DSC291: Machine Learning with Few Labels

Deep generative modeling
Generative adversarial learning

Zhiting Hu

Lecture 13, February 8, 2023

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Outline

- Generative adversarial networks (GANs)
- Normalizing Flow

Generative modeling

- In generative modeling, we'd like to train a network that models a distribution, such as a distribution over images.
- One way to judge the quality of the model is to sample from it.
- This field has seen rapid progress:

2009

2015

2018

Courtesy: Grosse CSC321 Lecture 19

Generative modeling

- In generative modeling, we’d like to train a network that models a distribution, such as a distribution over images.
- One way to judge the quality of the model is to sample from it.
- This field has seen rapid progress:

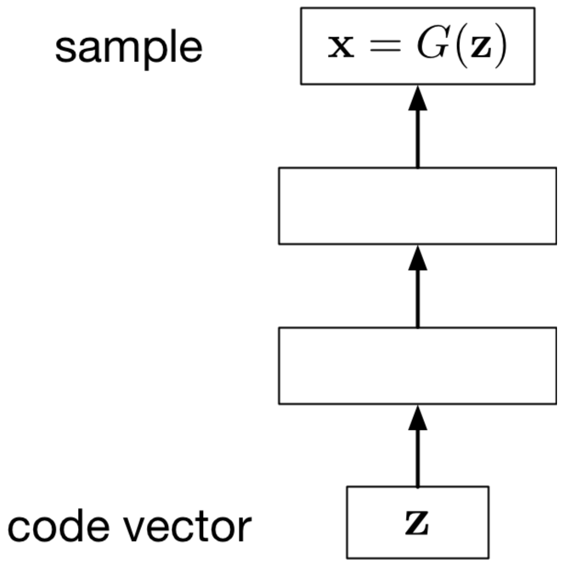
Figure courtesy: Ian Goodfellow

Generative modeling

- Modern approaches to generative modeling:
  - Variational Auto-encoder (Lecture #5)
  - Auto-regressive models (e.g., language model) (Lecture #6)
  - Generative adversarial networks (today)
  - Reversible architectures (today)
  - Diffusion models (later if time permits)

Implicit Generative Models

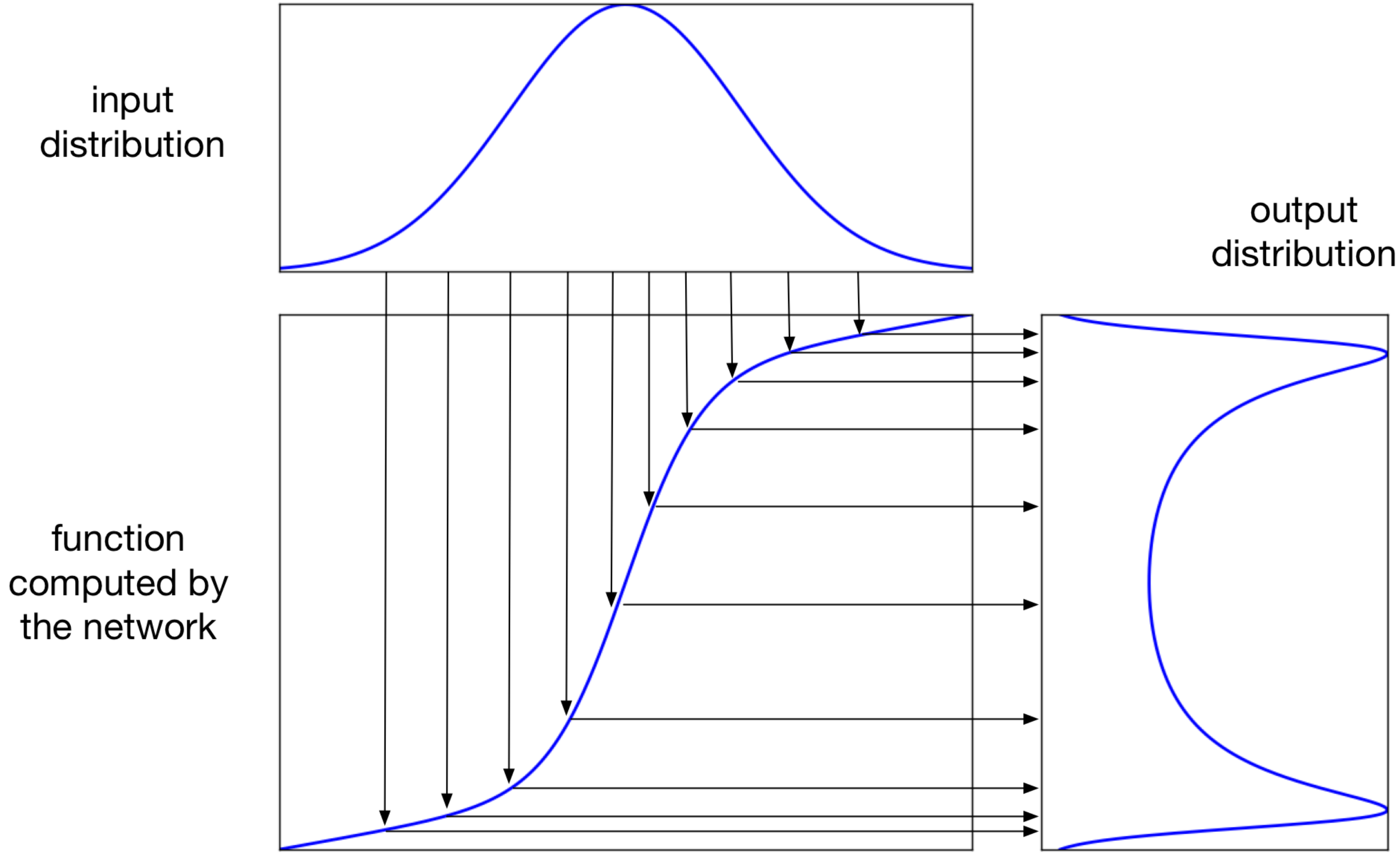
- Implicit generative models implicitly define a probability distribution
- Start by sampling the code vector $\mathbf{z}$ from a fixed, simple distribution (e.g. spherical Gaussian)
- The generator network computes a differentiable function G mapping $\mathbf{z}$ to an $\mathbf{x}$ in data space



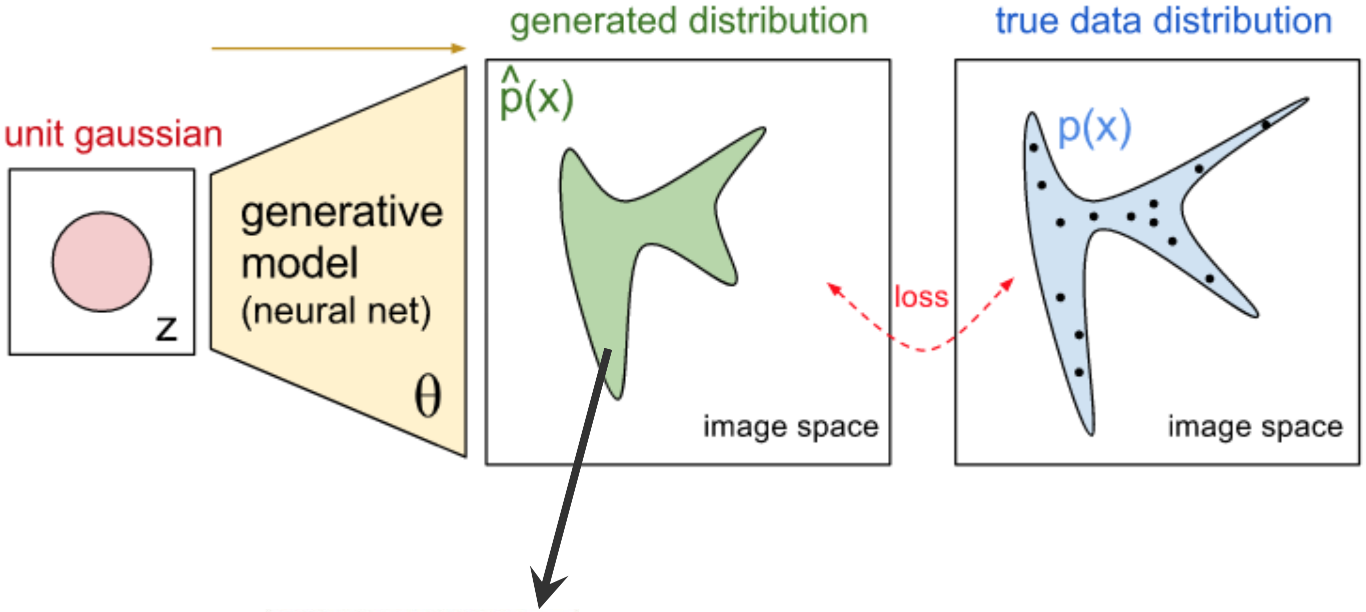
- a stochastic process to simulate data $\boldsymbol{x}$
- Intractable to evaluate likelihood

