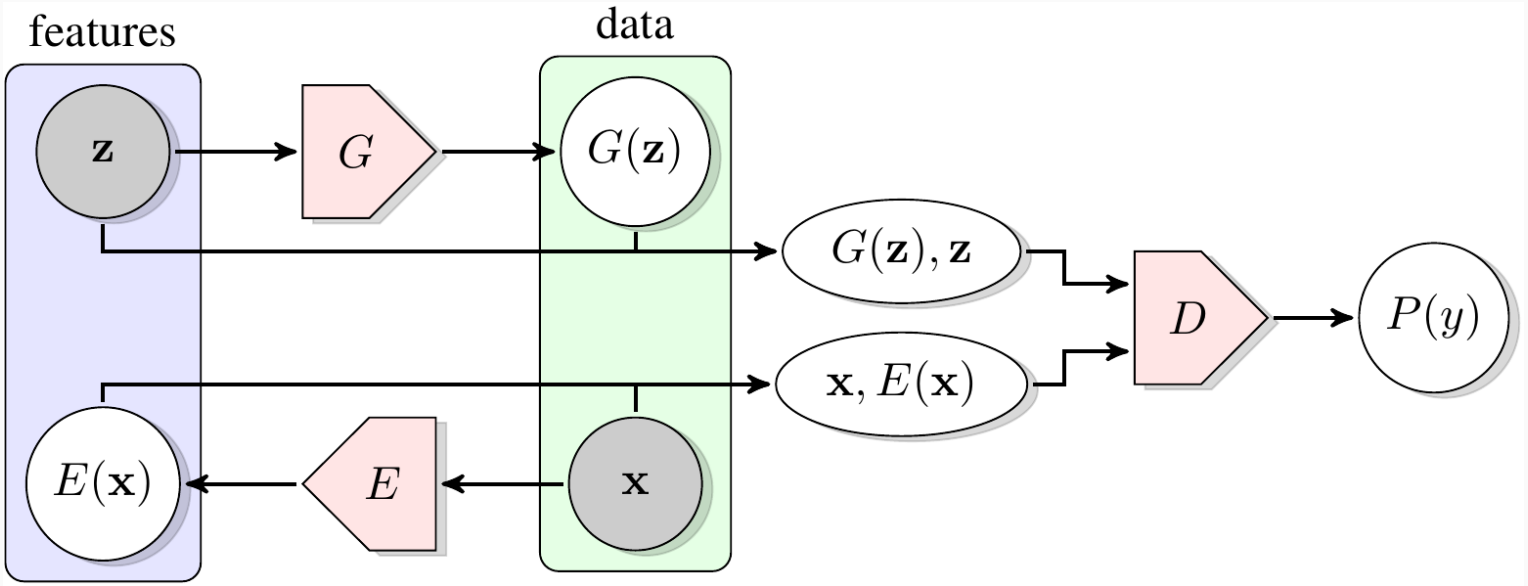
- Vanilla GAN lacks a mechanism to infer **z** from **x**
- Generator: maps latent variable **z** to data point **x**
- Encoder: infers latent representation **z** from data point **x**

Latent variable inference in GANs [Donahue et al., 2017]

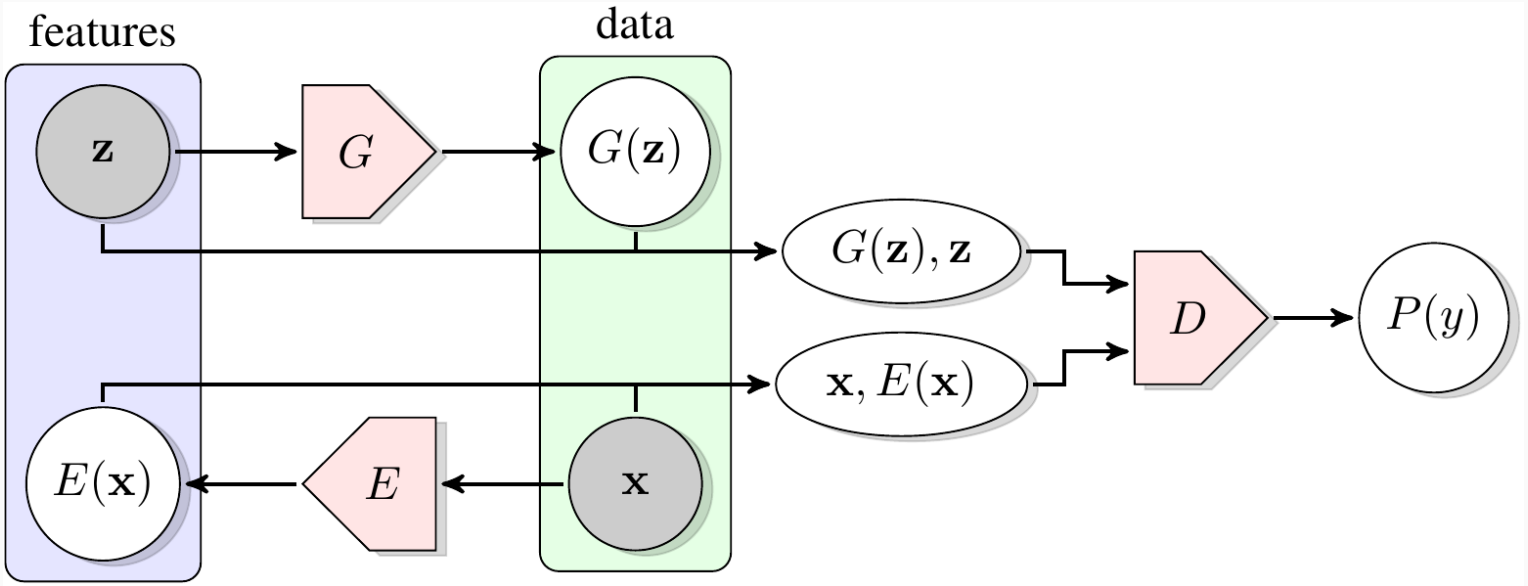
- Vanilla GAN lacks a mechanism to infer **z** from **x**
- Generator: maps latent variable **z** to data point **x**
- Encoder: infers latent representation **z** from data point **x**



Induced joint distributions over $(\mathbf{x}, \mathbf{z})$

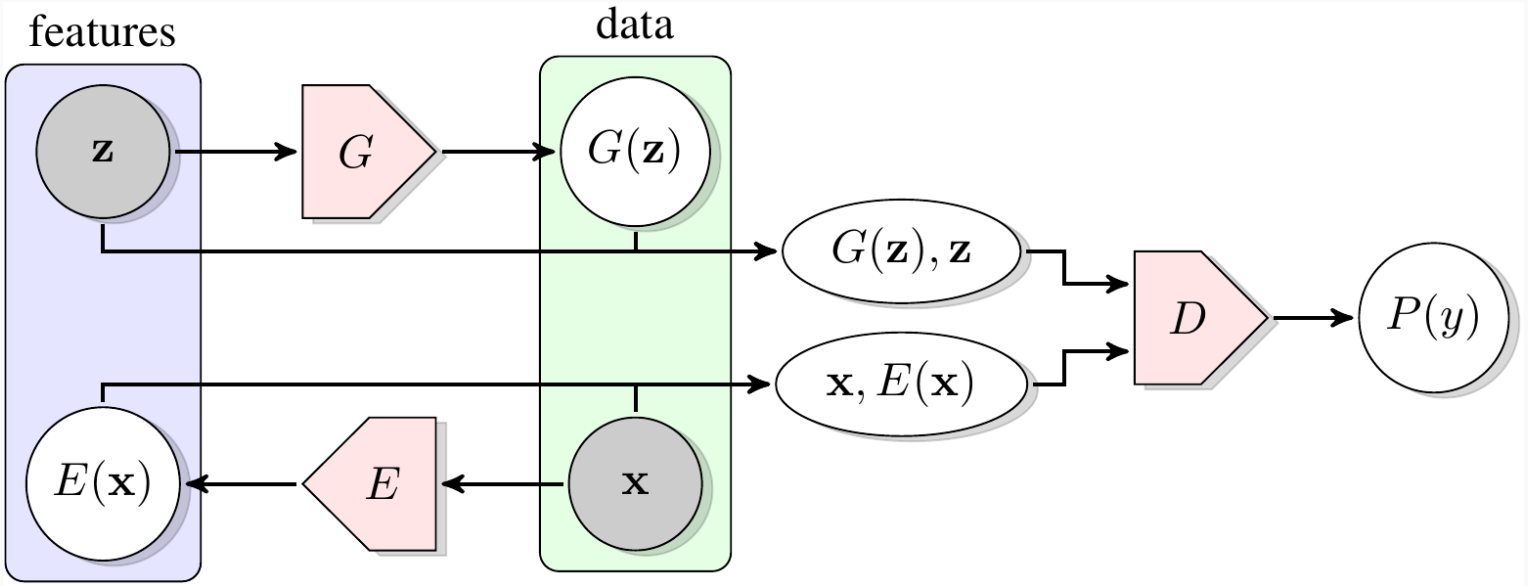


Induced joint distributions over $(\mathbf{x}, \mathbf{z})$



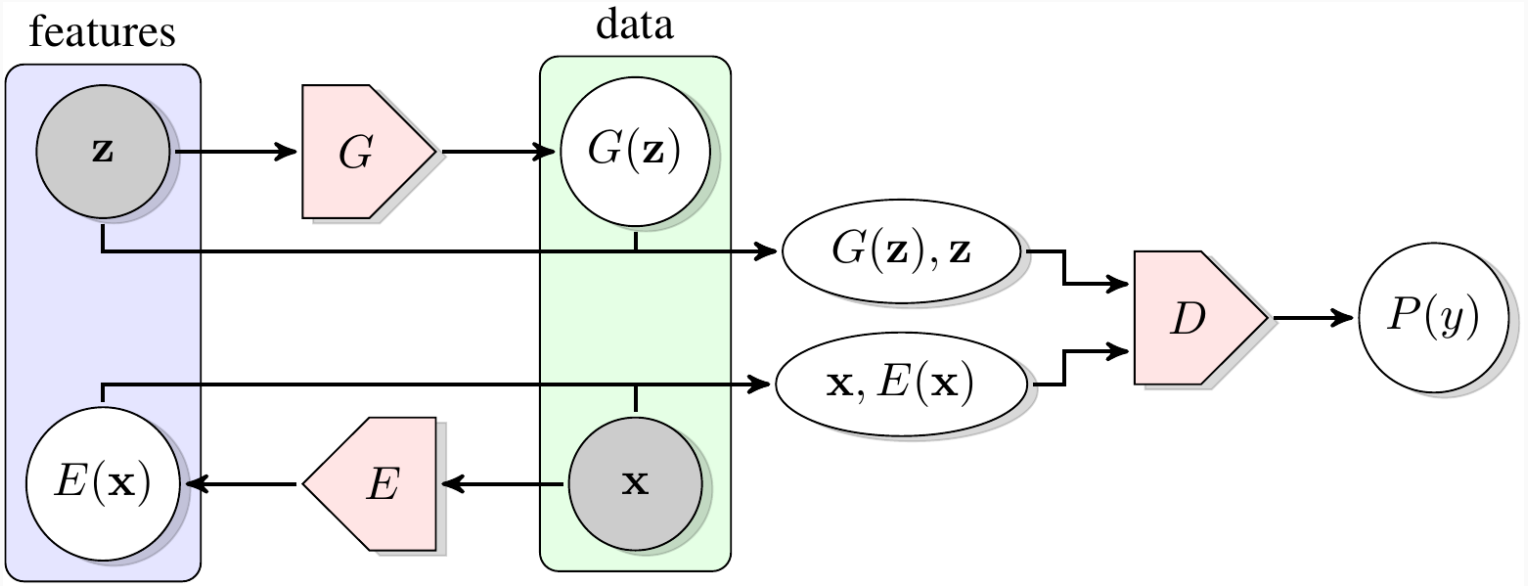
- Generator: $p_G(\mathbf{x}, \mathbf{z}) = p_{\mathbf{z}}(\mathbf{z})\ \delta\left(\mathbf{x} - G(\mathbf{z})\right)$

