DSC291: Machine Learning with Few Labels

Deep generative modeling
Generative adversarial learning

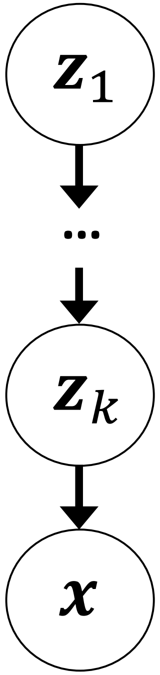
Zhiting Hu

Lecture 14, February 10, 2023

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

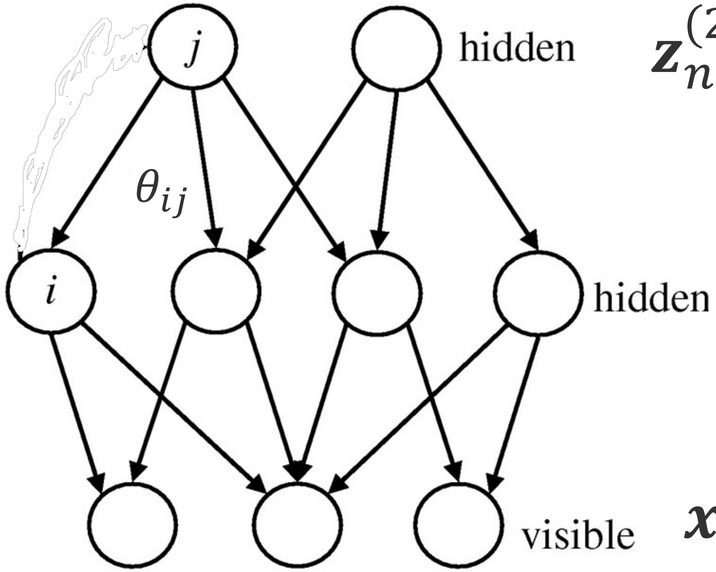
Deep generative models

- Define probabilistic distributions over a set of variables
- "Deep" means multiple layers of hidden variables!



Early forms of deep generative models

- Hierarchical Bayesian models
  - Sigmoid brief nets [Neal 1992]



$\boldsymbol{z}_n^{(2)} = \{0,1\}^J$

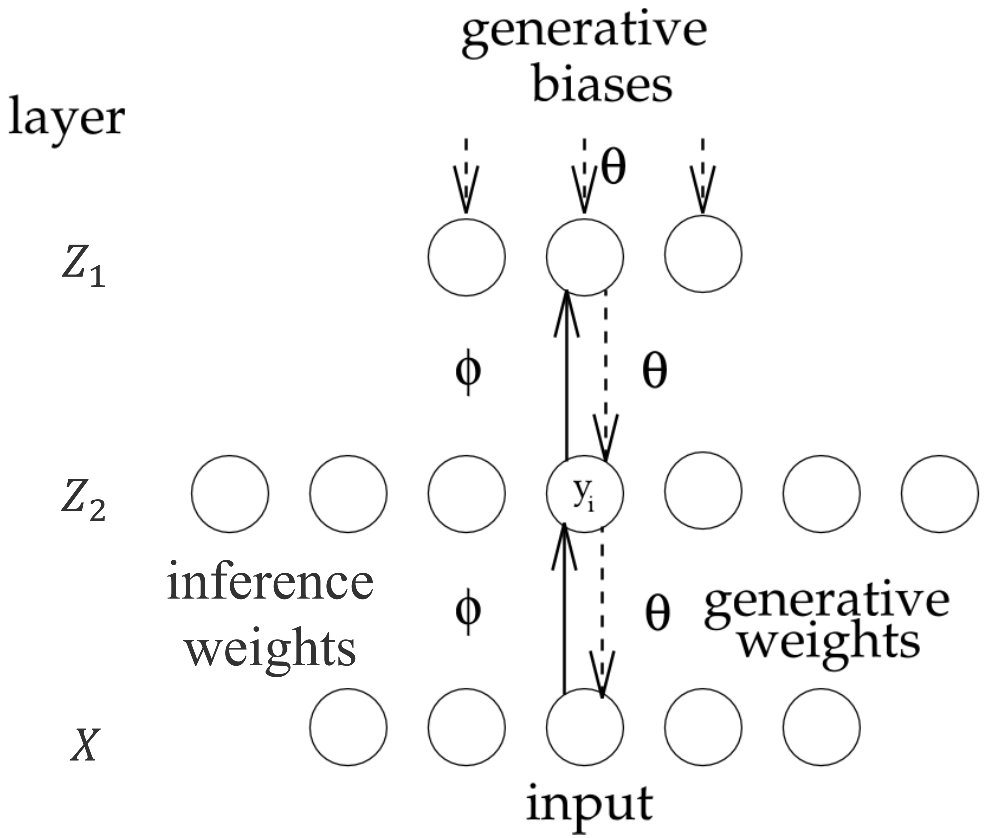
$\boldsymbol{z}_n^{(1)} = \{0,1\}^I$

$\boldsymbol{x}_n = \{0,1\}^K$

$$p\left(x_{kn} = 1 \middle| \boldsymbol{\theta}_k, \boldsymbol{z}_n^{(1)}\right) = \sigma\left(\boldsymbol{\theta}_k^T \boldsymbol{z}_n^{(1)}\right)$$
$$p\left(z_{in}^{(1)} = 1 \middle| \boldsymbol{\theta}_i, \boldsymbol{z}_n^{(2)}\right) = \sigma\left(\boldsymbol{\theta}_i^T \boldsymbol{z}_n^{(2)}\right)$$

Early forms of deep generative models

- Hierarchical Bayesian models
  - Sigmoid brief nets [Neal 1992]
- Neural network models
  - Helmholtz machines [Dayan et al.,1995]



[Dayan et al. 1995]

Early forms of deep generative models

- Hierarchical Bayesian models
  - Sigmoid brief nets [Neal 1992]

- Neural network models
  - Helmholtz machines [Dayan et al.,1995]
  - Predictability minimization [Schmidhuber 1995]