Implicit Generative Models

A 1-dimensional example:



Implicit Generative Models



https://blog.openai.com/generative-models/

Implicit Generative Models

- The advantage of implicit generative models: if you have some criterion for evaluating the quality of samples, then you can compute its gradient with respect to the network parameters, and update the network's parameters to make the sample a little better
- The idea behind **Generative Adversarial Networks (GANs)**: train two different networks
  - The generator network tries to produce realistic-looking samples
  - The discriminator network tries to figure out whether an image came from the training set or the generator network
- The generator network tries to fool the discriminator network

Generative Adversarial Nets (GANs)

- Generative model $\boldsymbol{x} = G_\theta(\boldsymbol{z}), \quad \boldsymbol{z} \sim p(\boldsymbol{z})$
  - Maps noise variable $\boldsymbol{z}$ to data space $\boldsymbol{x}$
  - Defines an implicit distribution over $\boldsymbol{x}$: $p_{g_\theta}(\boldsymbol{x})$
- Discriminator $D_\phi(\boldsymbol{x})$
  - Output the probability that $\boldsymbol{x}$ came from the data rather than the generator

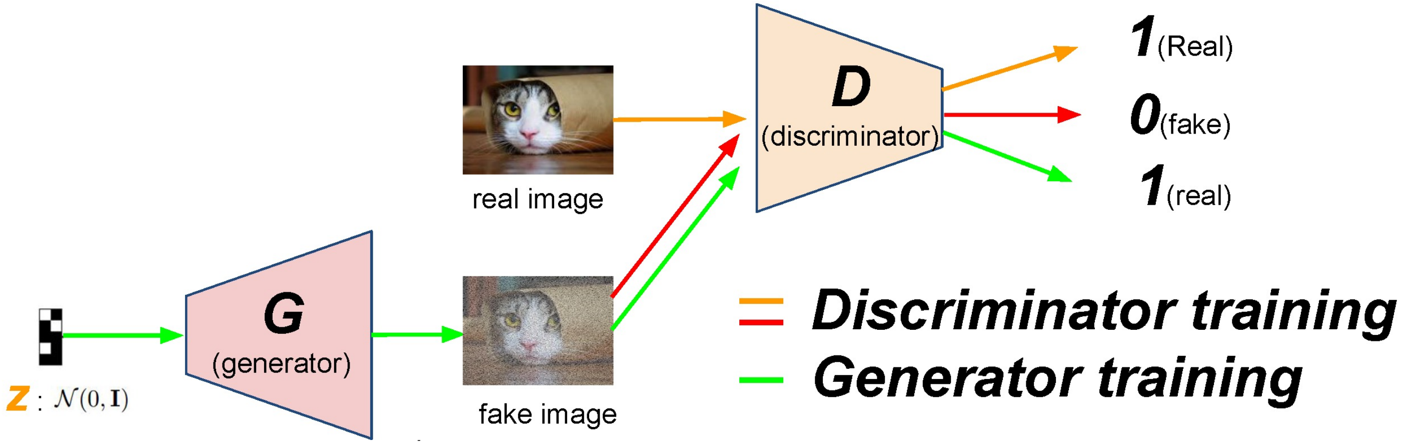


Figure courtesy: Kim

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

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

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

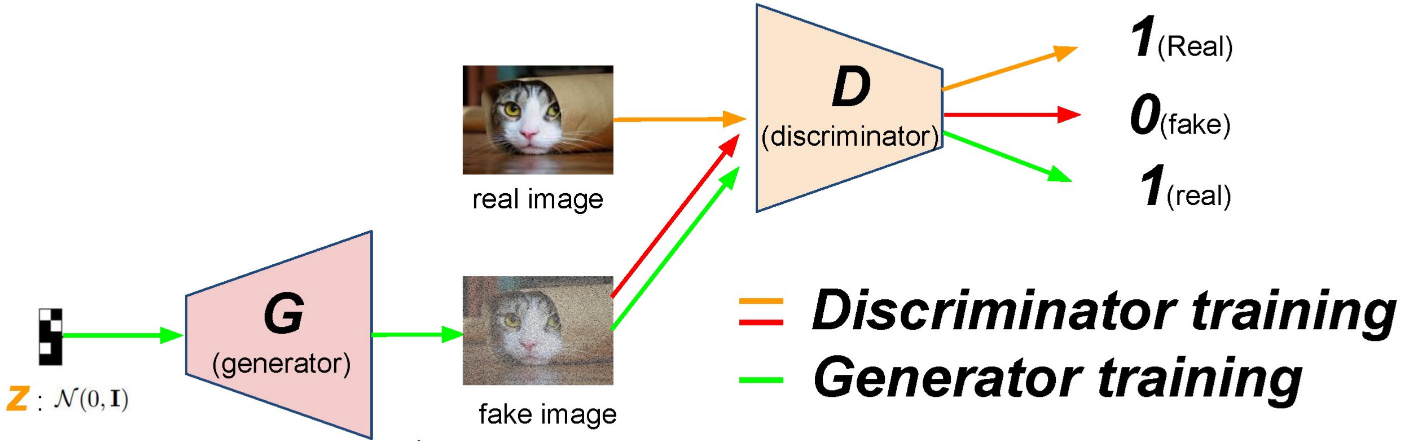
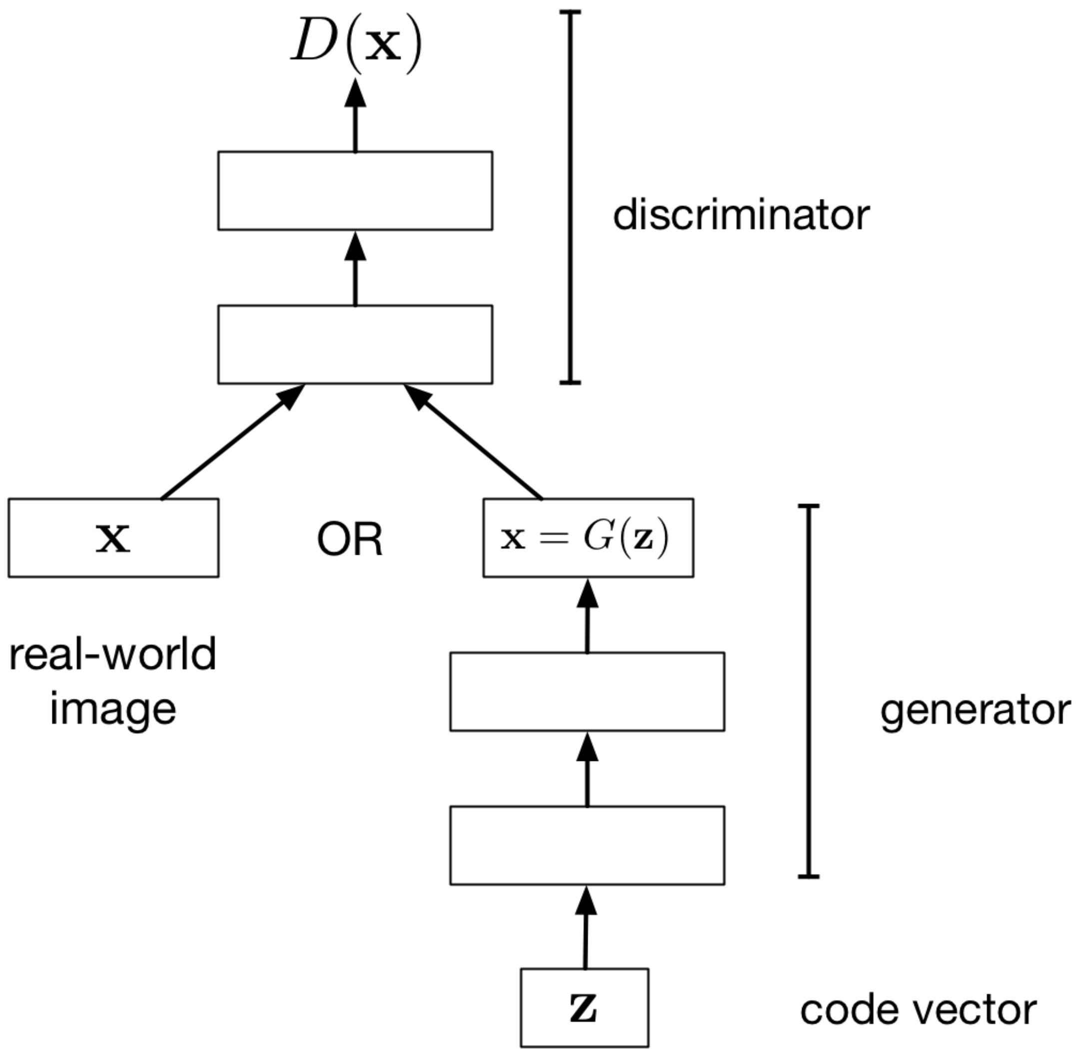