Induced joint distributions over $(\mathbf{x}, \mathbf{z})$



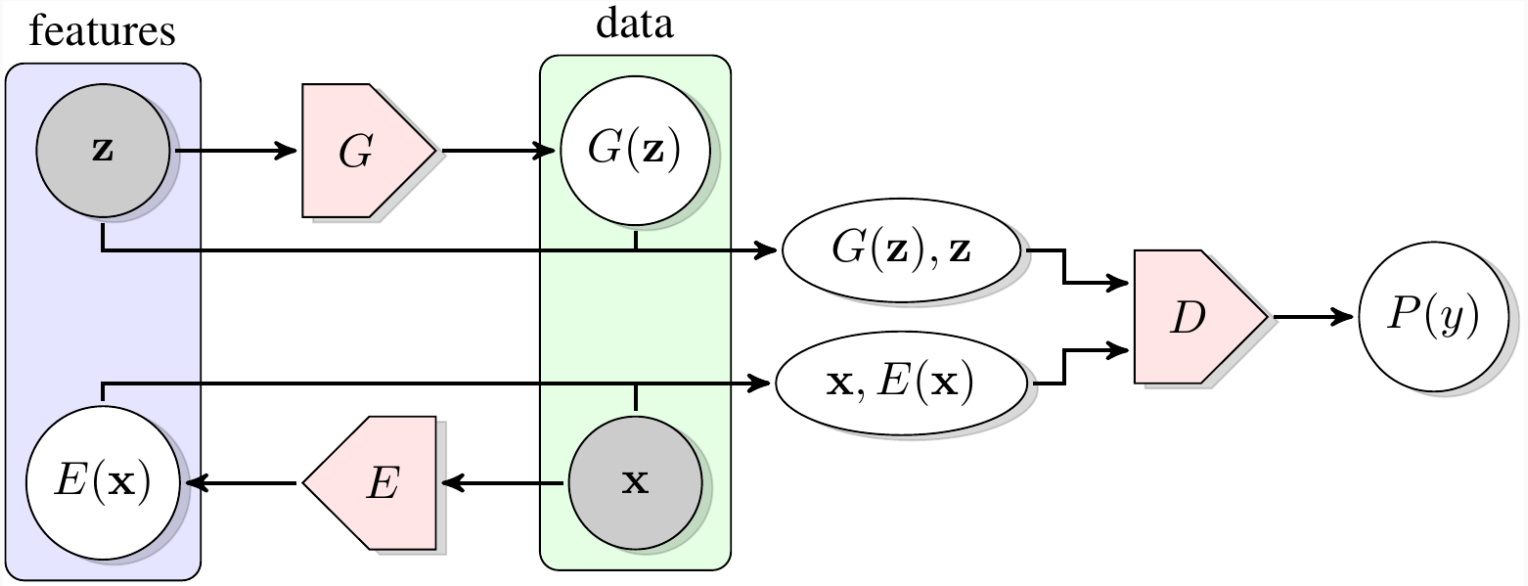
- Generator: $p_G(\mathbf{x}, \mathbf{z}) = p_{\mathbf{z}}(\mathbf{z})\ \delta\left(\mathbf{x} - G(\mathbf{z})\right)$
- Encoder: $p_E(\mathbf{x}, \mathbf{z}) = p_{\text{data}}(\mathbf{x})\ \delta\left(\mathbf{z} - E(\mathbf{x})\right)$

Induced joint distributions over $(\mathbf{x}, \mathbf{z})$



- Generator: $p_G(\mathbf{x}, \mathbf{z}) = p_{\mathbf{z}}(\mathbf{z})\ \delta\left(\mathbf{x} - G(\mathbf{z})\right)$
- Encoder: $p_E(\mathbf{x}, \mathbf{z}) = p_{\text{data}}(\mathbf{x})\ \delta\left(\mathbf{z} - E(\mathbf{x})\right)$
- Discriminator: pair $(\mathbf{x}, \mathbf{z})$ completed by generator or encoder?

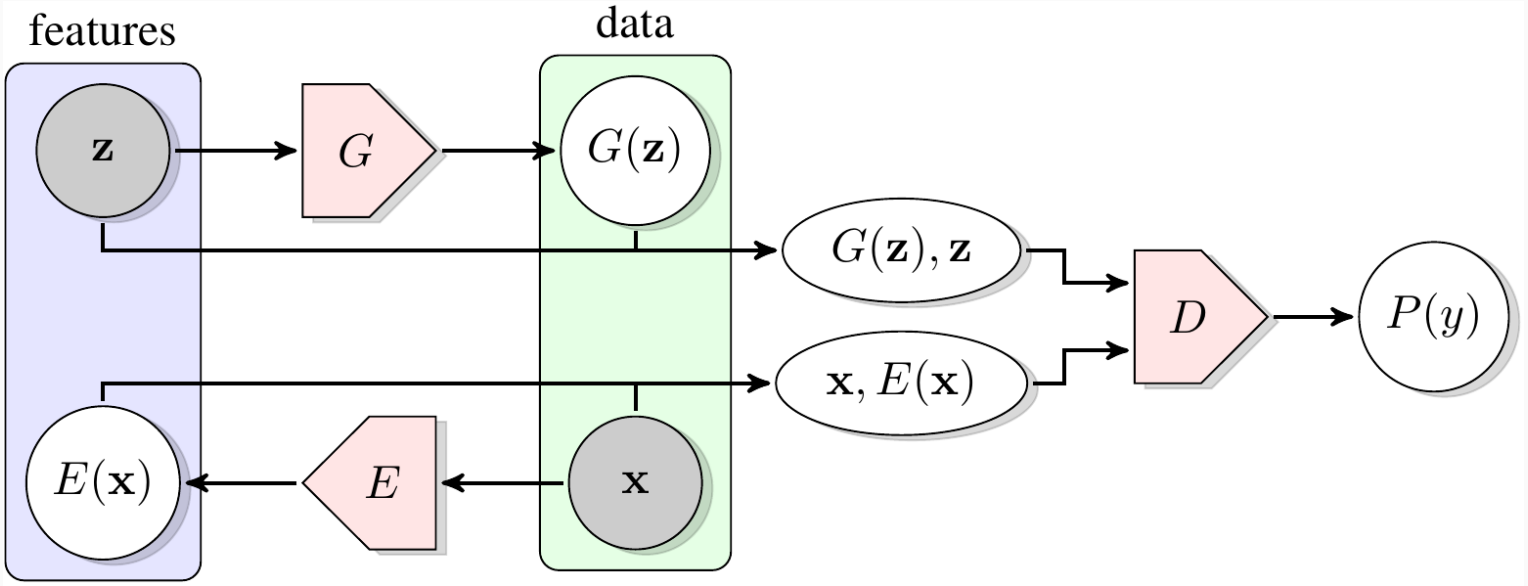
Bidirectional GANs [Donahue et al., 2017]



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

$$\min_{G,E} \max_{D} V(D, E, G)$$

Bidirectional GANs [Donahue et al., 2017]



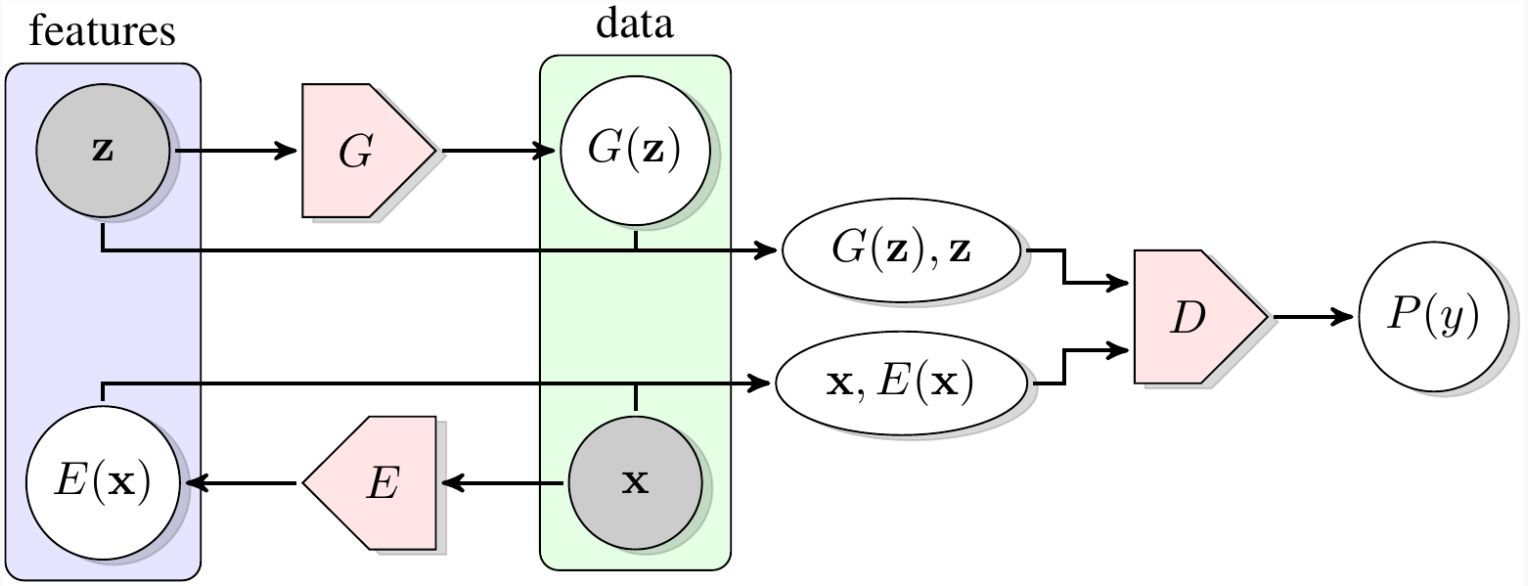
$$V(D, E, G) = \mathbb{E}_{p_{\text{data}}}[\ln D(\mathbf{x}, E(\mathbf{x}))] + \mathbb{E}_{p(\mathbf{z})}[\ln(1 - D(G(\mathbf{z}), \mathbf{z}))]$$

$$\min_{G,E} \max_{D} V(D, E, G)$$

- For optimal discriminator objective equals JS divergence

$$\max_{D} V(D, E, G) = 2D_{JS}\left(p_E(\mathbf{x}, \mathbf{z}) || p_G(\mathbf{x}, \mathbf{z})\right) - \ln 4$$

Bidirectional GANs [Donahue et al., 2017]



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

$$\min_{G,E} \max_{D} V(D, E, G)$$

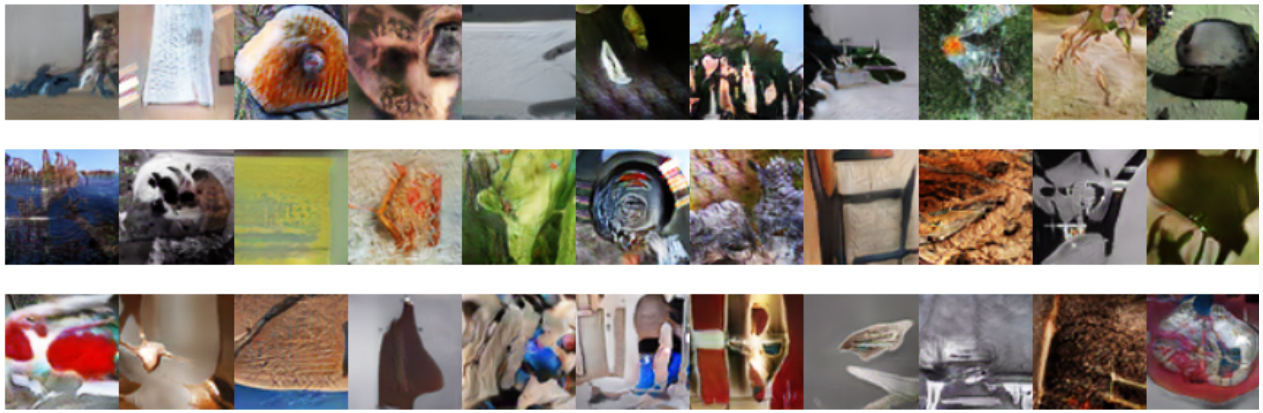
- For optimal discriminator objective equals JS divergence

$$\max_{D} V(D, E, G) = 2D_{JS}\left(p_E(\mathbf{x}, \mathbf{z}) || p_G(\mathbf{x}, \mathbf{z})\right) - \ln 4$$