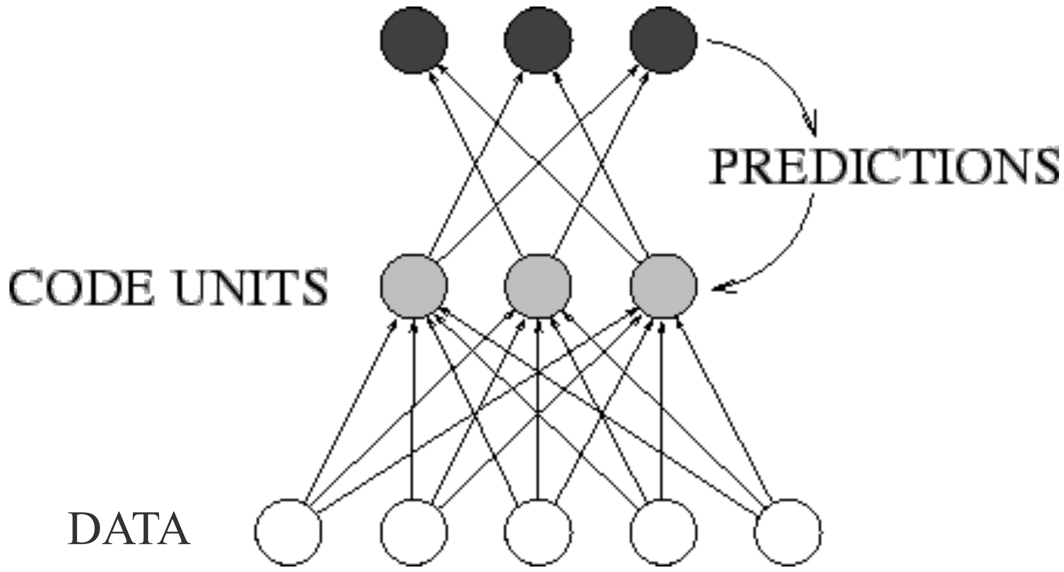


Figure courtesy: Schmidhuber 1996

Early forms of deep generative models

- Training of DGMs via an EM style framework
  - Sampling / data augmentation

$$\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2\}$$
$$\boldsymbol{z}_1^{new} \sim p(\boldsymbol{z}_1 | \boldsymbol{z}_2, \boldsymbol{x})$$
$$\boldsymbol{z}_2^{new} \sim p(\boldsymbol{z}_2 | \boldsymbol{z}_1^{new}, \boldsymbol{x})$$

  - Variational inference

$$\log p(\boldsymbol{x}) \geq \mathrm{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}, \boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z})) := \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$$

$$\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$$

  - Wake sleep

Wake: $\min_\theta \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$

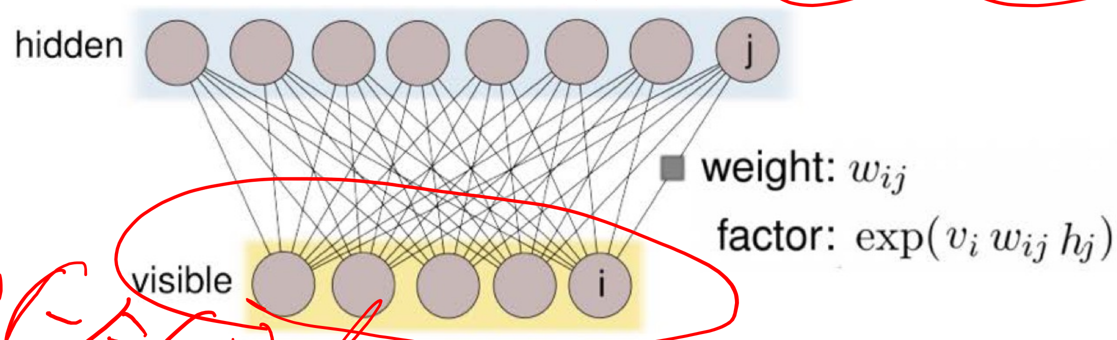
Sleep: $\min_\phi \mathbb{E}_{p_\theta(x|z)}[\log q_\phi(z|x)]$

Resurgence of deep generative models

- Restricted Boltzmann machines (RBMs) [Smolensky, 1986]
  - Building blocks of deep probabilistic models

$$p(v,h) = \frac{\exp\{\sum_{i,j} v_i w_{ij} h_j\}}{Z} \qquad -E(v,h)$$

energy-based function



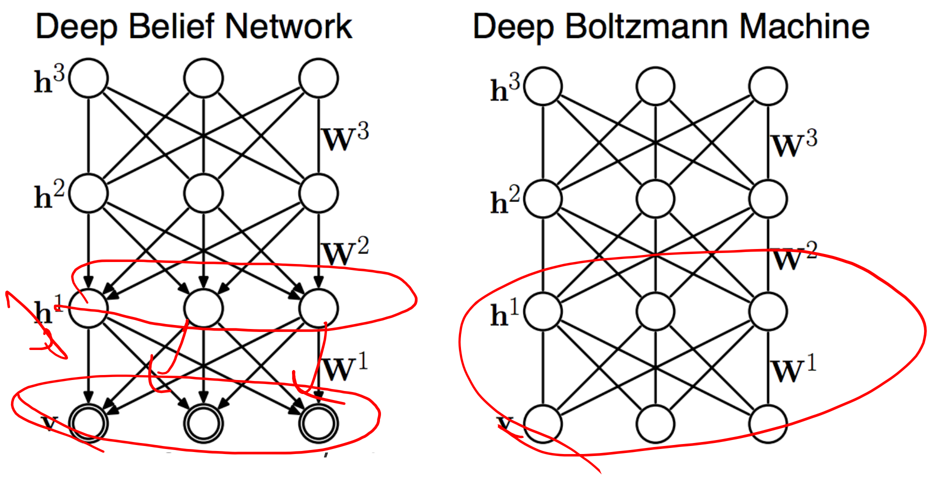
$$Z = \int_x \exp(-E(x))\,dx$$

$$p(x) = \frac{\exp\{-E(x)\}}{Z}$$

Gibbs distribution

Resurgence of deep generative models

- Restricted Boltzmann machines (RBMs) [Smolensky, 1986]
  - Building blocks of deep probabilistic models
- Deep belief networks (DBNs) [Hinton et al., 2006]
  - Hybrid graphical model
  - Inference in DBNs is problematic due to explaining away
- Deep Boltzmann Machines (DBMs) [Salakhutdinov & Hinton, 2009]
  - Undirected model

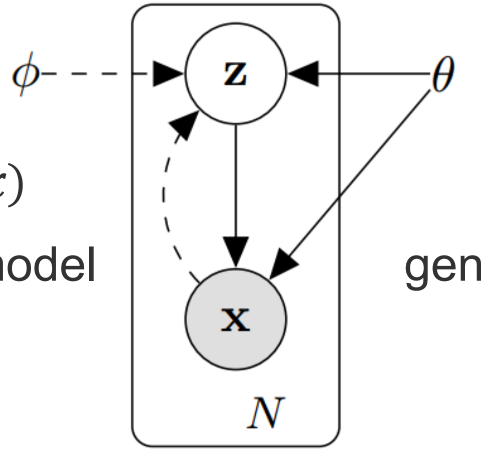


Resurgence of deep generative models

- Variational autoencoders (VAEs) [Kingma & Welling, 2014]
  / Neural Variational Inference and Learning (NVIL) [Mnih & Gregor, 2014]