Figure courtesy: Kim

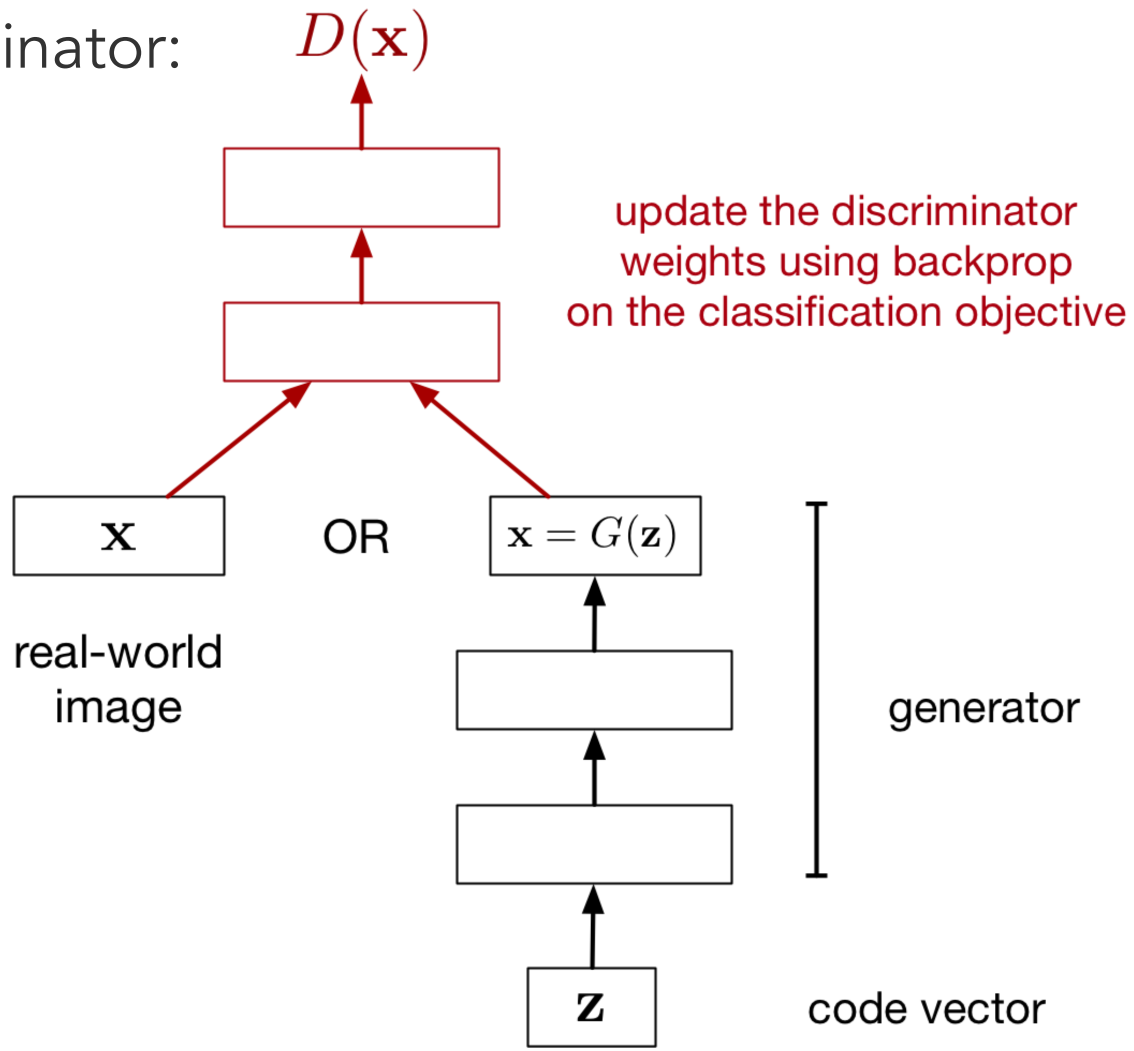
Generative Adversarial Nets (GANs)



Courtesy: Grosse CSC321 Lecture 19

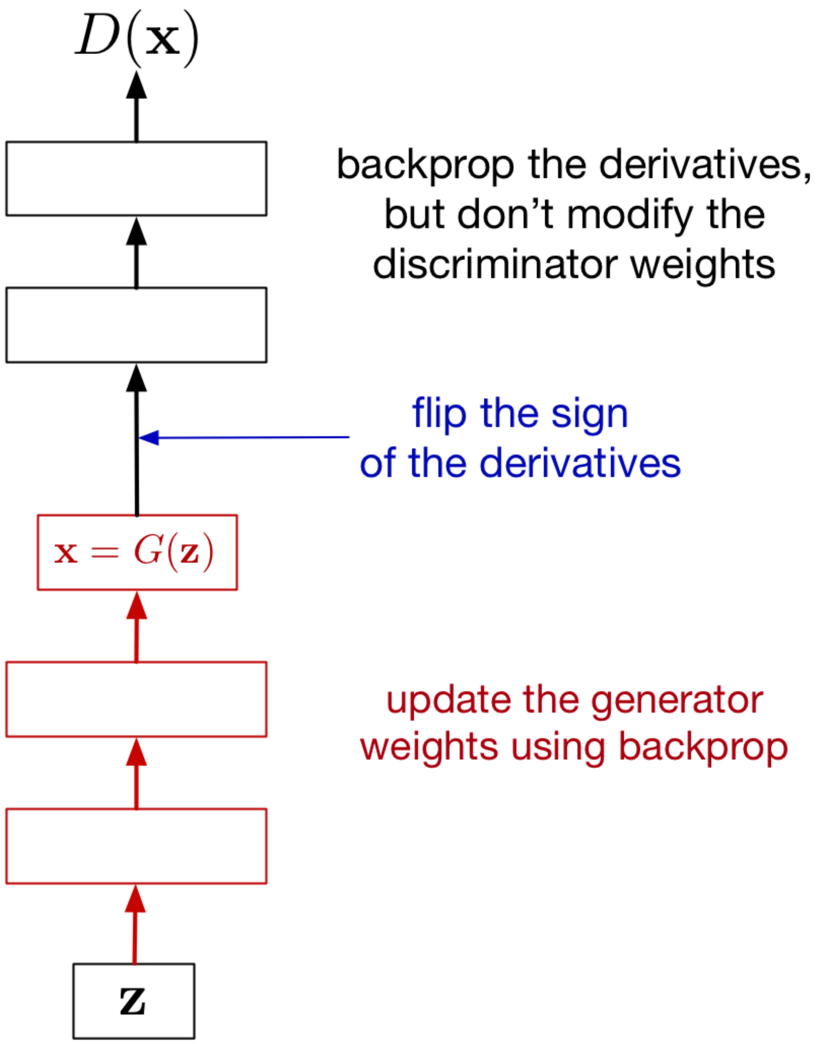
Generative Adversarial Nets (GANs)