- At optimum G and E are each others inverse

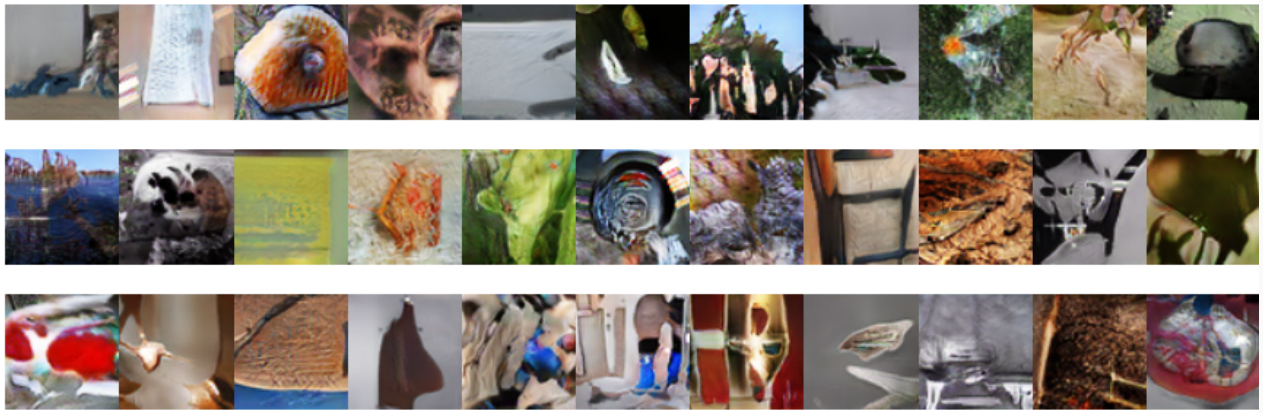
BiGAN samples, ImageNet 64×64

$G(\mathbf{z})$



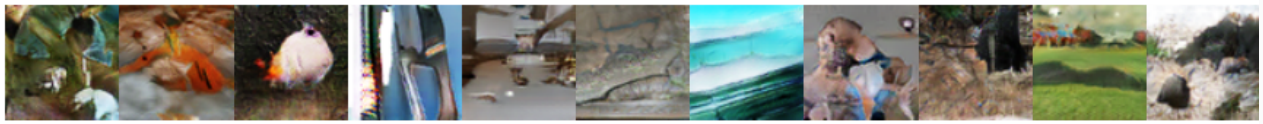
BiGAN samples, ImageNet 64×64

$G(\mathbf{z})$

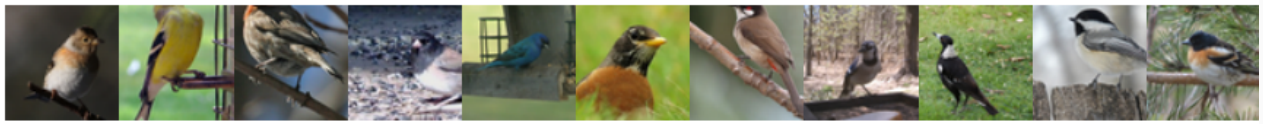


$\mathbf{x}$

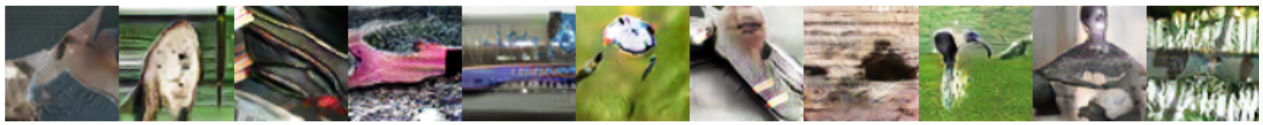
$G(E(\mathbf{x}))$



$\mathbf{x}$



$G(E(\mathbf{x}))$

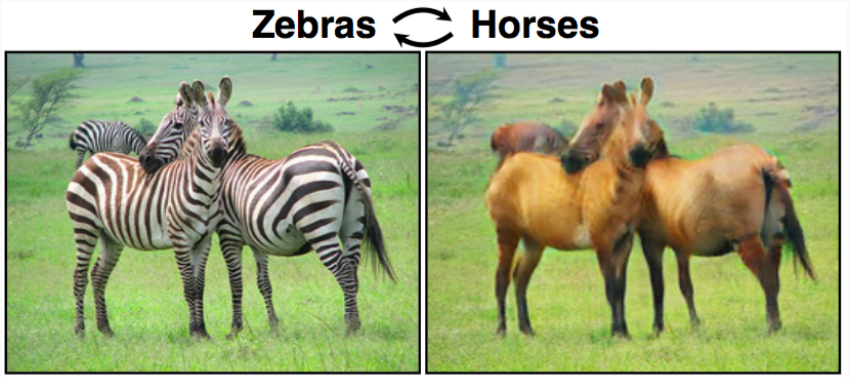


BiGAN: feature transfer to PASCAL VOC'07

	trained layers	Classification (% mAP) fc8	fc6-8	all	*FRCN* Detection (% mAP) all	*FCN* Segmentation (% mIU) all
sup.	ImageNet (Krizhevsky et al., 2012)	**77.0**	**78.8**	**78.3**	**56.8**	**48.0**
self-sup.	Agrawal et al. (2015)	31.2	31.0	54.2	43.9	-
	Pathak et al. (2016)	30.5	34.6	56.5	44.5	**30.0**
	Wang & Gupta (2015)	28.4	**55.6**	63.1	47.4	-
	Doersch et al. (2015)	**44.7**	55.1	**65.3**	**51.1**	-
unsup.	k-means (Krähenbühl et al., 2016)	32.0	39.2	56.6	45.6	32.6
	Discriminator (D)	30.7	40.5	56.4	-	-
	Latent Regressor (LR)	36.9	47.9	57.1	-	-
	Joint LR	37.1	47.9	56.5	-	-
	Autoencoder (ℓ_2)	24.8	16.0	53.8	41.9	-
	BiGAN (ours)	37.5	48.7	58.9	46.2	34.9
	BiGAN, 112×112 E (ours)	**40.7**	**52.3**	**60.1**	**46.9**	**35.2**

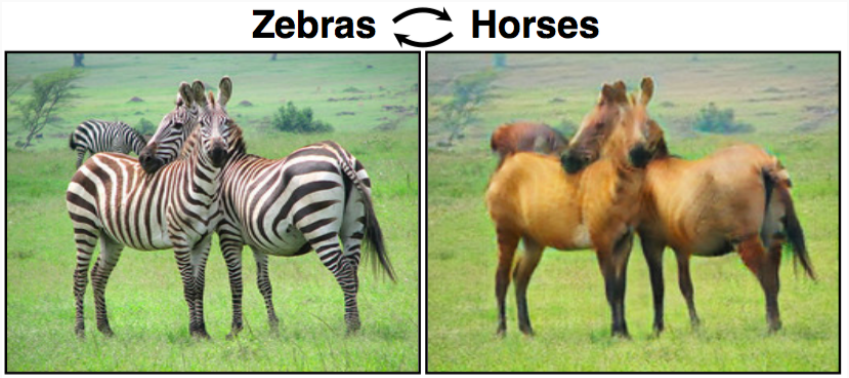
- Encoder network used to pre-train/initialize recognition net
- Similar performance of BiGAN and self-supervised methods
  - Self-sup: CNN trained using image structure as supervision
  - Discriminator: uses GAN discriminator layers as features
  - Latent regressor: trains network to invert GAN (jointly)

Unpaired image-to-image translation [Zhu et al., 2017]



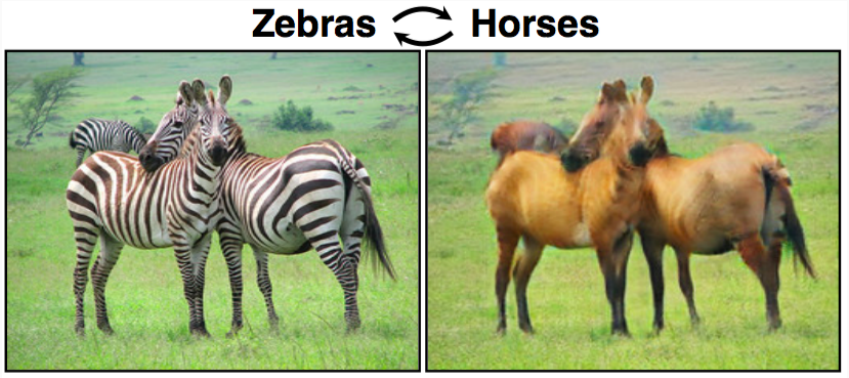
- Learn 2-way mapping between different image domains

Unpaired image-to-image translation [Zhu et al., 2017]



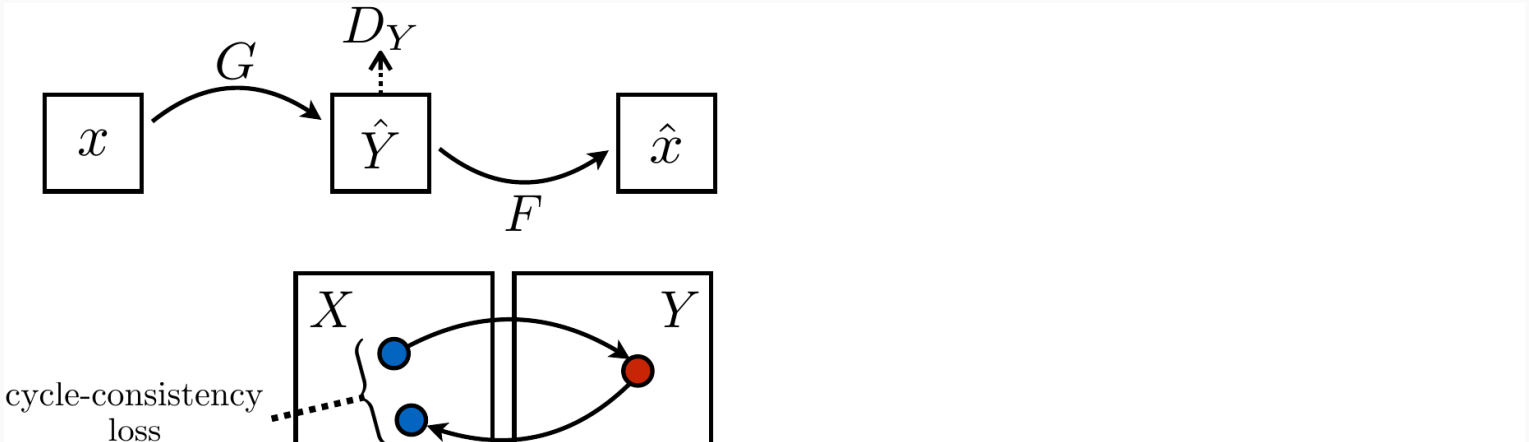
- Learn 2-way mapping between different image domains
- Without using supervised aligned training samples

Unpaired image-to-image translation [Zhu et al., 2017]

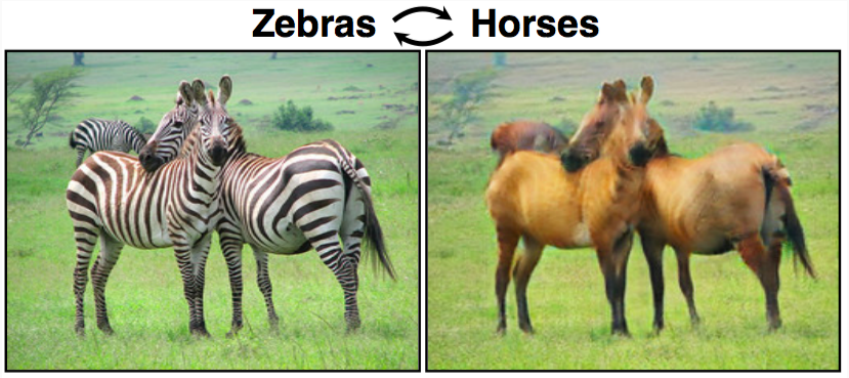


- Learn 2-way mapping between different image domains
- Without using supervised aligned training samples