$q_\phi(\boldsymbol{z}|\boldsymbol{x})$

inference model



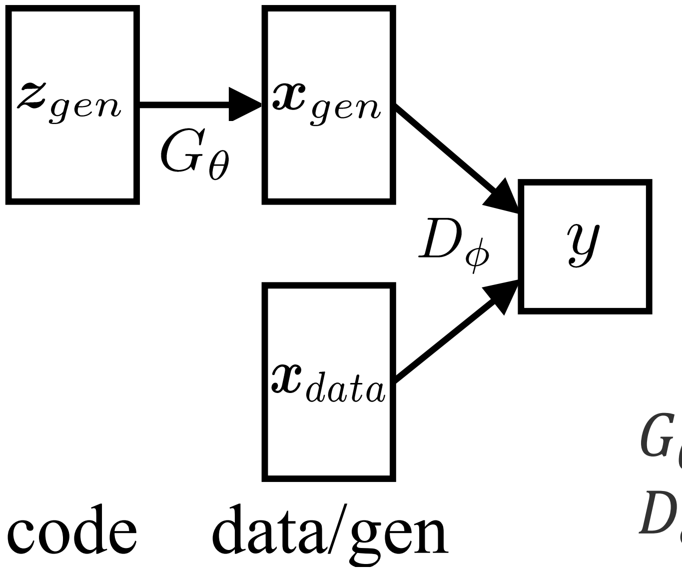
$p_\theta(\boldsymbol{x}|\boldsymbol{z})$

generative model

Figure courtesy: Kingma & Welling, 2014

Resurgence of deep generative models

- Variational autoencoders (VAEs) [Kingma & Welling, 2014]
  / Neural Variational Inference and Learning (NVIL) [Mnih & Gregor, 2014]
- Generative adversarial networks (GANs) [Goodfellow et al,. 2014]



G_θ: generative model ?
D_ϕ: discriminator

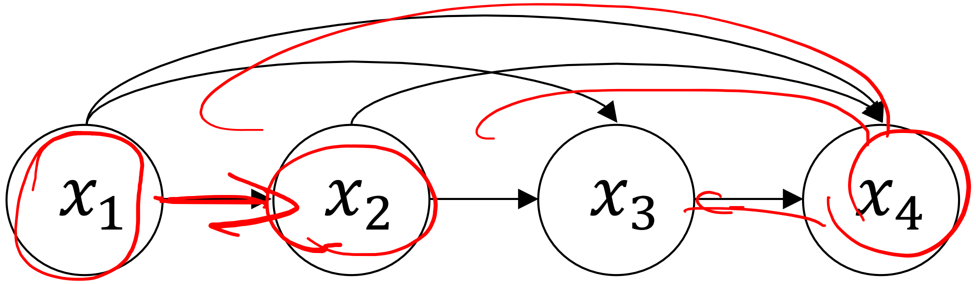
Resurgence of deep generative models

- Variational autoencoders (VAEs) [Kingma & Welling, 2014]
  / Neural Variational Inference and Learning (NVIL) [Mnih & Gregor, 2014]
- Generative adversarial networks (GANs) [Goodfellow et al,. 2014]
- Generative moment matching networks (GMMNs) [Li et al., 2015; Dziugaite et al., 2015]

Resurgence of deep generative models

- Variational autoencoders (VAEs) [Kingma & Welling, 2014] / Neural Variational Inference and Learning (NVIL) [Mnih & Gregor, 2014]
- Generative adversarial networks (GANs) [Goodfellow et al,. 2014]
- Generative moment matching networks (GMMNs) [Li et al., 2015; Dziugaite et al., 2015]
- Autoregressive neural networks
- Reversible architectures
- Diffusion models

flow



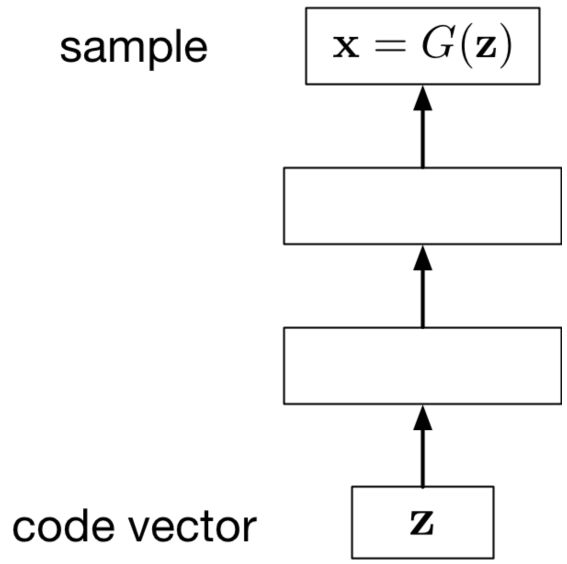
Generative Adversarial Networks

Recap: Implicit Generative Models

$z \sim p_0(z)$

$x = G_\theta(z)$

- Implicit generative models implicitly define a probability distribution
- Start by sampling the code vector $\mathbf{z}$ from a fixed, simple distribution (e.g. spherical Gaussian)
- The generator network computes a differentiable function G mapping $\mathbf{z}$ to an $\mathbf{x}$ in data space



- a stochastic process to simulate data $\boldsymbol{x}$
- Intractable to evaluate likelihood

Recap: Generative Adversarial Nets (GANs)

- Learning
  - A minimax game between the generator and the discriminator
  - Train D to maximize the probability of assigning the correct label to both training examples and generated samples
  - Train G to fool the discriminator

$$\max_D \mathcal{L}_D = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})} [\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim G(\boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z})} [\log(1 - D(\boldsymbol{x}))]$$

$$\min_G \mathcal{L}_G = \mathbb{E}_{\boldsymbol{x} \sim G(\boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z})} [\log(1 - D(\boldsymbol{x}))] \,.$$