Updating the discriminator:



Generative Adversarial Nets (GANs)

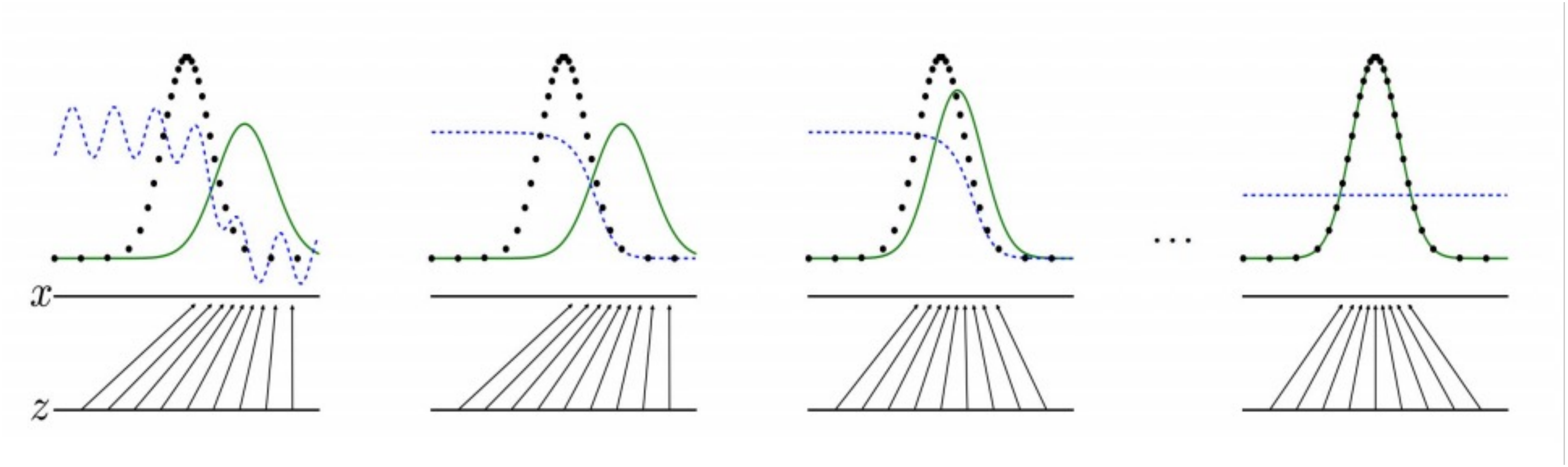
Updating the generator:



Courtesy: Grosse CSC321 Lecture 19

Generative Adversarial Nets (GANs)

Alternating training of the generator and discriminator:



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: $p_g = p_{data}$
- Proof:

Optimality of GANs

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

[Goodfellow et al., 2014]

Optimality of GANs

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D^*_G(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \rightarrow a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$.

[Goodfellow et al., 2014]

Optimality of GANs

- The minimax game can now be reformulated as

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

[Goodfellow et al., 2014]

Optimality of GANs

- The minimax game can now be reformulated as

$$
\begin{aligned}
C(G) =& \max_D V(G, D) \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D^*_G(G(\boldsymbol{z})))] \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))] \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

[Goodfellow et al., 2014]

Optimality of GANs

- The minimax game can now be reformulated as

$$
\begin{aligned}
C(G) =& \max_D V(G, D) \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

$$
\begin{aligned}
C(G) &= -\log(4) + KL\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + KL\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) \\
&= -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)
\end{aligned}
$$

Jensen-Shannon Divergence

[Goodfellow et al., 2014]

A better loss function

- We introduced the minimax cost function for the generator:

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

- One problem with this is saturation.
- Here, if the generated sample is really bad, the discriminator's prediction is close to 0, and the generator's cost is flat.

A better loss function: non-saturating GAN

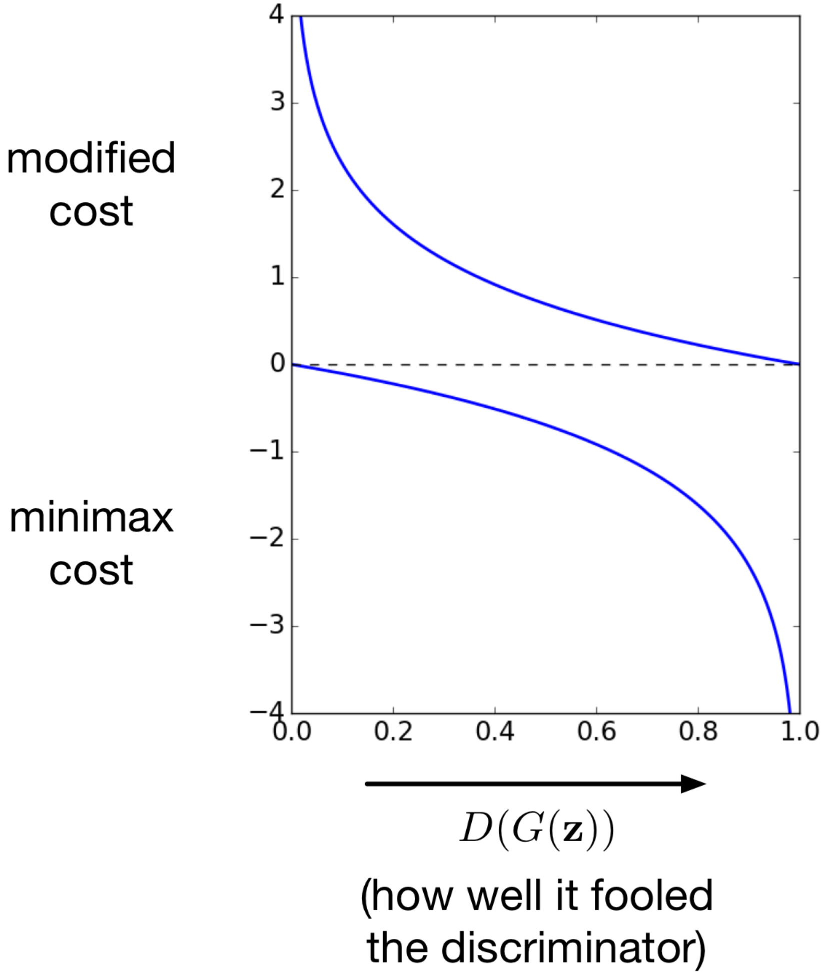
- Original minimax cost:

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

- 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

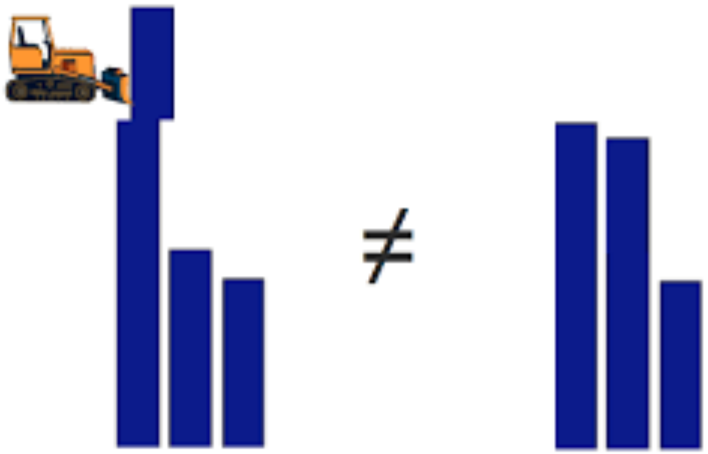
Wasserstein GAN (WGAN)