1. Discriminator ensures realistic samples in each domain

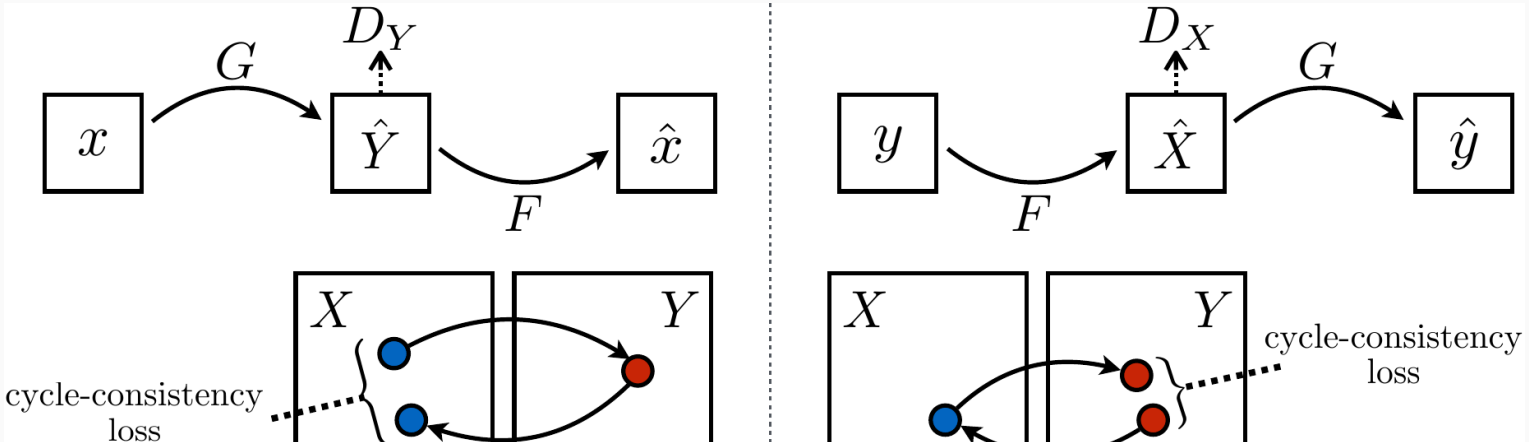


Unpaired image-to-image translation [Zhu et al., 2017]

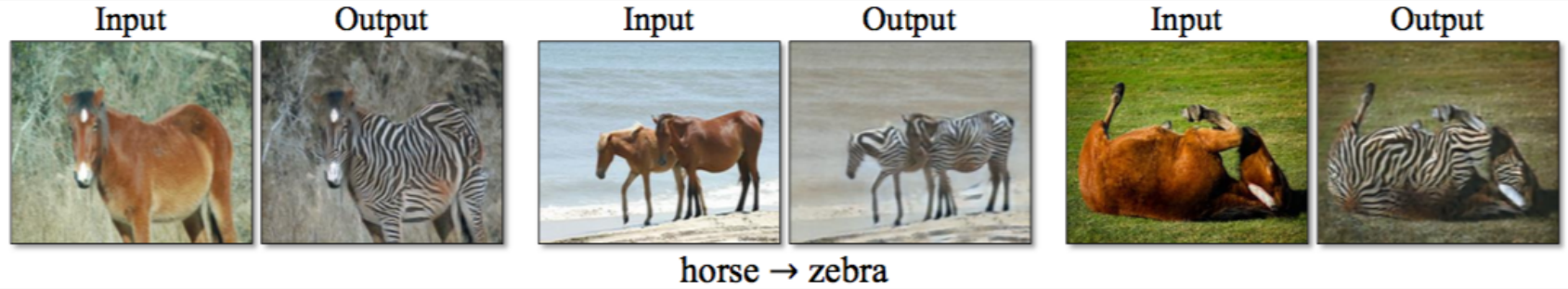


- Learn 2-way mapping between different image domains
- Without using supervised aligned training samples

1. Discriminator ensures realistic samples in each domain
2. Cycle-consistency loss ensures alignment

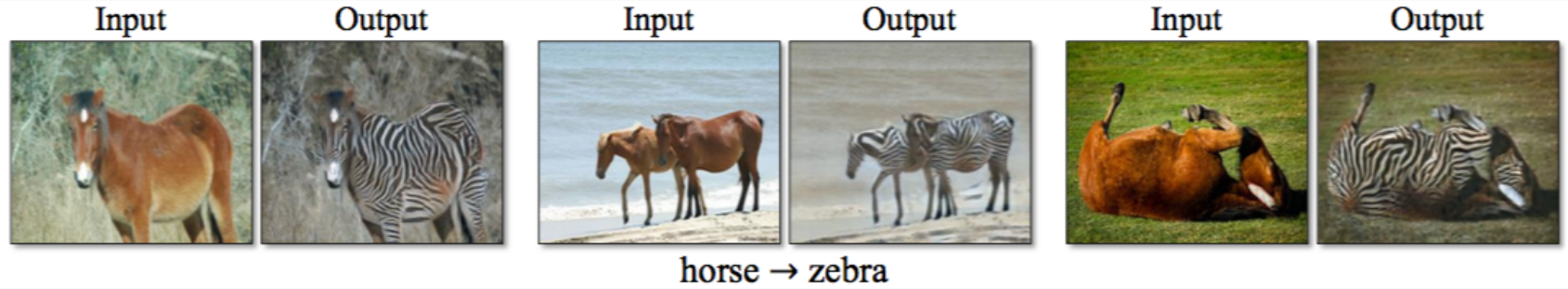


Some successful examples



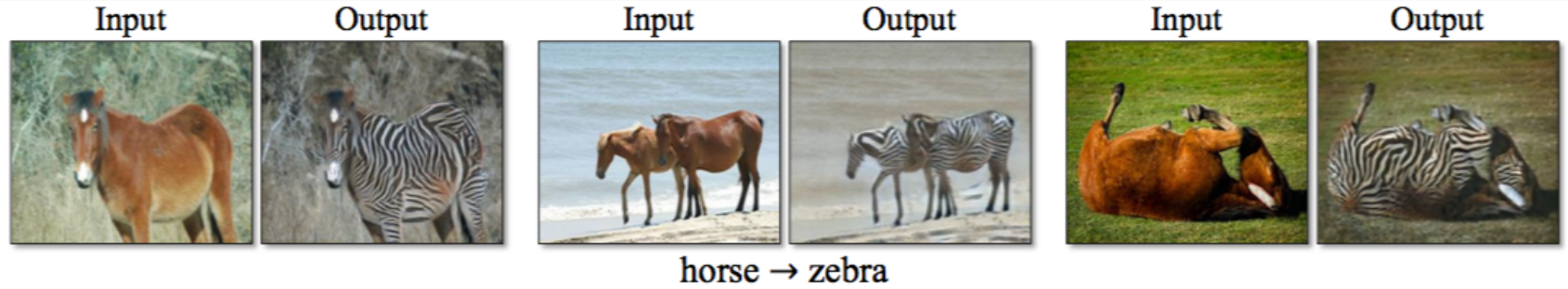
Some successful examples

- Without using any supervised/aligned examples!



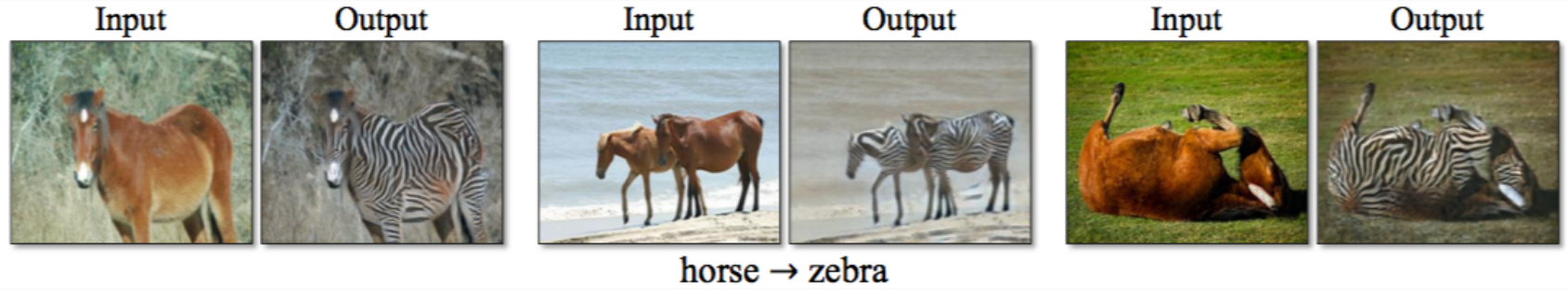
Some successful examples

- Without using any supervised/aligned examples!



Some successful examples

- Without using any supervised/aligned examples!



And a failure case

Wrap-up on GANs

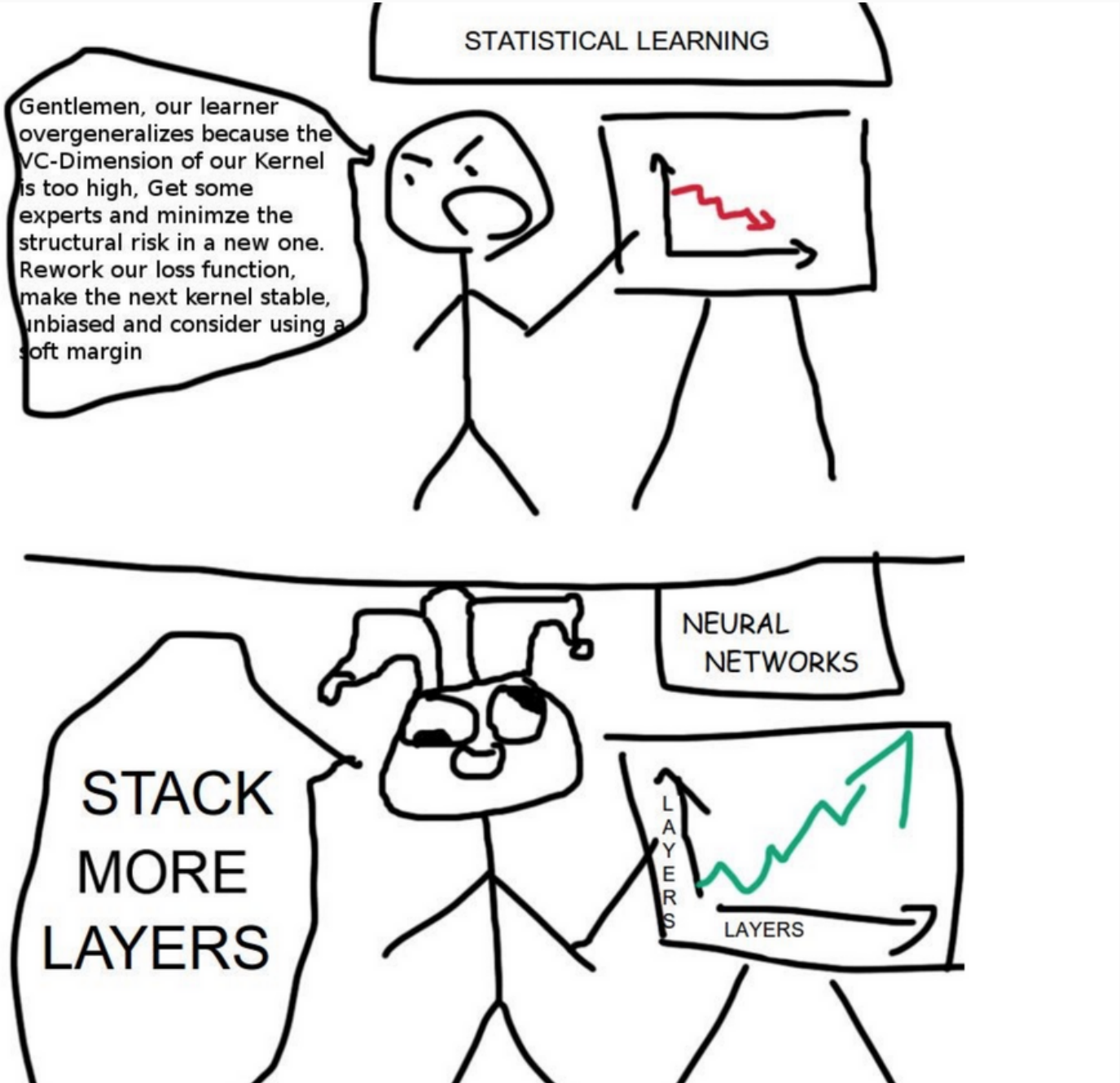
Summary of what we discussed

- Improved losses using lipschitz constraints, inspired by earth-mover distance
- Adversarially trained inference networks.
- Style transfer

Pause

We are done talking about GANs.