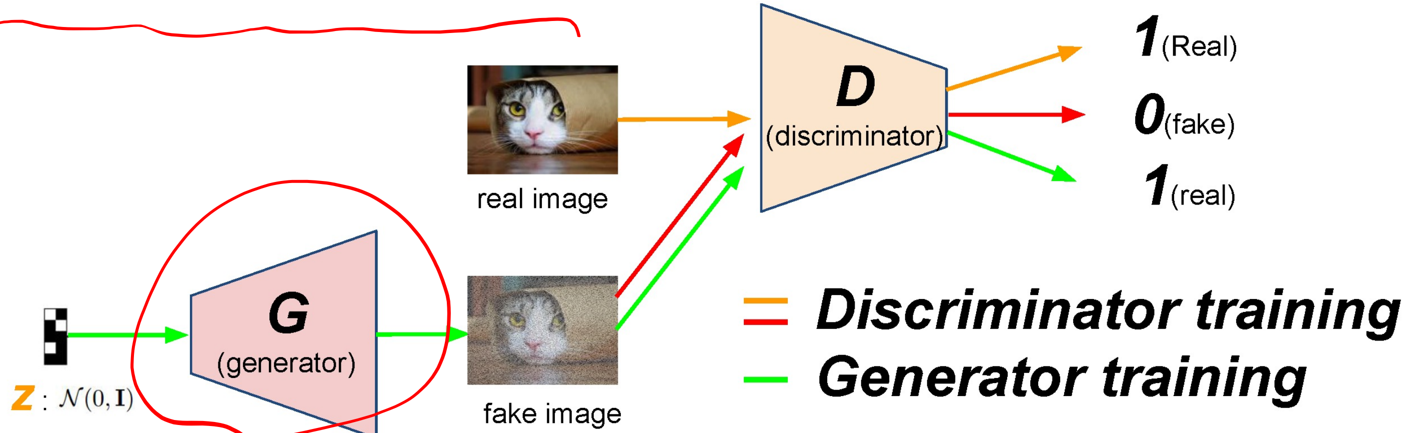


Figure courtesy: Kim

Recap: Optimality of GANs

- Objectives:

$$\max_D \mathcal{L}_D = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z}), \boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1 - D(\boldsymbol{x}))\right]$$
$$\min_G \mathcal{L}_G = \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z}), \boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log(1 - D(\boldsymbol{x}))\right].$$

- Global optimality: $p_g = p_{data}$

$JSD(P_\theta, P_{data})$

KL $\neq$ KL

Recap: A better loss function: non-saturating GAN

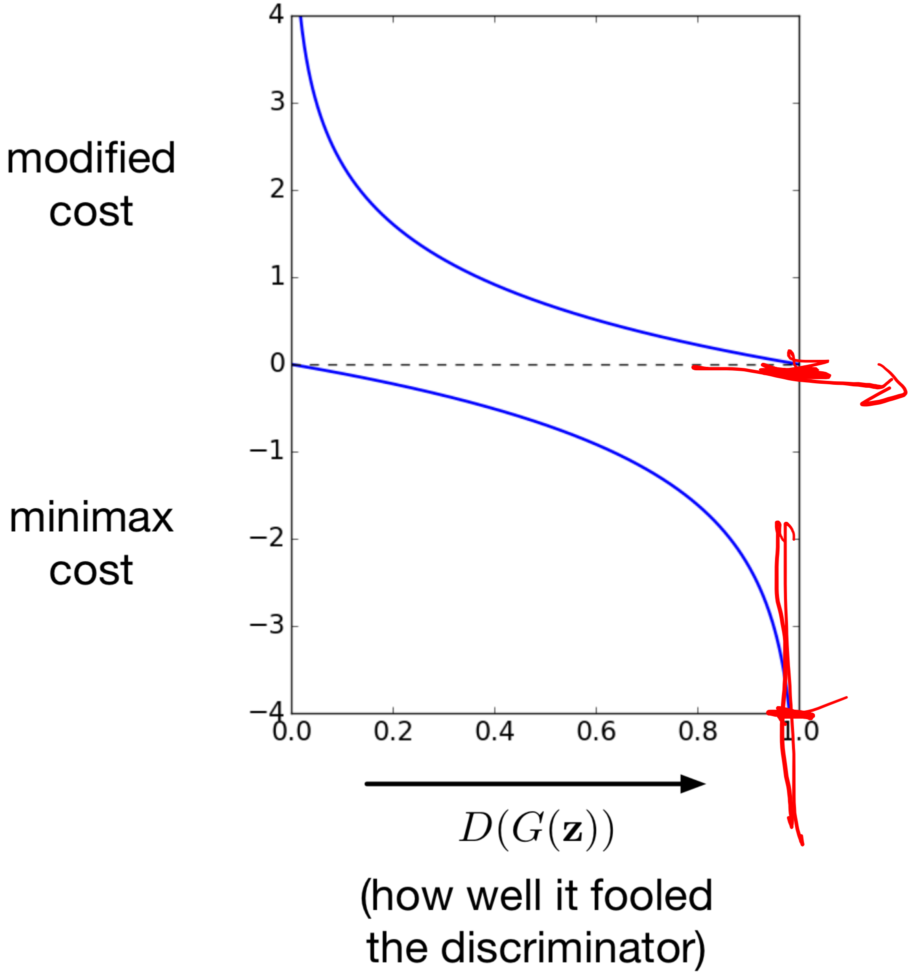
- Original minimax cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- Modified generator cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$$

- This fixes the saturation problem.



Wasserstein GAN (WGAN)

- If our data are on a low-dimensional manifold of a high dimensional space, the model's manifold and the true data manifold can have a negligible intersection in practice

[Arjovsky et al., 2017] Slide adapted from bhiksha

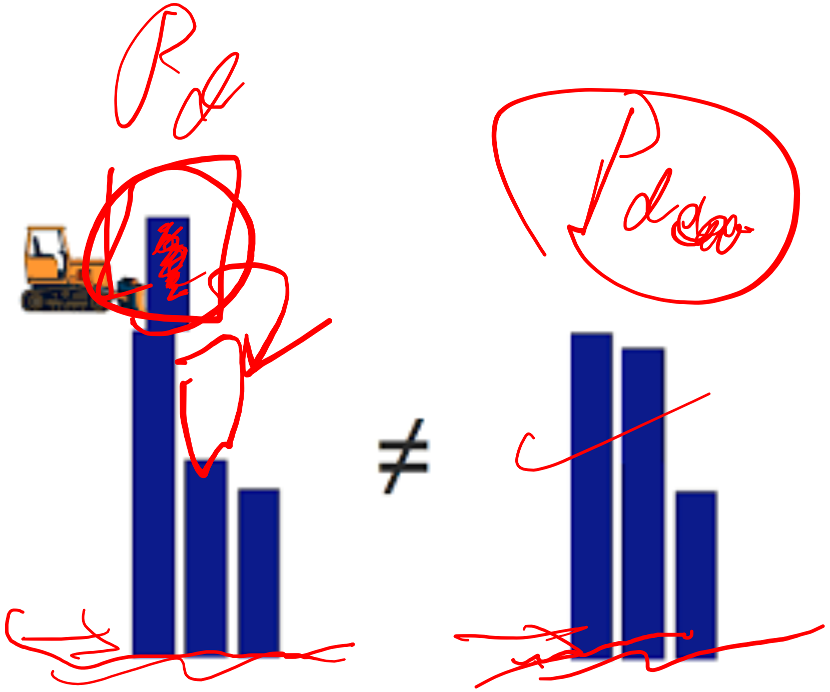
Wasserstein GAN (WGAN)