- 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

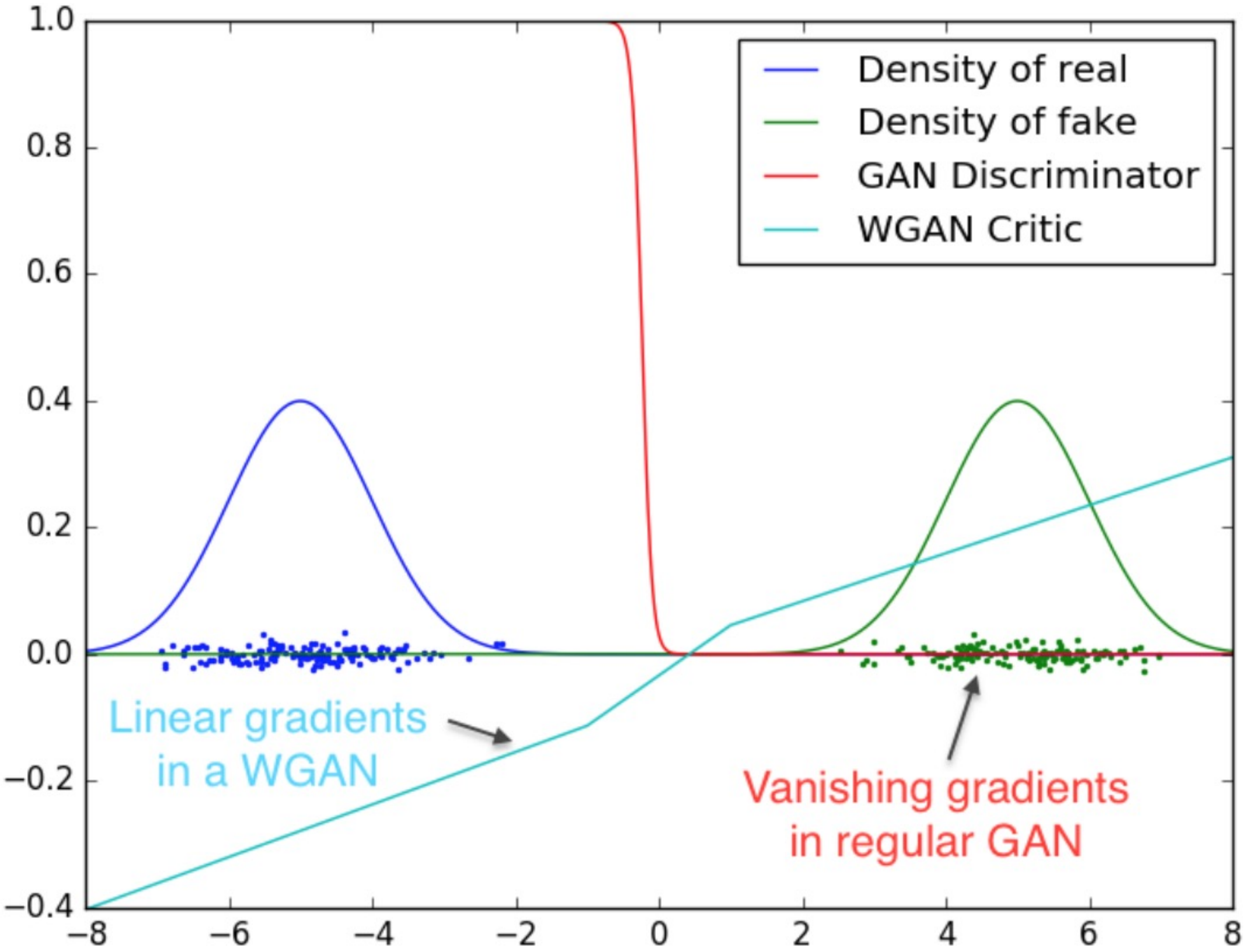
Wasserstein GAN (WGAN)

- Objective

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

- $||D||_L \leq K$: 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

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

- 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

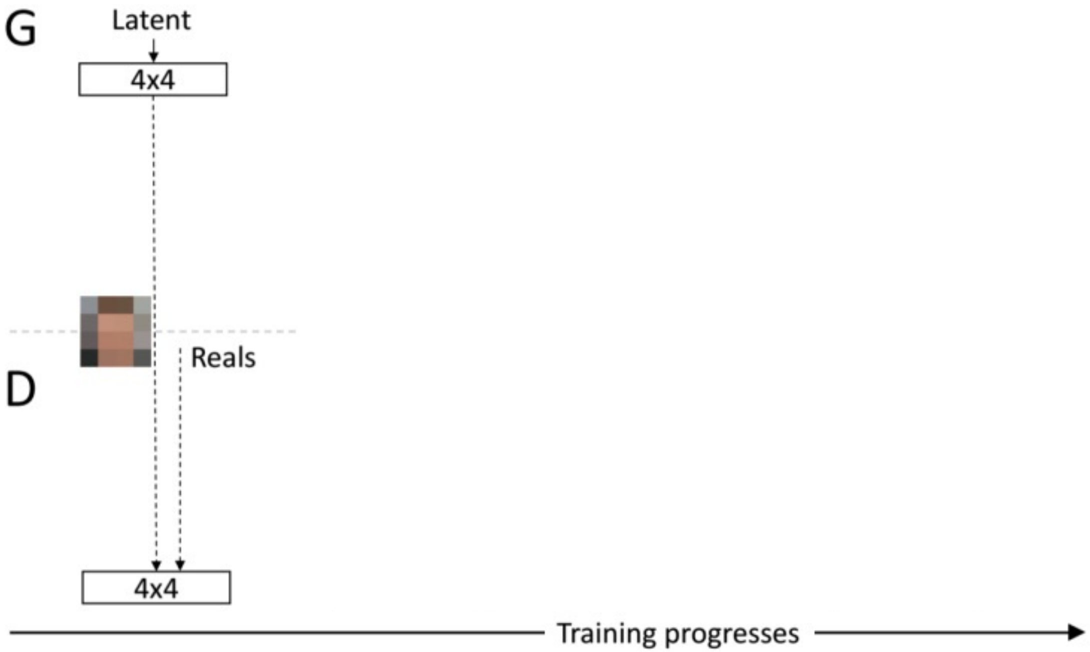
- Augment the standard objective to account for ϕ:

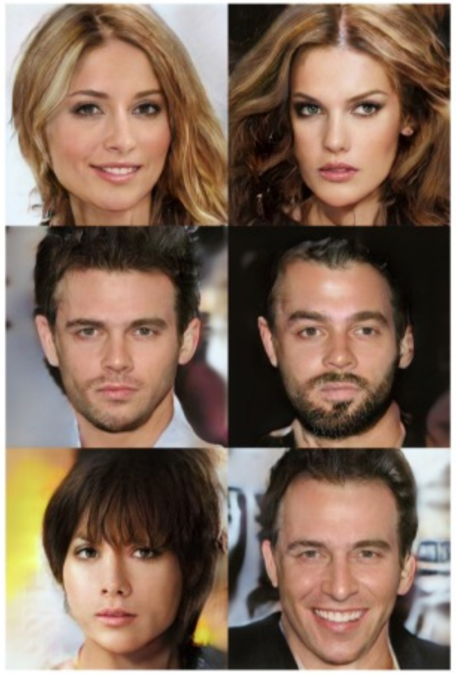
$$\min_{\theta} \max_{\phi} \min_{q} -\alpha \mathbb{H}(q) + \beta \mathbb{D}\Big(q(\boldsymbol{x}), p_\theta(\boldsymbol{x}) \Big) - \mathbb{E}_{q(\boldsymbol{x})}\Big[f_\phi(\boldsymbol{x}) \Big] + \mathbb{E}_{p_d(\boldsymbol{x})}\Big[f_\phi(\boldsymbol{x}) \Big]$$