It do be like that



Part III

Improving Variational Auto-encoders

Improving variational autoencoders

Improving variational autoencoders

- **Reminder:** VAEs optimize the ELBO, with a KL divergence to bound the data log-likelihood

$$F(\mathbf{x}, \theta, \phi) = \ln p(\mathbf{x}) - D\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right) \tag{8}$$

Improving variational autoencoders

- **Reminder:** VAEs optimize the ELBO, with a KL divergence to bound the data log-likelihood

$$F(\mathbf{x}, \theta, \phi) = \ln p(\mathbf{x}) - D\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right) \tag{8}$$

  - Generally true posterior is not Gaussian: loose bound

Improving variational autoencoders

- **Reminder:** VAEs optimize the ELBO, with a KL divergence to bound the data log-likelihood

$$F(\mathbf{x}, \theta, \phi) = \ln p(\mathbf{x}) - D\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right) \tag{8}$$

  - Generally true posterior is not Gaussian: loose bound
  - Encourages true posterior to match variational factored Gaussian produced by recognition net

Improving variational autoencoders

- **Reminder:** VAEs optimize the ELBO, with a KL divergence to bound the data log-likelihood

$$F(\mathbf{x}, \theta, \phi) = \ln p(\mathbf{x}) - D\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right) \tag{8}$$

  - Generally true posterior is not Gaussian: loose bound
  - Encourages true posterior to match variational factored Gaussian produced by recognition net

- Making progress

Improving variational autoencoders

- **Reminder:** VAEs optimize the ELBO, with a KL divergence to bound the data log-likelihood

$$F(\mathbf{x}, \theta, \phi) = \ln p(\mathbf{x}) - D\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right) \tag{8}$$

  - Generally true posterior is not Gaussian: loose bound
  - Encourages true posterior to match variational factored Gaussian produced by recognition net

- Making progress
  1. More accurate bound for given posterior

Improving variational autoencoders

- **Reminder:** VAEs optimize the ELBO, with a KL divergence to bound the data log-likelihood

$$F(\mathbf{x}, \theta, \phi) = \ln p(\mathbf{x}) - D\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right) \tag{8}$$

  - Generally true posterior is not Gaussian: loose bound
  - Encourages true posterior to match variational factored Gaussian produced by recognition net

- Making progress
  1. More accurate bound for given posterior
  2. Enlarge the family of variational posteriors
     - Hierarchical latent variables
     - Improved flexibility with flows

Importance weighted autoencoders [Burda et al., 2016]

- Construct tighter lower bound using importance sampling

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$F_k(\mathbf{x}, \theta, \phi) = \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right]$$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
F_k(\mathbf{x}, \theta, \phi) &= \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&\leq \ln \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right]
\end{aligned}
$$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
F_k(\mathbf{x}, \theta, \phi) &= \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&\leq \ln \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&= \ln \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [w(\mathbf{x}, \mathbf{z})]
\end{aligned}
$$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
F_k(\mathbf{x}, \theta, \phi) &= \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&\leq \ln \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&= \ln \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [w(\mathbf{x}, \mathbf{z})] \\
&= \ln p(\mathbf{x})
\end{aligned}
$$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
F_k(\mathbf{x}, \theta, \phi) &= \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&\leq \ln \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&= \ln \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [w(\mathbf{x}, \mathbf{z})] \\
&= \ln p(\mathbf{x})
\end{aligned}
$$

1. VAE lower bound recovered for $k = 1$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
F_k(\mathbf{x}, \theta, \phi) &= \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&\leq \ln \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&= \ln \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [w(\mathbf{x}, \mathbf{z})] \\
&= \ln p(\mathbf{x})
\end{aligned}
$$

1. VAE lower bound recovered for $k = 1$
2. More samples tighten the bound: $F_k \leq F_{k+1} \leq \ln p(\mathbf{x})$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
F_k(\mathbf{x}, \theta, \phi) &= \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&\leq \ln \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&= \ln \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [w(\mathbf{x}, \mathbf{z})] \\
&= \ln p(\mathbf{x})
\end{aligned}
$$

1. VAE lower bound recovered for $k = 1$
2. More samples tighten the bound: $F_k \leq F_{k+1} \leq \ln p(\mathbf{x})$
3. If the weights are bounded, then $F_k \to \ln p(\mathbf{x})$ as $k \to \infty$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
F_k(\mathbf{x}, \theta, \phi) &= \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&\leq \ln \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&= \ln \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [w(\mathbf{x}, \mathbf{z})] \\
&= \ln p(\mathbf{x})
\end{aligned}
$$

1. VAE lower bound recovered for $k = 1$
2. More samples tighten the bound: $F_k \leq F_{k+1} \leq \ln p(\mathbf{x})$
3. If the weights are bounded, then $F_k \to \ln p(\mathbf{x})$ as $k \to \infty$

- Use as objective to train models for $k \approx 10$

Importance weighted autoencoders [Burda et al., 2016]

- **Construct tighter lower bound using importance sampling**
- Define importance weights $w(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
F_k(\mathbf{x}, \theta, \phi) &= \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&\leq \ln \mathbb{E}_{\mathbf{z}_1, \ldots, \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{z}_i) \right] \\
&= \ln \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [w(\mathbf{x}, \mathbf{z})] \\
&= \ln p(\mathbf{x})
\end{aligned}
$$

1. VAE lower bound recovered for $k = 1$
2. More samples tighten the bound: $F_k \leq F_{k+1} \leq \ln p(\mathbf{x})$
3. If the weights are bounded, then $F_k \to \ln p(\mathbf{x})$ as $k \to \infty$

- Use as objective to train models for $k \approx 10$
- Use as likelihood estimator for (IW-)VAE with $k \approx 10^3$

Training procedure importance weighted autoencoders

- Gradients of importance weighted lower bound

$$\nabla F_k(\mathbf{x}) = \mathbb{E}_{\mathbf{z}_{1:k} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\sum_{i=1}^{k} \widetilde{w}_i \nabla \big(\ln p(\mathbf{x}, \mathbf{z}_i) - \ln q_\phi(\mathbf{z}_i|\mathbf{x}) \big) \right]$$

Training procedure importance weighted autoencoders

- Gradients of importance weighted lower bound

$$\nabla F_k(\mathbf{x}) = \mathbb{E}_{\mathbf{z}_{1:k} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\sum_{i=1}^{k} \widetilde{w}_i \nabla \big(\ln p(\mathbf{x}, \mathbf{z}_i) - \ln q_\phi(\mathbf{z}_i|\mathbf{x}) \big) \right]$$

- Similar to VAE, but samples weighted w.r.t. true posterior
  - Normalized importance weights $\widetilde{w}_i = w(\mathbf{x}, \mathbf{z}_i) / \sum_{j=1}^{k} w(\mathbf{x}, \mathbf{z}_j)$

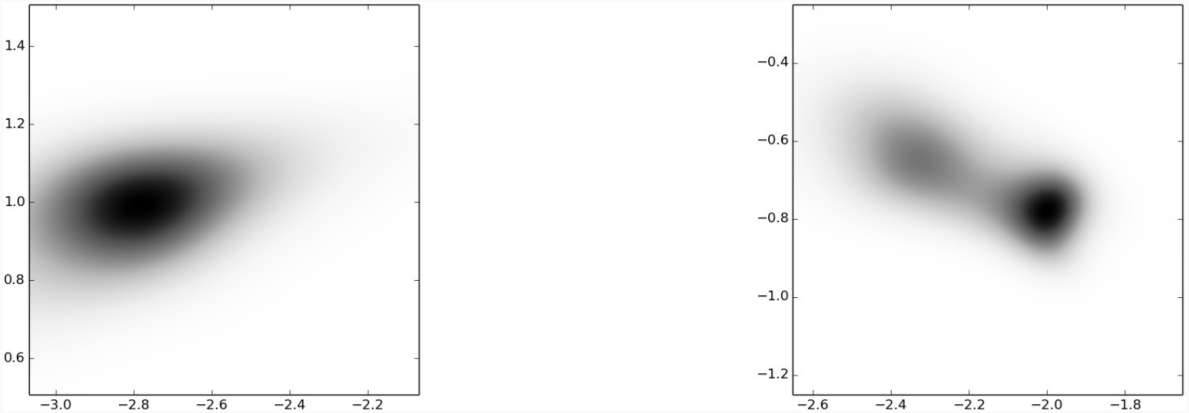
Training procedure importance weighted autoencoders

- Gradients of importance weighted lower bound

$$\nabla F_k(\mathbf{x}) = \mathbb{E}_{\mathbf{z}_{1:k} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\sum_{i=1}^{k} \widetilde{w}_i \nabla \big(\ln p(\mathbf{x}, \mathbf{z}_i) - \ln q_\phi(\mathbf{z}_i|\mathbf{x}) \big) \right]$$

- Similar to VAE, but samples weighted w.r.t. true posterior
  - Normalized importance weights $\widetilde{w}_i = w(\mathbf{x}, \mathbf{z}_i) / \sum_{j=1}^{k} w(\mathbf{x}, \mathbf{z}_j)$

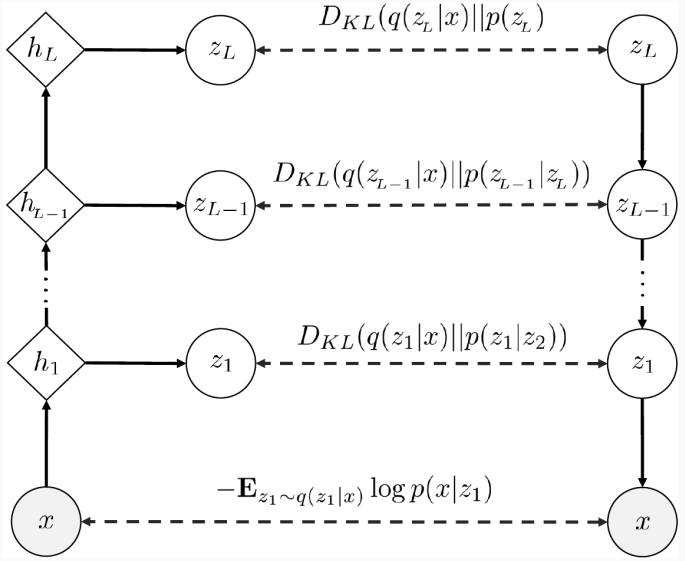
- Allows for more accurate models with complex posteriors



True posterior $p(\mathbf{z}|\mathbf{x})$ VAE (left) and IW-VAE (right)

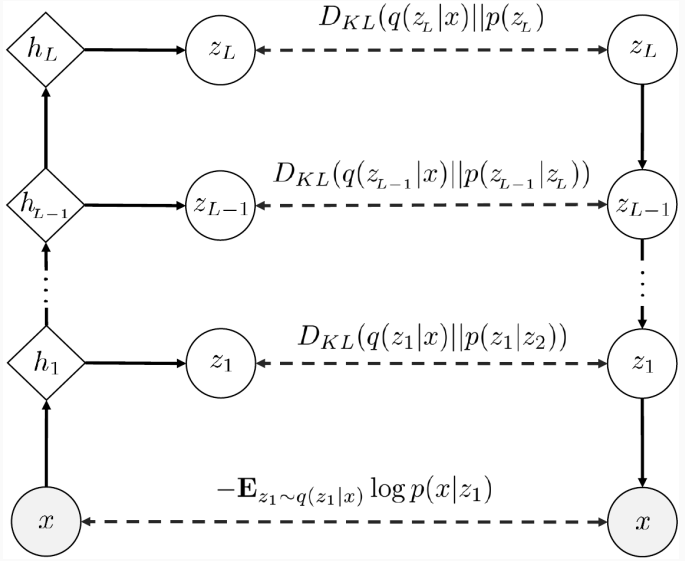
Top-down hierarchical sampling

- **Multiple levels of latent variables at increasing resolutions**



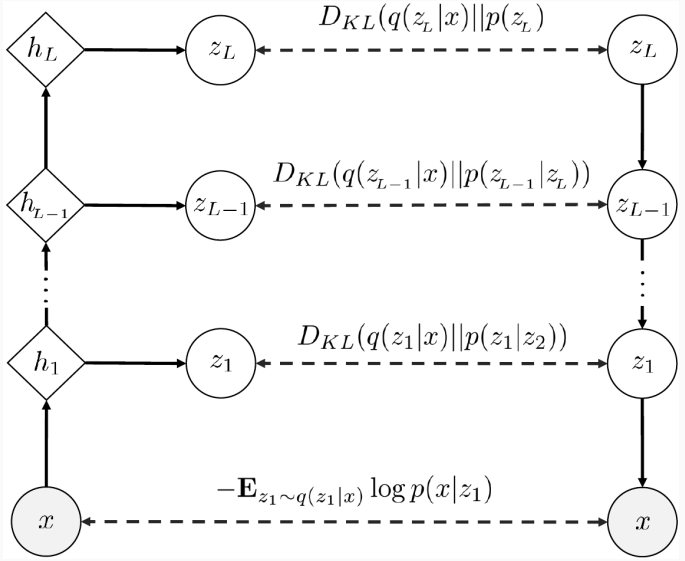
Top-down hierarchical sampling

- **Multiple levels of latent variables at increasing resolutions**
- Autoregressive distribution over latent variables in 2D grid ($p(z_1)p(z_2|z_1)$)