- If our data are on a low-dimensional manifold of a high dimensional space, the model's manifold and the true data manifold can have a negligible intersection in practice
- The loss function and gradients may not be continuous and well behaved

[Arjovsky et al., 2017] Slide adapted from bhiksha

Wasserstein GAN (WGAN)

- If our data are on a low-dimensional manifold of a high dimensional space, the model's manifold and the true data manifold can have a negligible intersection in practice
- The loss function and gradients may not be continuous and well behaved
- The Wasserstein Distance is well defined
    - Earth Mover's Distance
    - Minimum transportation cost for making one pile of dirt in the shape of one probability distribution to the shape of the other distribution



[Arjovsky et al., 2017] Slide adapted from bhiksha

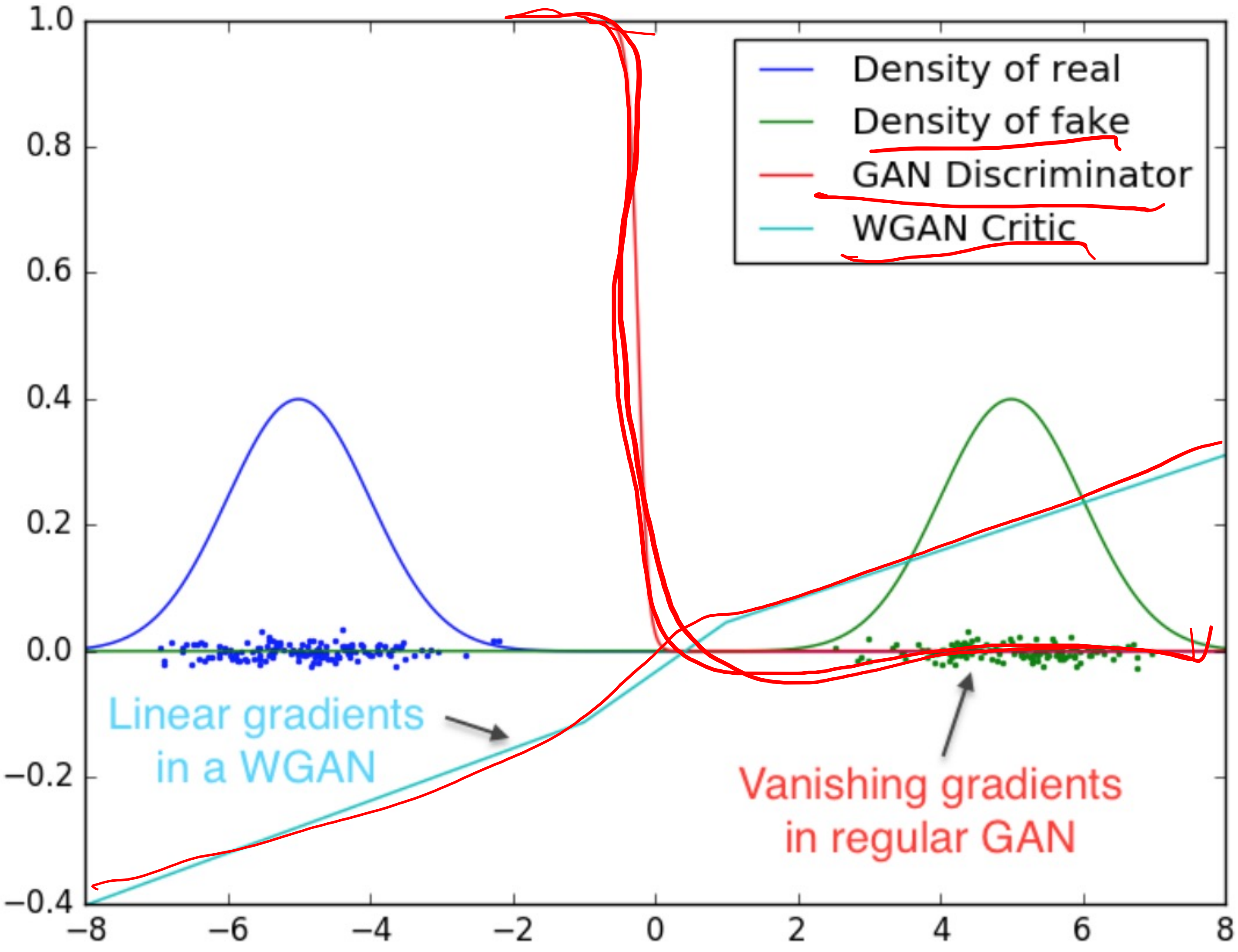
Wasserstein GAN (WGAN)

- Objective

$$\min_{p_g} \quad W(p_{data}, p_g) = \frac{1}{K} \sup_{||D||_L \le K} \mathrm{E}_{x\sim p_{data}}[D(x)] - \mathrm{E}_{x\sim p_g}[D(x)]$$

- $||D||_L \le K$: K- Lipschitz continuous
- Use gradient-clipping to ensure D has the Lipschitz continuity

WGAN vs Vanilla GAN



Standard Equation and GANs

- Recall SE:

$$\min_{q,\theta} -\alpha\mathbb{H}(q) + \beta\mathbb{D}\Big(q(\boldsymbol{x}), p_\theta(\boldsymbol{x}) \Big) - \mathbb{E}_{q(\boldsymbol{x})}\Big[f(\boldsymbol{x}) \Big]$$

- In MLE, f is a fixed function

$$f \coloneqq f_{data}(\boldsymbol{x}\,;\,\mathcal{D}) = \log \mathbb{E}_{\boldsymbol{x}^*\sim\mathcal{D}}\left[\mathbb{1}_{\boldsymbol{x}^*}(\boldsymbol{x})\right]$$

- Intuitively, see f as a similarity metric that measures similarity of sample $\boldsymbol{x}$ against real data $\mathcal{D}$
- Instead of the above manually fixed metric, can we **learn** a metric f_ϕ?