- 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



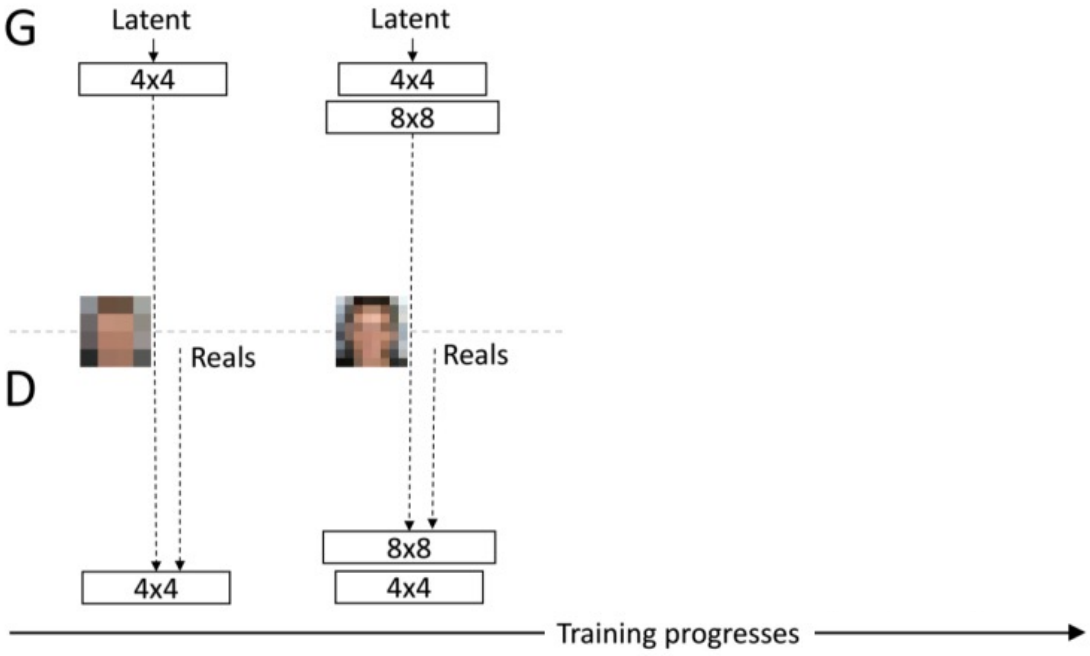


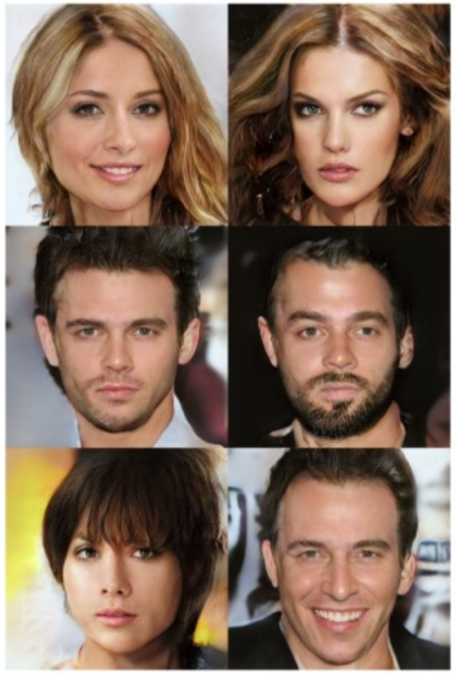
[Karras et al., 2018]

Progressive GAN

Low resolution images

add in
additional
layers





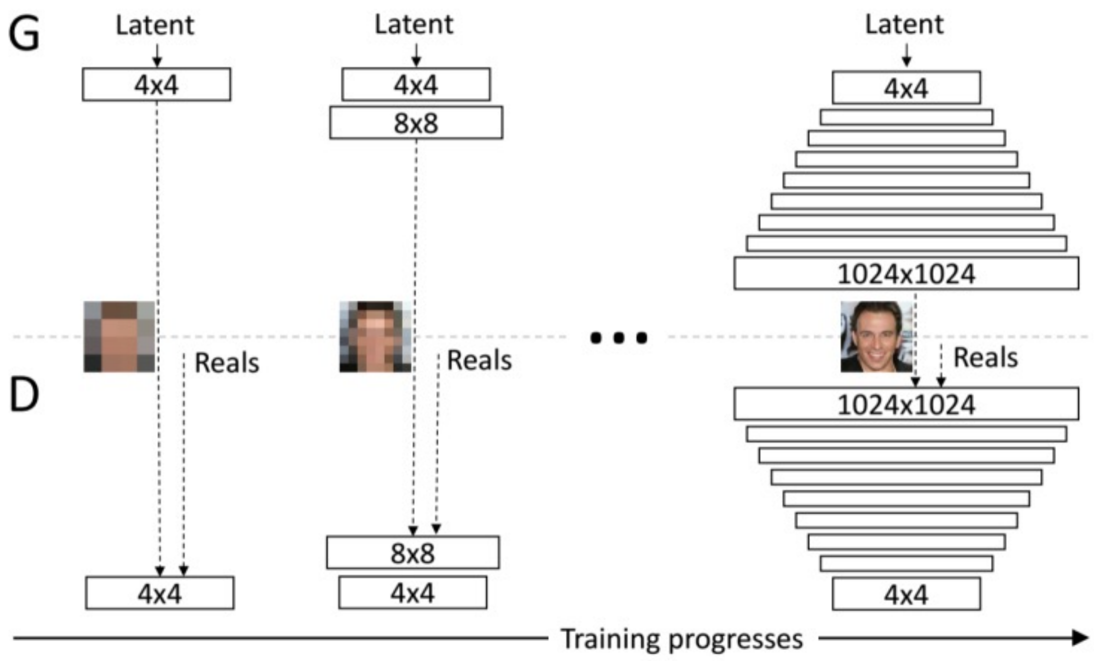
[Karras et al., 2018]

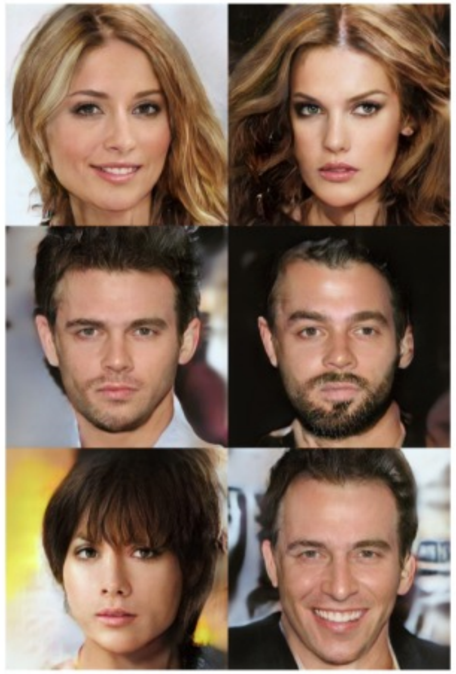
Progressive GAN

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]

Outline

- Generative Adversarial Networks (GANs)
  - Vanilla GAN, Wasserstein GAN, Progressive GAN, BigGAN