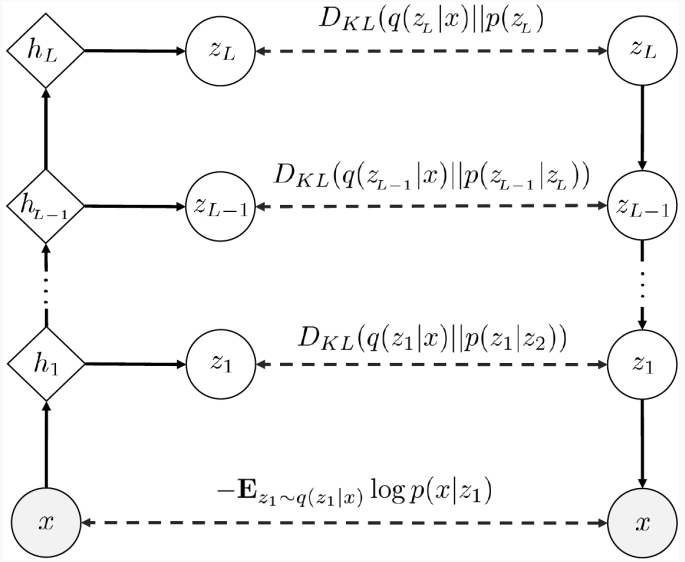
Top-down hierarchical sampling

- **Multiple levels of latent variables at increasing resolutions**
- Autoregressive distribution over latent variables in 2D grid ($p(z_1)p(z_2|z_1)$)
- Latent variables must be sampled in the same order when encoding or when sampling



Top-down hierarchical sampling

- **Multiple levels of latent variables at increasing resolutions**
- Autoregressive distribution over latent variables in 2D grid ($p(z_1)p(z_2|z_1)$)
- Latent variables must be sampled in the same order when encoding or when sampling
- Extended VAE log-likelihood bound

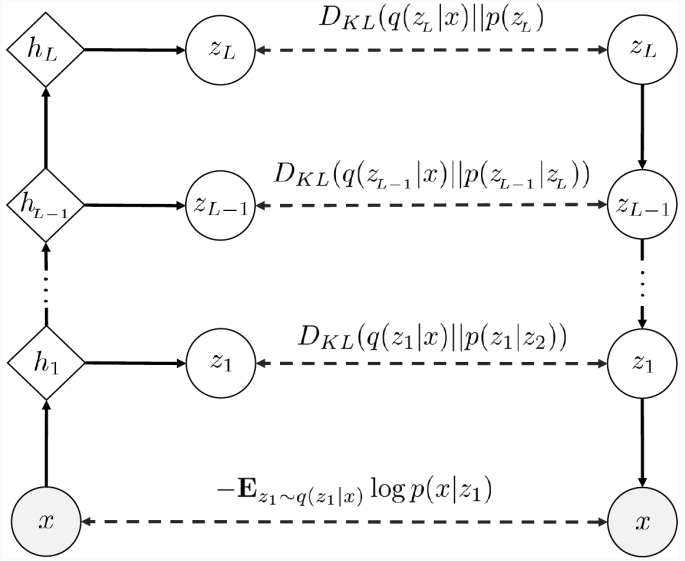


$$F = \ln p(\mathbf{x}) - D_{KL}(q(\mathbf{z}_{1:L}|\mathbf{x})||p(\mathbf{z}_{1:L}|\mathbf{x})$$

$$= \underbrace{\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}[\ln p(\mathbf{x}|\mathbf{z}_1)]}_{\text{Reconstruction}} - \underbrace{\sum_{i=1}^{L} \mathbb{E}_{q(\mathbf{z}_{i+1})}[D_{KL}(q(\mathbf{z}_i|\mathbf{x})||p(\mathbf{z}_i|\mathbf{z}_{i+1})])}_{\text{Regularization}}$$

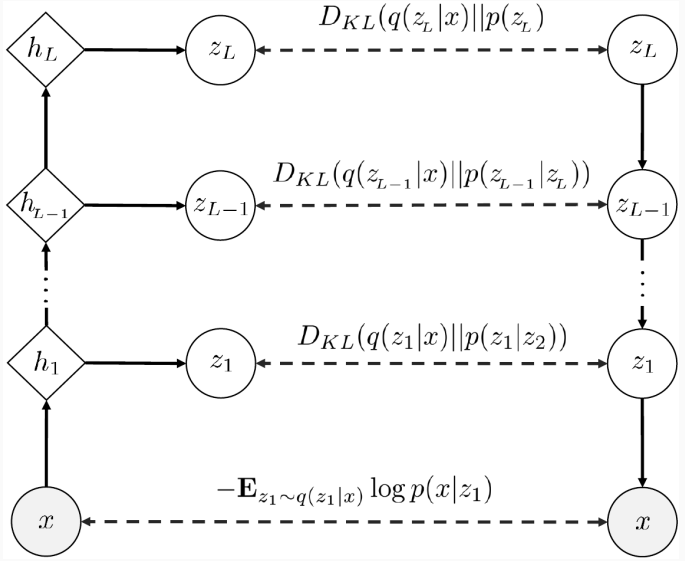
Top-down hierarchical sampling

- **Multiple levels of latent variables at increasing resolutions**
- Autoregressive distribution over latent variables in 2D grid ($p(z_1)p(z_2|z_1)$)
- Latent variables must be sampled in the same order when encoding or when sampling



Top-down hierarchical sampling

- **Multiple levels of latent variables at increasing resolutions**
- Autoregressive distribution over latent variables in 2D grid ($p(z_1)p(z_2|z_1)$)
- Latent variables must be sampled in the same order when encoding or when sampling
- Posterior $q(\mathbf{z}|\mathbf{x})$ is no longer gaussian
  - $q(z_2|x) = \int_{z_1} q(z_1|x)q(z_2|x, z_1)$



Variational inference with normalizing flows

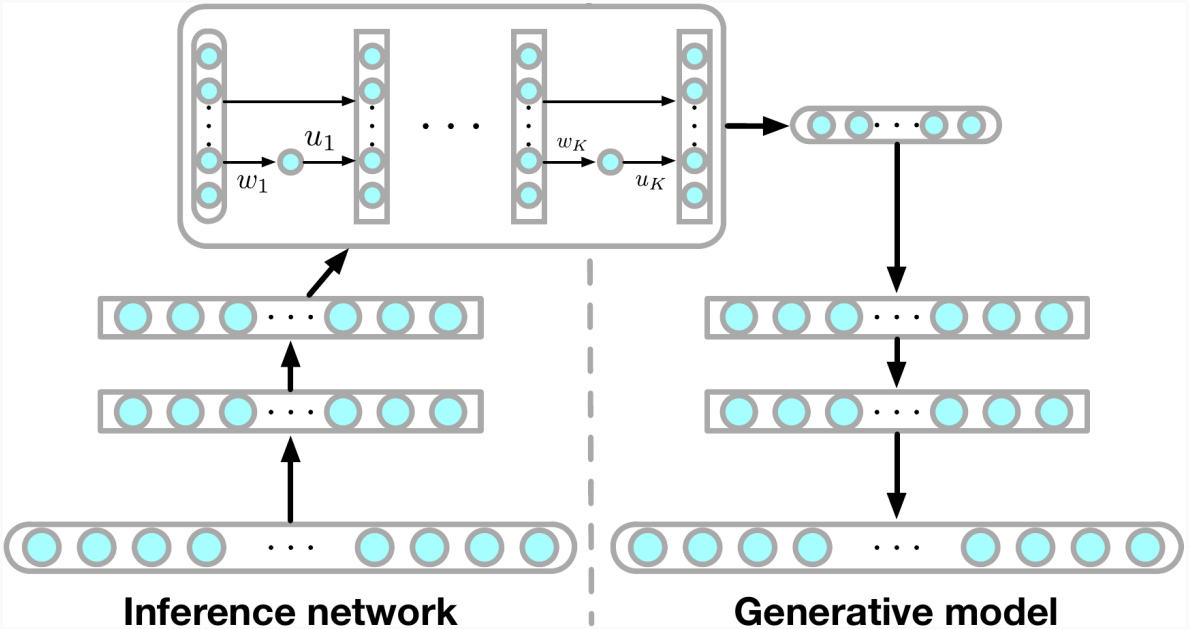
[Rezende and Mohamed, 2015]

- Improve posterior with series of invertible transformations

Variational inference with normalizing flows

[Rezende and Mohamed, 2015]

- **Improve posterior with series of invertible transformations**
- Variational inference (in VAE) uses limited class of posteriors
  - For example, Gaussian with diagonal covariance
  - Optimizing loose bound on data log-likelihood



Normalizing flows

- Let density "flow" through set of invertible transformations
- 'Reshape' the density, tractable because invertible

$$\mathbf{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z}_0),$$

$$\ln q_K(\mathbf{z}_K) = \ln q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_k} \right|$$

Normalizing flows

- Let density "flow" through set of invertible transformations
- 'Reshape' the density, tractable because invertible

$$\mathbf{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z}_0),$$

$$\ln q_K(\mathbf{z}_K) = \ln q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_k} \right|$$

- Linear-time determinant for planar and radial flows

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u} h(\mathbf{w}^\top \mathbf{z} + b)$$

$$f(\mathbf{z}) = \mathbf{z} + \beta h(\alpha, r)(\mathbf{z} - \mathbf{z_0})$$

Normalizing flows

- Let density "flow" through set of invertible transformations
- 'Reshape' the density, tractable because invertible

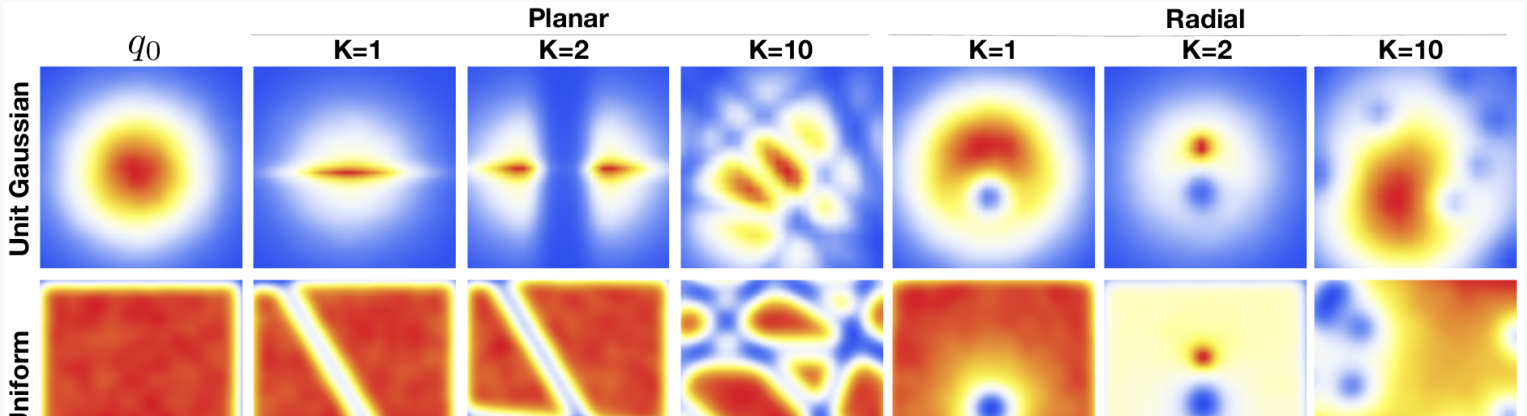
$$\mathbf{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z}_0),$$

$$\ln q_K(\mathbf{z}_K) = \ln q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_k} \right|$$