Standard Equation and GANs

- Augment the standard objective to account for ϕ:

$$\min_{\theta} \max_{\phi} \min_{q} - \alpha \mathbb{H}(q) + \beta \mathbb{D}\Big(q(\boldsymbol{x}), p_{\theta}(\boldsymbol{x}) \Big) - \mathbb{E}_{q(\boldsymbol{x})}\Big[f_{\phi}(\boldsymbol{x}) \Big] + \mathbb{E}_{p_d(\boldsymbol{x})}\Big[f_{\phi}(\boldsymbol{x}) \Big]$$

- Set $\alpha = 0, \beta = 1$. Under mild conditions, the objective recovers:
  - Vanilla GAN [Goodfellow et al., 2014], when $\mathbb{D}$ is JS-divergence and f_{ϕ} is a binary classifier
  - f-GAN [Nowozin et al., 2016], when $\mathbb{D}$ is f-divergence
  - W-GAN [Arjovsky et al., 2017], when $\mathbb{D}$ is Wasserstein distance and f_{ϕ} is a 1-Lipschitz function

Progressive GAN

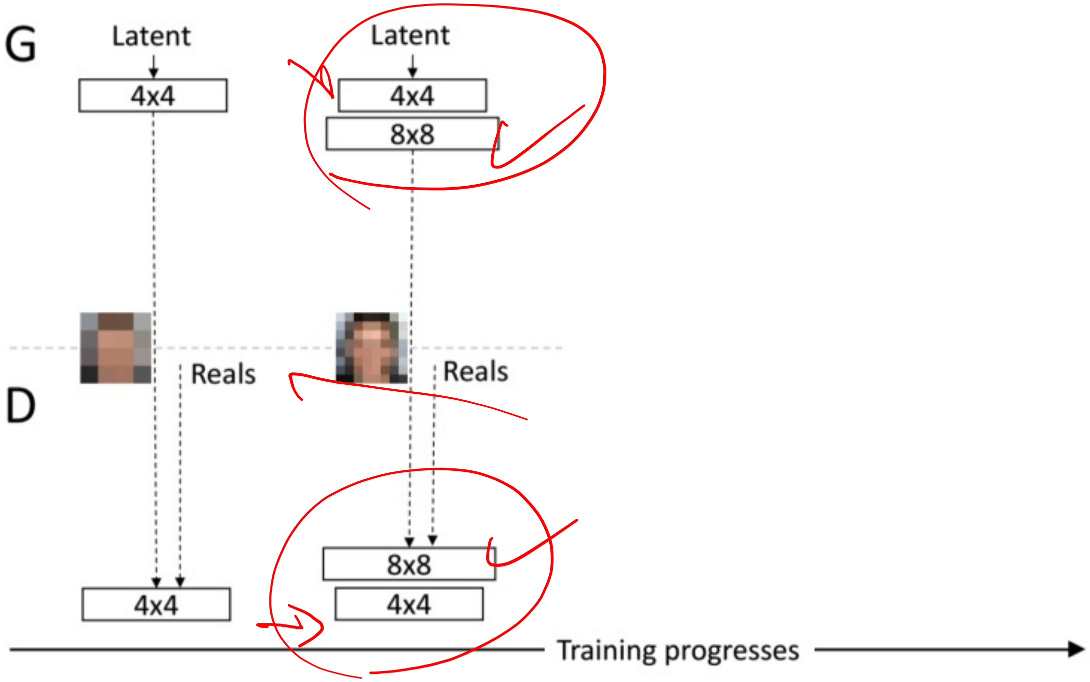
Low resolution images

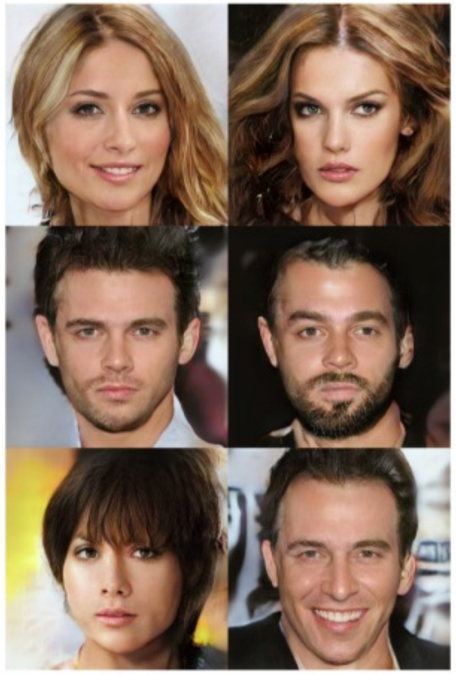
[Karras et al., 2018]

Progressive GAN

Low resolution images

add in additional layers





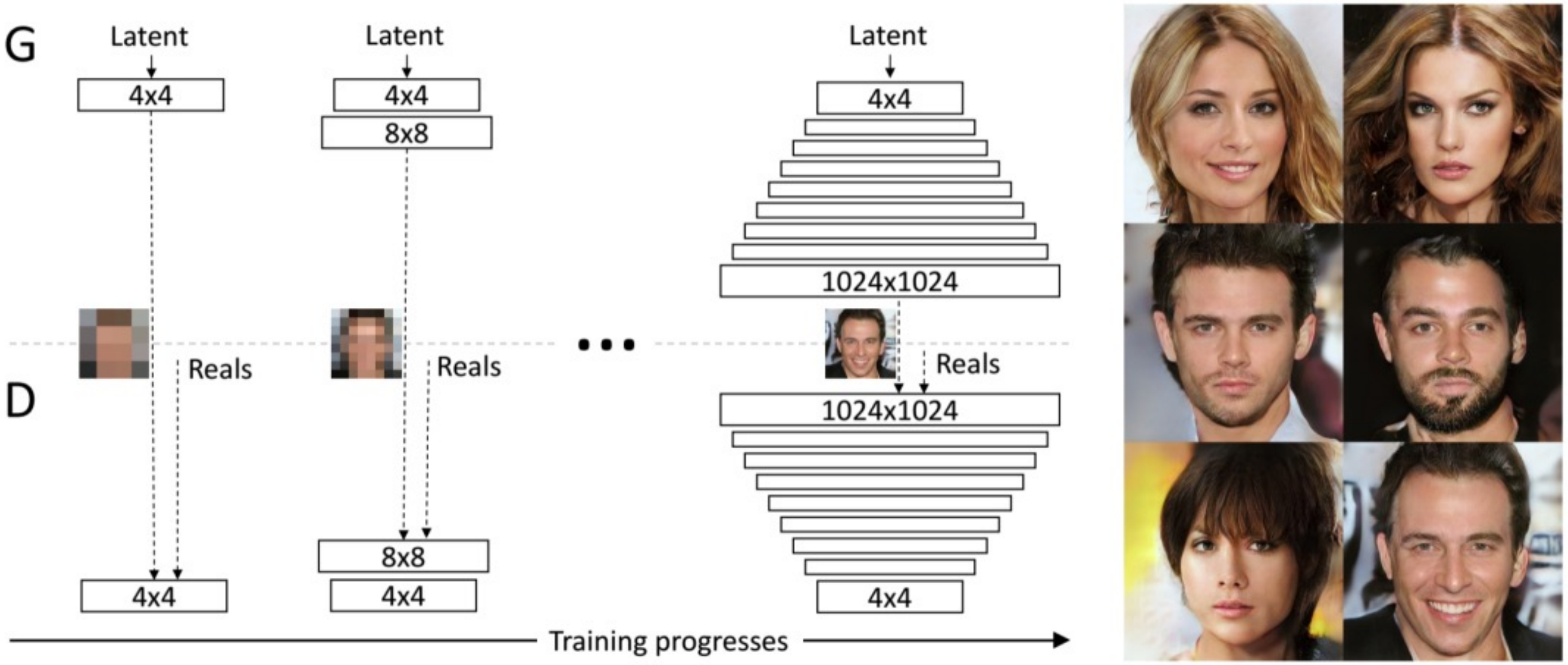
[Karras et al., 2018]