- Normalizing Flow (NF)
  - Basic Concepts
  - GLOW

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

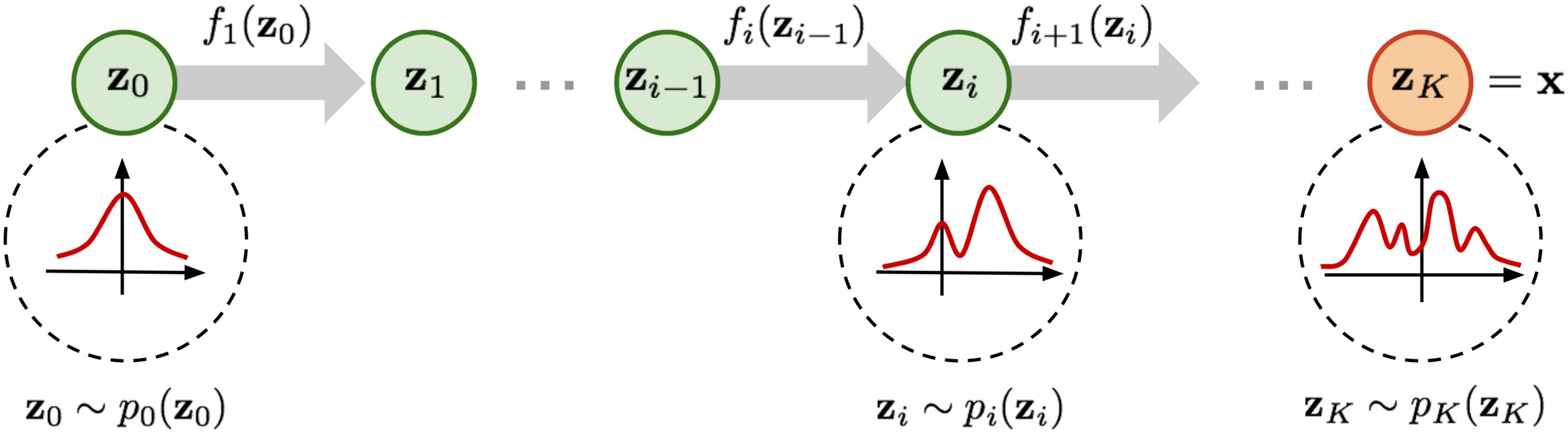
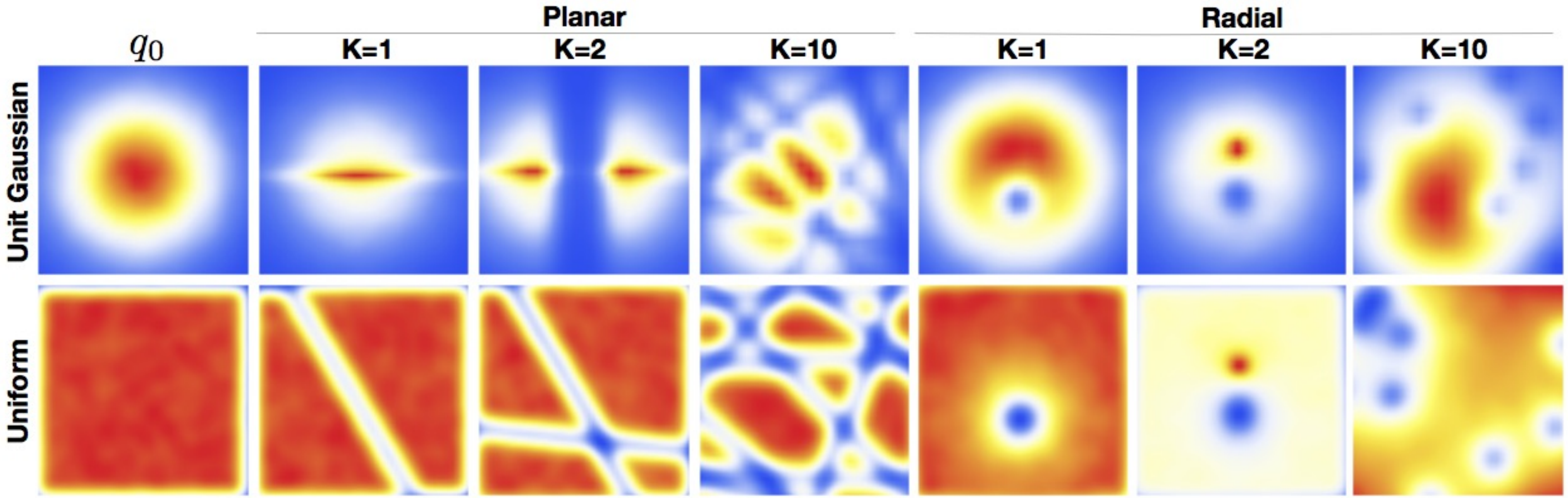


Figure courtesy: Lilian Weng

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

$$
\begin{aligned}
\boldsymbol{z} &\sim p(\boldsymbol{z}) \\
\boldsymbol{x} &= f(\boldsymbol{z})
\end{aligned}
$$

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

Normalizing Flow (NF)

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

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

Transformation function f

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

- Invertible

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

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

$$z_0 \sim p(z_0)$$
$$x = z_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0)$$

inference: $z_i = f_i^{-1}(z_{i-1})$

density: $p(z_i) = p(z_{i-1}) \left| \det \frac{dz_{i-1}}{dz_i} \right|$

Transformation function f_i

- Invertible
- Jacobian determinant easy to compute

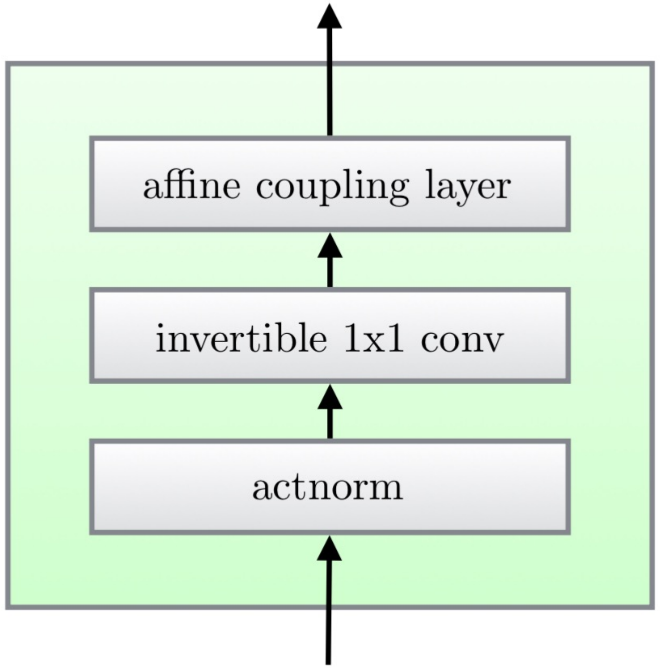
  e.g., choose df_i^{-1}/dz_i to be a triangular matrix

training: maximizes data log-likelihood

$$\log p(x) = \log p(z_0) + \sum_{i=1}^{K} \log \left| \det \frac{dz_{i-1}}{dz_i} \right|$$

GLOW

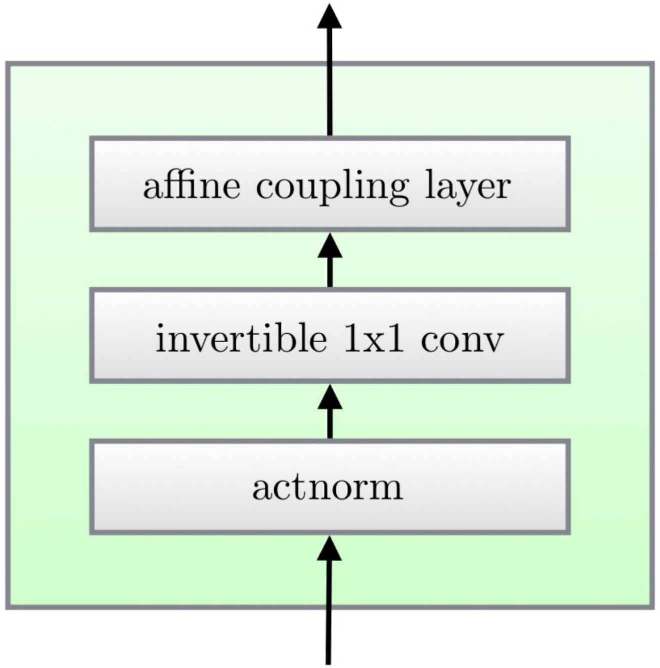
- [Kingma and Dhariwal., 2018]



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

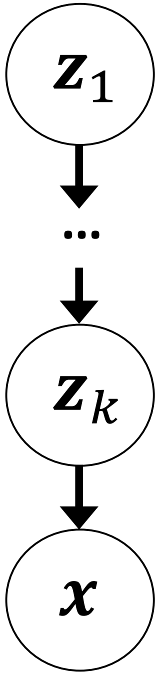
- Normalizing Flow
  - Transforms a simple distribution into a complex one by applying a sequence of transformation functions

Questions?