- Linear-time determinant for planar and radial flows

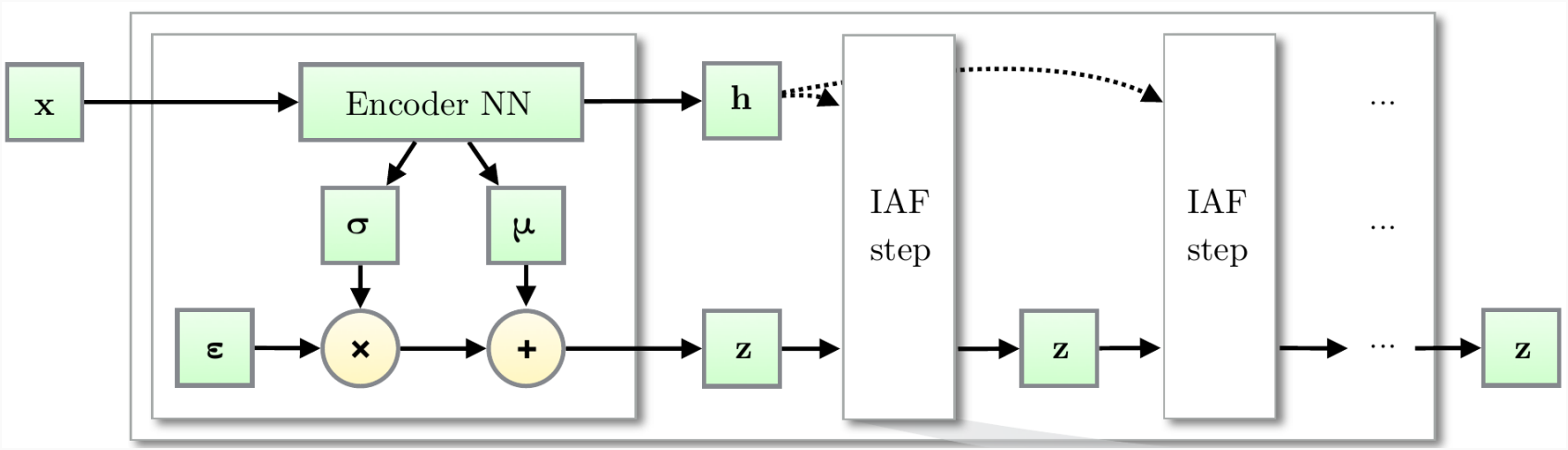
$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u} h(\mathbf{w}^\top \mathbf{z} + b)$$

$$f(\mathbf{z}) = \mathbf{z} + \beta h(\alpha, r)(\mathbf{z} - \mathbf{z_0})$$



Autoregressive flow [Kingma et al., 2016]

- Restictive flows in [Rezende and Mohamed, 2015]
  - Planar flow similar to MLP with single hidden unit
- Use autoregressive transformations in flow
  - Rich and tractable class of transformations
  - Fewer transformations needed



Autoregressive flow [Kingma et al., 2016]

- Class of affine transformations with respect to $\mathbf{z}$

$$\mathbf{z}_{t+1} = \mu_t + \sigma_t \odot \mathbf{z}_t$$

Autoregressive flow [Kingma et al., 2016]

- Class of affine transformations with respect to $\mathbf{z}$

$$\mathbf{z}_{t+1} = \mu_t + \sigma_t \odot \mathbf{z}_t$$

- Autoregressive computation of affine parameters

$$\mu_{t,i+1} = f(\mathbf{z}_{t,1:i}) \qquad \sigma_{t,i+1} = g(\mathbf{z}_{t,1:i})$$

Autoregressive flow [Kingma et al., 2016]

- Class of affine transformations with respect to $\mathbf{z}$

$$\mathbf{z}_{t+1} = \mu_t + \sigma_t \odot \mathbf{z}_t$$

- Autoregressive computation of affine parameters

$$\mu_{t,i+1} = f(\mathbf{z}_{t,1:i}) \qquad \sigma_{t,i+1} = g(\mathbf{z}_{t,1:i})$$

- Triangular Jacobian, log-determinant $\sum_{i=1}^{D} \log \sigma_{t,i}$

Autoregressive flow [Kingma et al., 2016]

- Class of affine transformations with respect to $\mathbf{z}$

$$\mathbf{z}_{t+1} = \mu_t + \sigma_t \odot \mathbf{z}_t$$

- Autoregressive computation of affine parameters

$$\mu_{t,i+1} = f(\mathbf{z}_{t,1:i}) \qquad \sigma_{t,i+1} = g(\mathbf{z}_{t,1:i})$$

- Triangular Jacobian, log-determinant $\sum_{i=1}^{D} \log \sigma_{t,i}$

- Parallel computations of $\mathbf{z}_t, \mu_t, \sigma_t$ across dimensions

Autoregressive flow [Kingma et al., 2016]

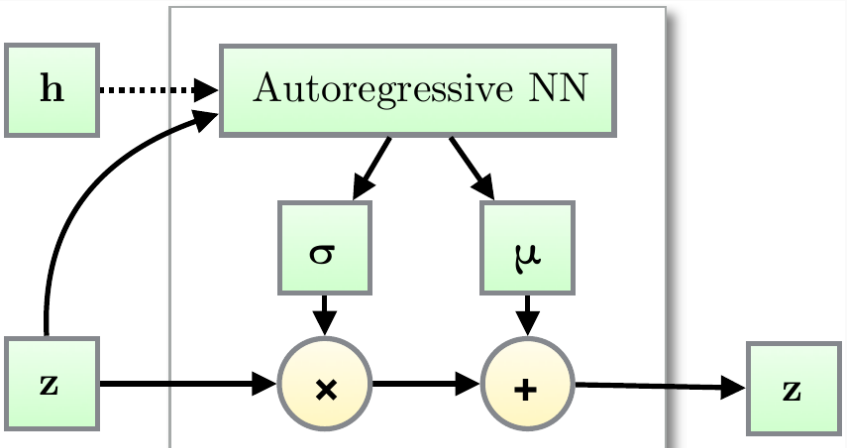
- Class of affine transformations with respect to $\mathbf{z}$

$$\mathbf{z}_{t+1} = \mu_t + \sigma_t \odot \mathbf{z}_t$$

- Autoregressive computation of affine parameters

$$\mu_{t,i+1} = f(\mathbf{z}_{t,1:i}) \qquad \sigma_{t,i+1} = g(\mathbf{z}_{t,1:i})$$

- Triangular Jacobian, log-determinant $\sum_{i=1}^{D} \log \sigma_{t,i}$
- Parallel computations of $\mathbf{z}_t, \mu_t, \sigma_t$ across dimensions
- Free to chose form of autoregressive dependency



Recap

Ways to improve the tightness of the ELBO:

- Importance weighted autoencoder
- Hierarchical top-down sampling
- Flow transormations

Beyond conditional independance assumptions

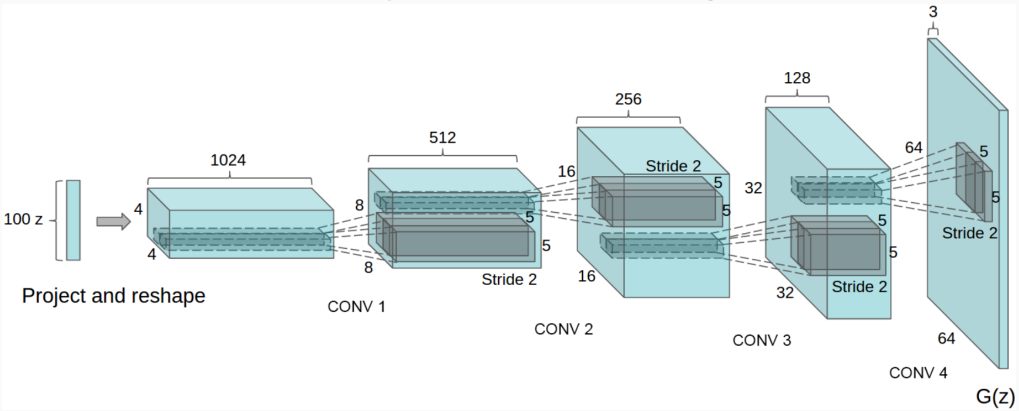
Standard VAE decoders make a conditional independance on x given z.

- Similar to L_2 regression. Bad metric of image similarity
- Leads to blurry images, and over-generalization

Hybrid VAE autoregressive models [Gulrajani et al., 2017b, Chen et al., 2017]

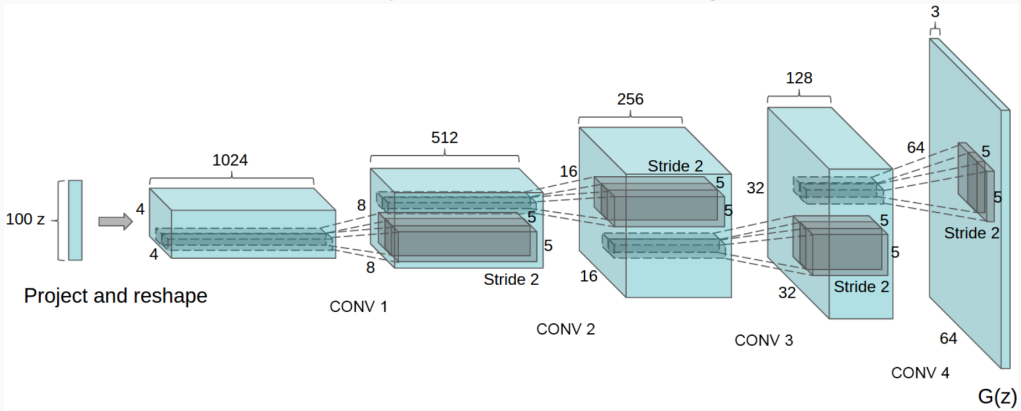
Hybrid VAE autoregressive models [Gulrajani et al., 2017b, Chen et al., 2017]

- Variational autoencoder
  - Latent variable **z** generates global dependencies
  - Pixels conditionally independent given code

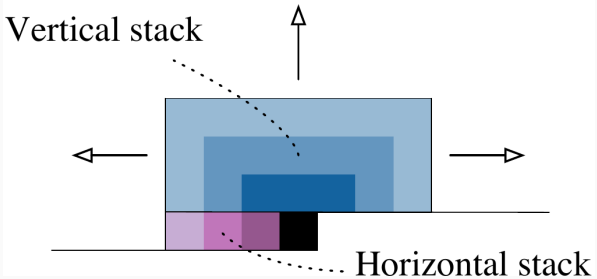


Hybrid VAE autoregressive models [Gulrajani et al., 2017b, Chen et al., 2017]

- Variational autoencoder
  - Latent variable **z** generates global dependencies
  - Pixels conditionally independent given code

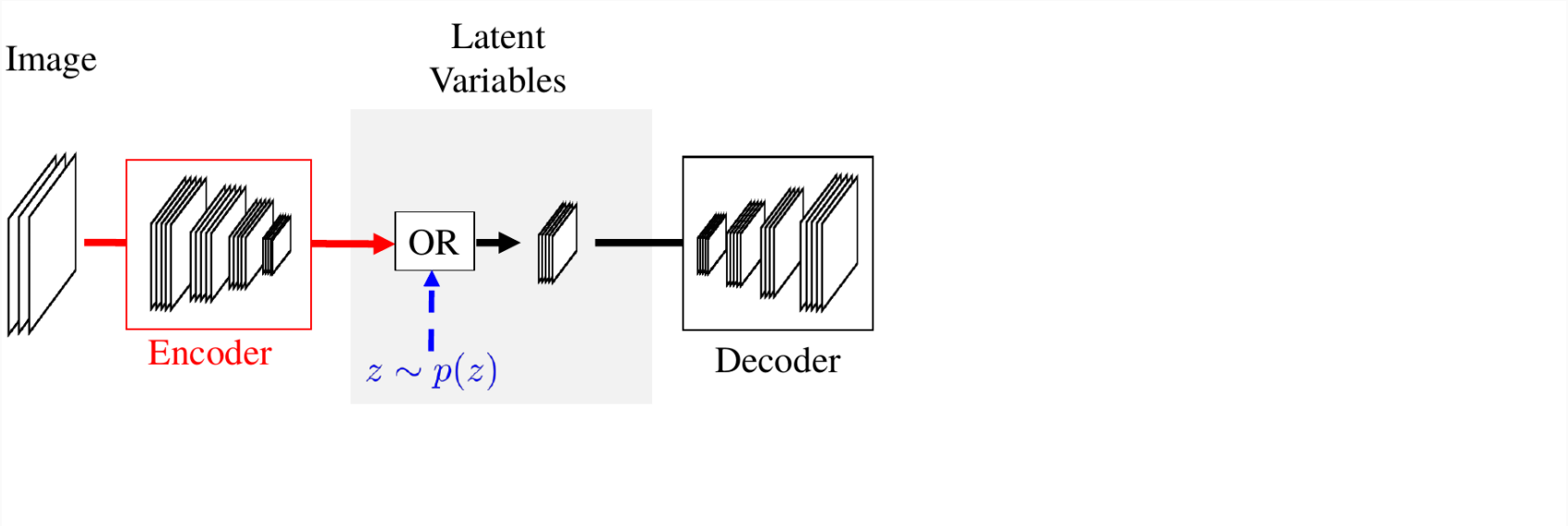


- Autoregressive PixelCNN
  - Needs many layers to induce long-range dependencies
  - Doesn't learn latent representation