Progressive GAN

Low resolution images

add in
additional
layers

High resolution images



[Karras et al., 2018]

BigGAN

[Brock et al., 2018]

BigGAN

- GANs benefit dramatically from scaling

[Brock et al., 2018]

BigGAN

- GANs benefit dramatically from scaling
- 2x – 4x more parameters
- 8x larger batch size
- Simple architecture changes that improve scalability

[Brock et al., 2018]

BigGAN

- GANs benefit dramatically from scaling
- 2x – 4x more parameters
- 8x larger batch size
- Simple architecture changes that improve scalability

[Brock et al., 2018]

BigGAN

- GANs benefit dramatically from scaling
- 2x – 4x more parameters
- 8x
- Sir

[Brock et al., 2018]

Normalizing Flow (NF)

Normalizing Flow (NF)

$z \sim p_\theta(z)$ $x \sim q_\theta(x)$

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

Figure courtesy: Lilian Weng

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

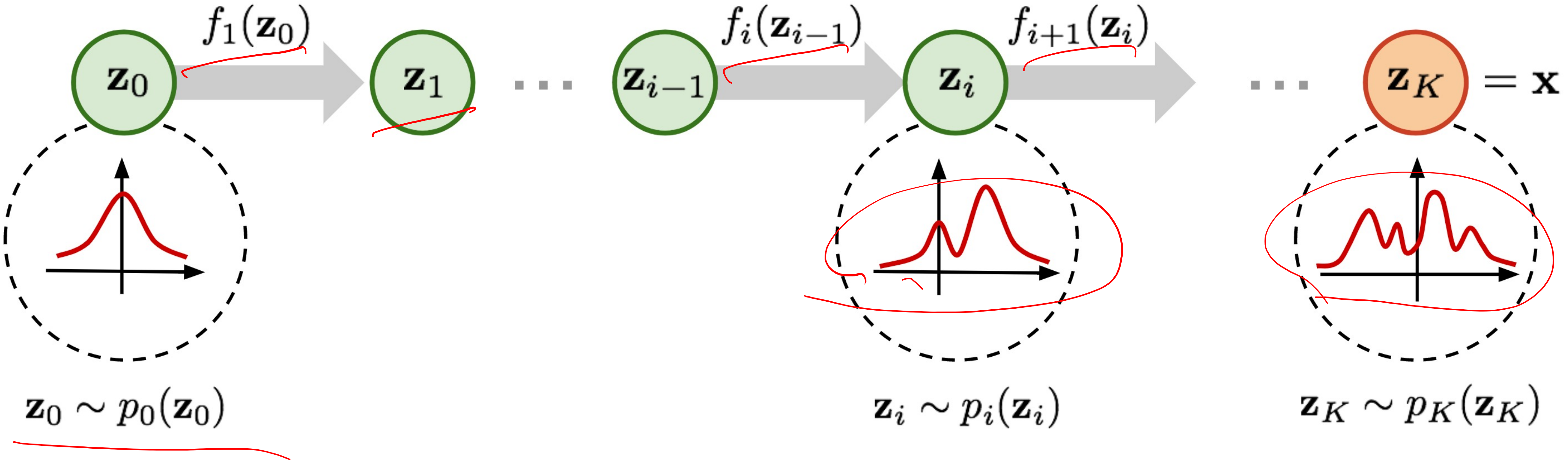
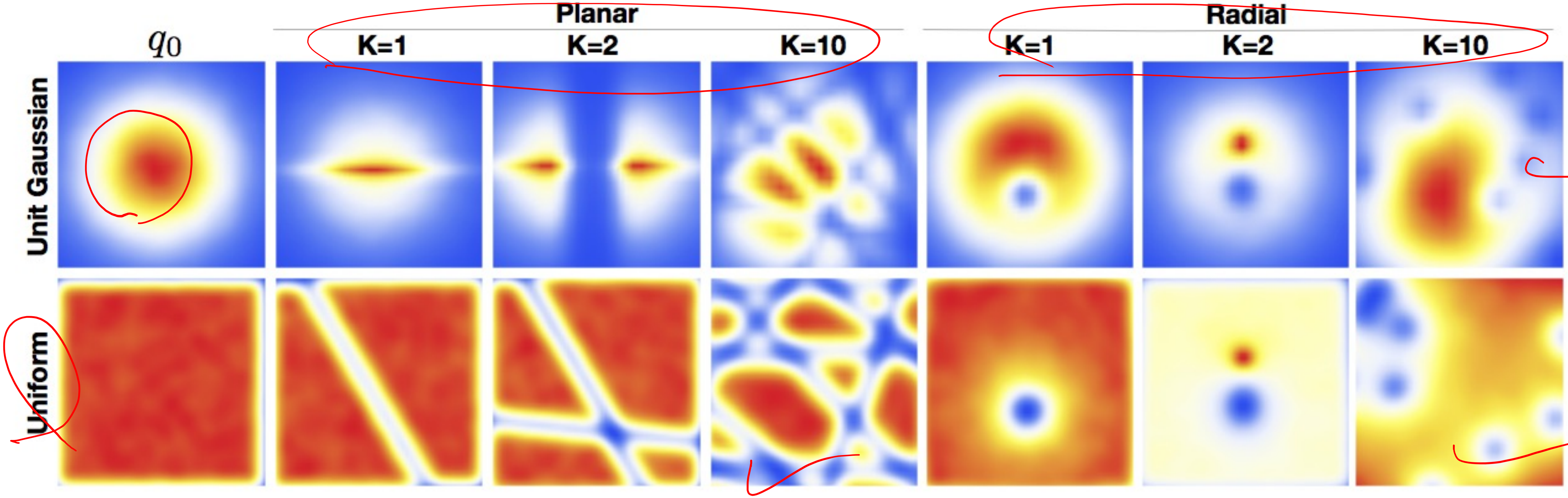