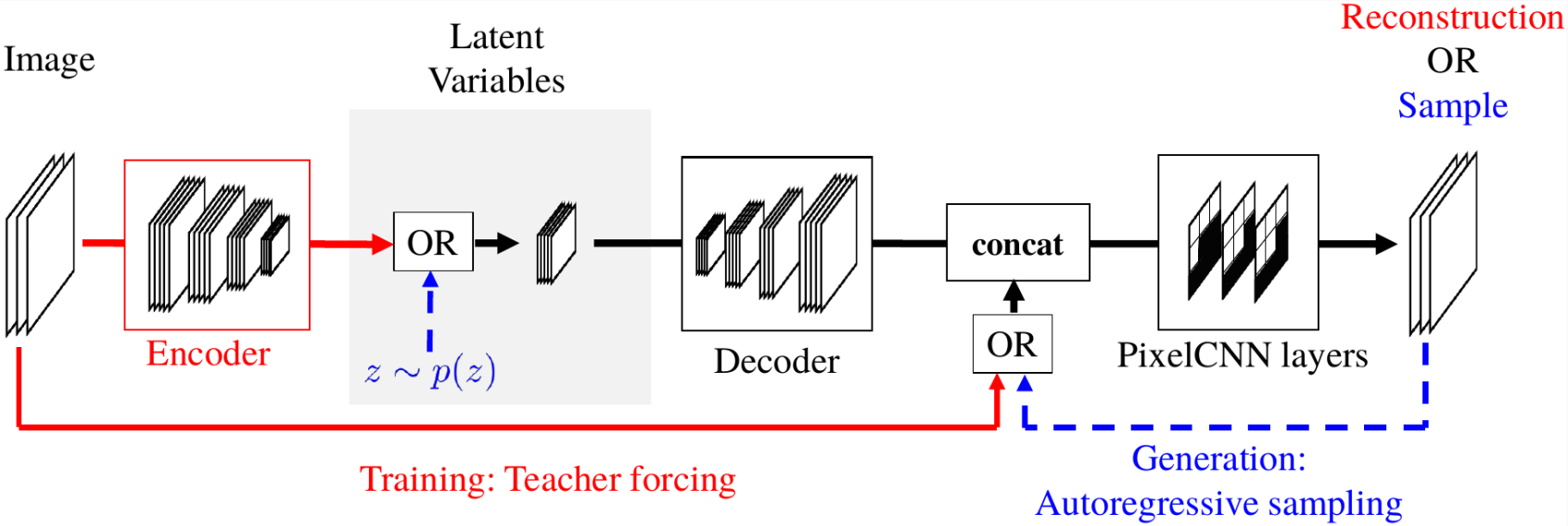
Hybrid PixelVAE model [Gulrajani et al., 2017b]

- Latent var. input to deterministic upsampling decoder $f(\mathbf{z})$



Hybrid PixelVAE model [Gulrajani et al., 2017b]

- Latent var. input to deterministic upsampling decoder $f(\mathbf{z})$
- Pixel-CNN layers induce local pixel dependencies

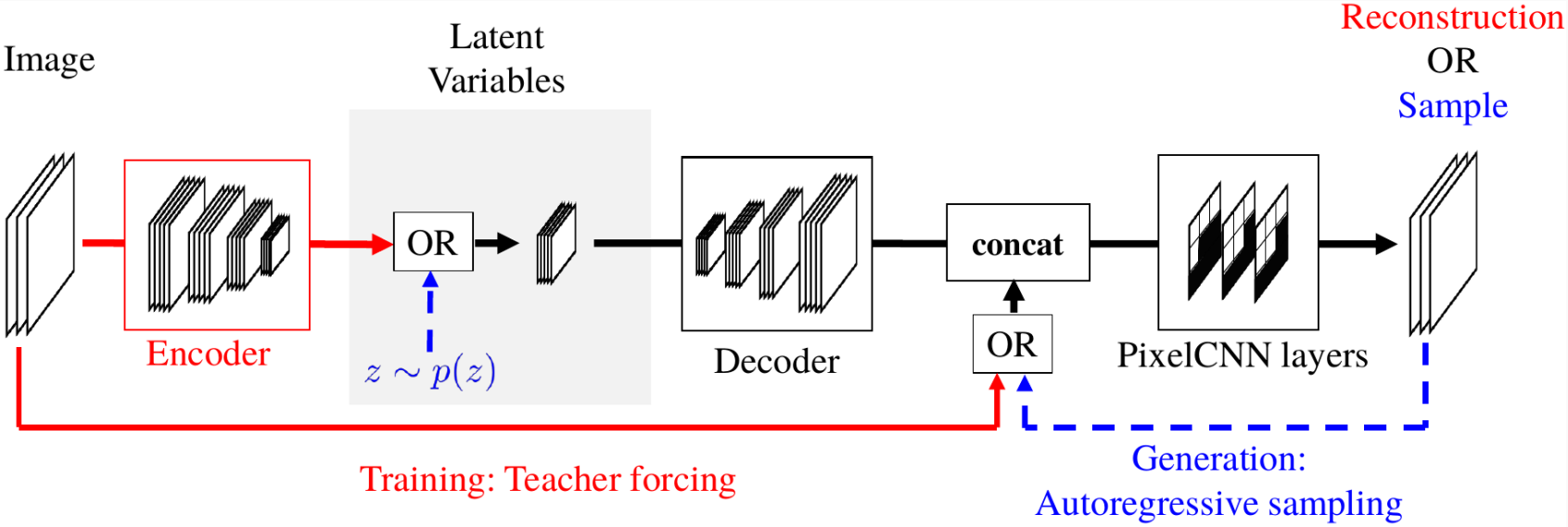


Hybrid PixelVAE model [Gulrajani et al., 2017b]

- Latent var. input to deterministic upsampling decoder $f(\mathbf{z})$
- Pixel-CNN layers induce local pixel dependencies

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{9}$$

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{z}) \prod_i p(x_i | \mathbf{x}_{<i}, f(\mathbf{z})) \tag{10}$$



Samples PixelVAE model LSUN dataset

- Model with three levels of stochasticity
  - Latent variables at 1×1
  - Latent variables at 8×8
  - PixelCNN at 64×64

Samples PixelVAE model LSUN dataset

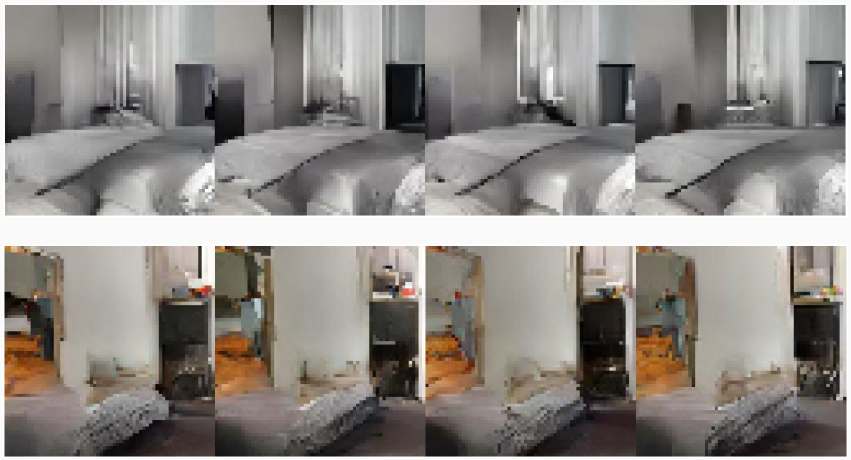
- Model with three levels of stochasticity
  - Latent variables at 1×1
  - Latent variables at 8×8
  - PixelCNN at 64×64

Re-sampling PixelCNN only

Samples PixelVAE model LSUN dataset

- Model with three levels of stochasticity
  - Latent variables at 1×1
  - Latent variables at 8×8
  - PixelCNN at 64×64

Re-sampling PixelCNN only



Samples PixelVAE model LSUN dataset

- Model with three levels of stochasticity
  - Latent variables at 1×1
  - Latent variables at 8×8
  - PixelCNN at 64×64

Re-sampling PixelCNN only

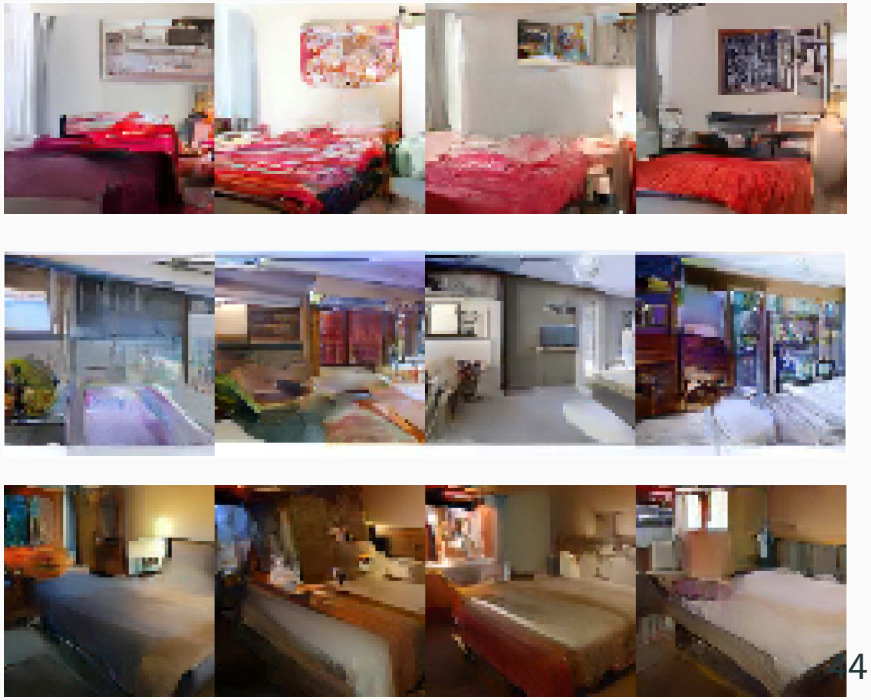
Samples PixelVAE model LSUN dataset

- Model with three levels of stochasticity
  - Latent variables at 1×1
  - Latent variables at 8×8
  - PixelCNN at 64×64

- Hierarchical representation learning

Re-sampling PixelCNN only

Re-sampling 8×8 + PixelCNN



References i


Arjovsky, M., Chintala, S., and Bottou, L. (2017).
Wasserstein generative adversarial networks.
In *ICML*.


Burda, Y., Salakhutdinov, R., and Grosse, R. (2016).
Importance weighted autoencoders.
In *ICLR*.


Chen, X., Kingma, D., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. (2017).
Variational lossy autoencoder.
In *ICLR*.


Donahue, J., Krähenbühl, P., and Darrell, T. (2017).
Adversarial feature learning.
In *ICLR*.

References ii


Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017a).
Improved training of Wasserstein GANs.
In *NeurIPS*.


Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., and Courville, A. (2017b).
PixelVAE: A latent variable model for natural images.
In *ICLR*.


Kingma, D., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016).
Improved variational inference with inverse autoregressive flow.

In *NeurIPS*.

References iii


Rezende, D. and Mohamed, S. (2015).
Variational inference with normalizing flows.
In *ICML*.


Zhu, J.-Y., Park, T., Isola, P., and Efros, A. (2017).
Unpaired image-to-image translation using cycle-consistent adversarial networks.
In *ICCV*.