Figure courtesy: Lilian Weng

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions



[Rezende & Mohamed, 2015]

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

inference: $\boldsymbol{z} = f^{-1}(\boldsymbol{x})$

Transformation function f

- Invertible

$p_\theta(x)$ $x \leftarrow g_\theta(z)$

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

inference: $\boldsymbol{z} = f^{-1}(\boldsymbol{x})$

density: $p(\boldsymbol{x}) = p(\boldsymbol{z}) \left|\det \frac{d\boldsymbol{z}}{d\boldsymbol{x}}\right|$

$$= p(f^{-1}(\boldsymbol{x})) \left|\det \frac{df^{-1}}{d\boldsymbol{x}}\right|$$

$\det \frac{df^{-1}}{d\boldsymbol{x}}$ -- Jacobian determinant

Transformation function f

- Invertible

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

inference: $\boldsymbol{z} = f^{-1}(\boldsymbol{x})$

density: $p(\boldsymbol{x}) = p(\boldsymbol{z}) \left|\det \frac{d\boldsymbol{z}}{d\boldsymbol{x}}\right|$

$$= p(f^{-1}(\boldsymbol{x})) \left|\det \frac{df^{-1}}{d\boldsymbol{x}}\right|$$

$\det \frac{df^{-1}}{d\boldsymbol{x}}$ -- Jacobian determinant

Transformation function f

- Invertible
- Jacobian determinant easy to compute
  e.g., choose $df^{-1}/d\boldsymbol{x}$ to be a triangular matrix

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z}_0 \sim p(\boldsymbol{z}_0)$$
$$\boldsymbol{x} = \boldsymbol{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\boldsymbol{z}_0)$$

inference: $\boldsymbol{z}_i = f_i^{-1}(\boldsymbol{z}_{i-1})$

density: $p(\boldsymbol{z}_i) = p(\boldsymbol{z}_{i-1}) \left|\det \frac{d\boldsymbol{z}_{i-1}}{d\boldsymbol{z}_i}\right|$

Transformation function f_i

- Invertible
- Jacobian determinant easy to compute

  e.g., choose $df_i^{-1}/d\boldsymbol{z}_i$ to be a triangular matrix

Normalizing Flow (NF)

- Transforms a simple distribution into a complex one by applying a sequence of transformation functions

$$\boldsymbol{z}_0 \sim p(\boldsymbol{z}_0)$$
$$\boldsymbol{x} = \boldsymbol{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\boldsymbol{z}_0)$$

inference: $\boldsymbol{z}_i = f_i^{-1}(\boldsymbol{z}_{i-1})$

density: $p(\boldsymbol{z}_i) = p(\boldsymbol{z}_{i-1}) \left| \det \frac{d\boldsymbol{z}_{i-1}}{d\boldsymbol{z}_i} \right|$

Transformation function f_i

- Invertible
- Jacobian determinant easy to compute

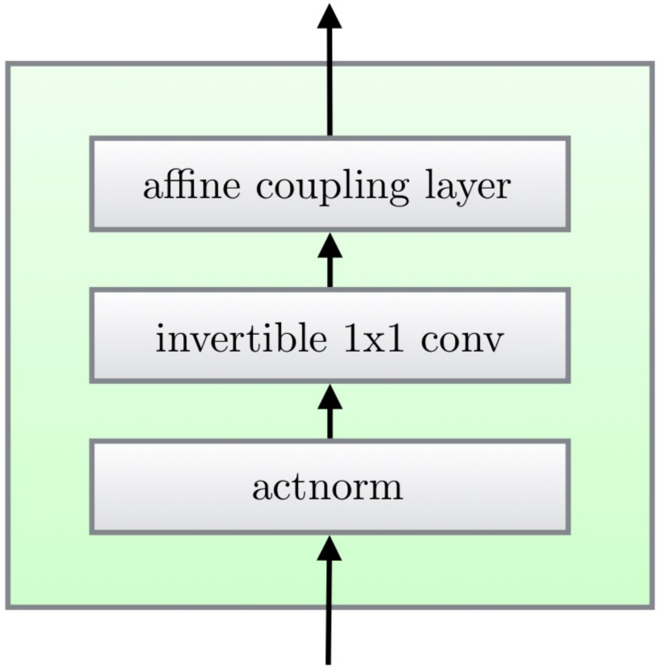
  e.g., choose $df_i^{-1}/d\boldsymbol{z}_i$ to be a triangular matrix

training: maximizes data log-likelihood

$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{z}_0) + \sum_{i=1}^{K} \log \left| \det \frac{d\boldsymbol{z}_{i-1}}{d\boldsymbol{z}_i} \right|$$

GLOW

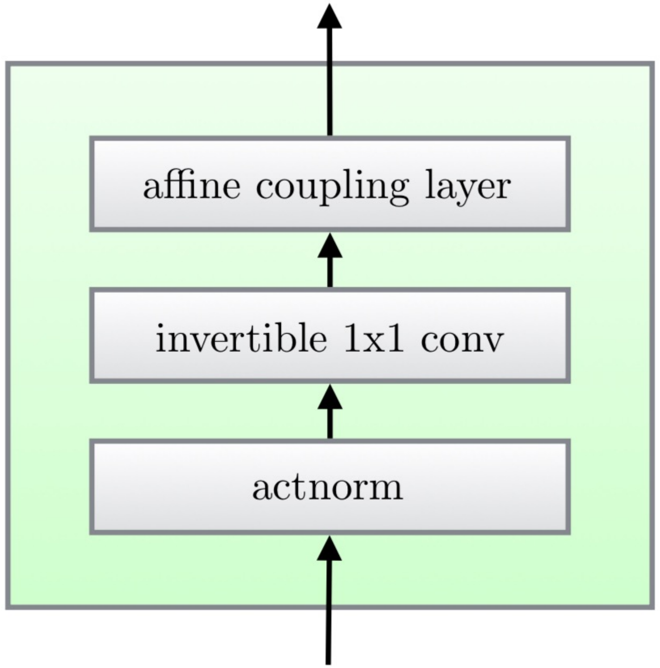
- [Kingma and Dhariwal., 2018]



One step of flow in the Glow model

GLOW

- [Kingma and Dhariwal., 2018]



One step of flow in the Glow model

Key Takeaways

- GANs:
  - Implicit generative model
  - Minimax formulation
  - non-saturating GANs
  - WGAN

- Normalizing Flow
  - Transforms a simple distribution into a complex one by applying a sequence of transformation functions

Questions?