Backups

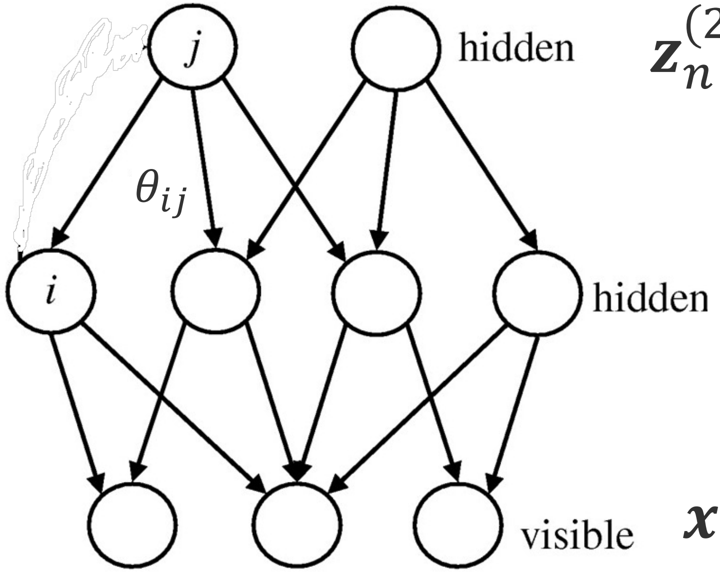
Deep generative models

- 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]



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

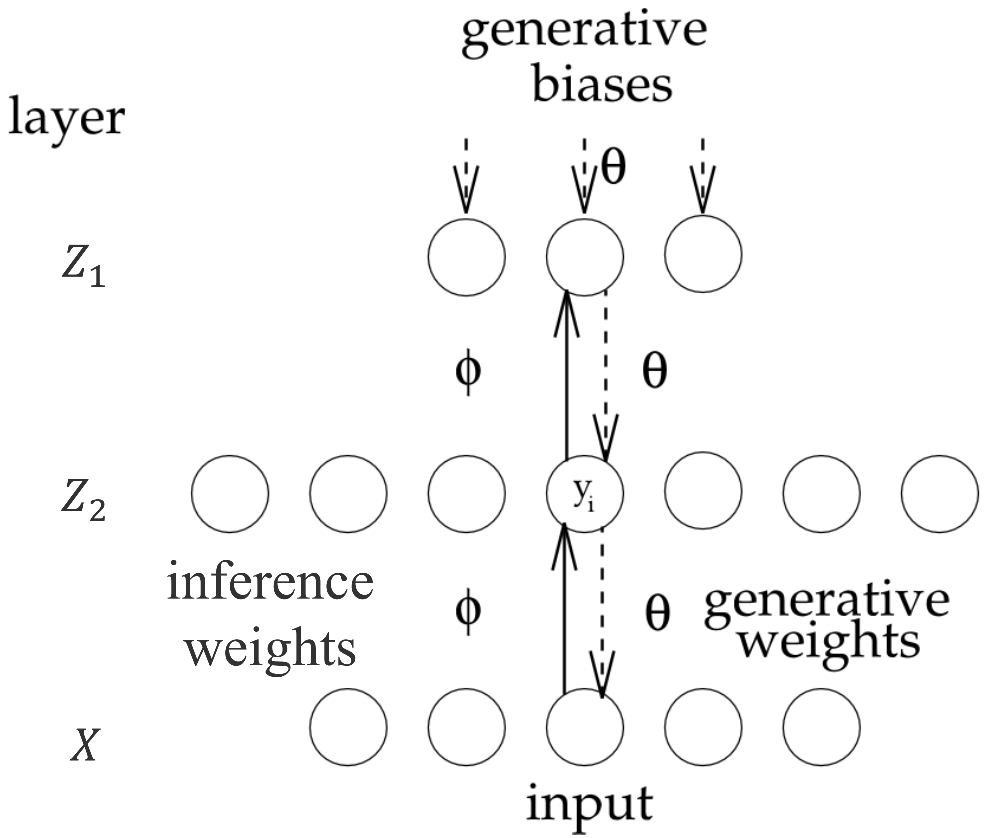
$$\boldsymbol{z}_n^{(1)} = \{0,1\}^I$$

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

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

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]

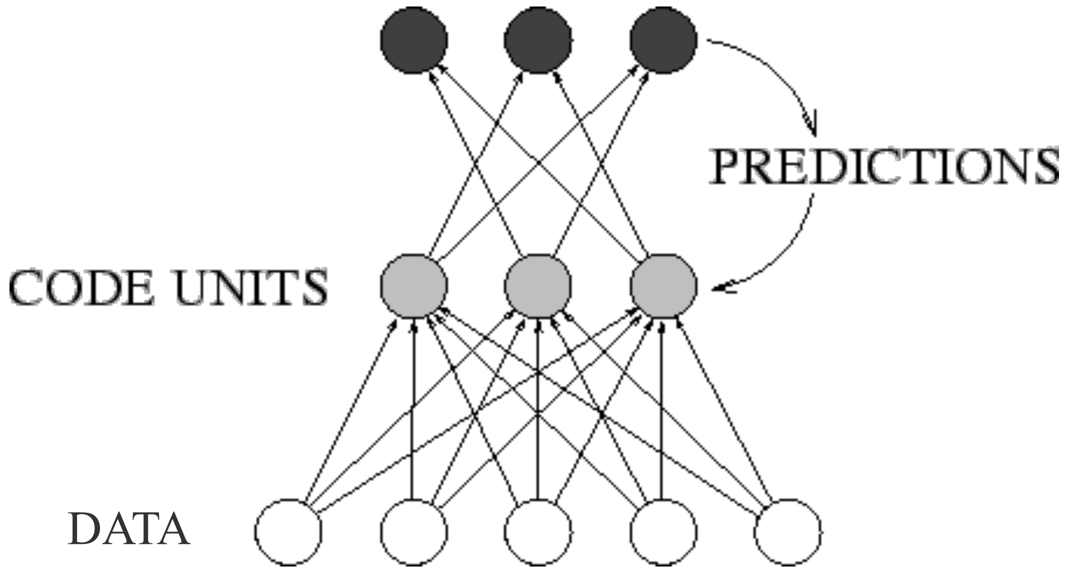


Figure courtesy: Schmidhuber 1996

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

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

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

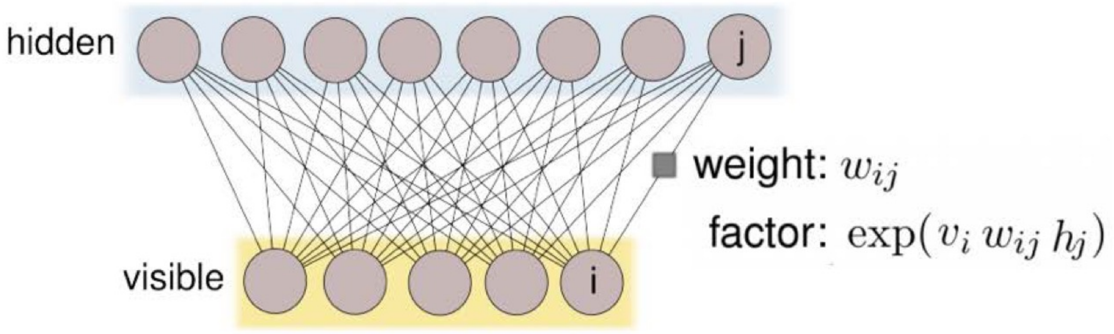
  - Wake sleep

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

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

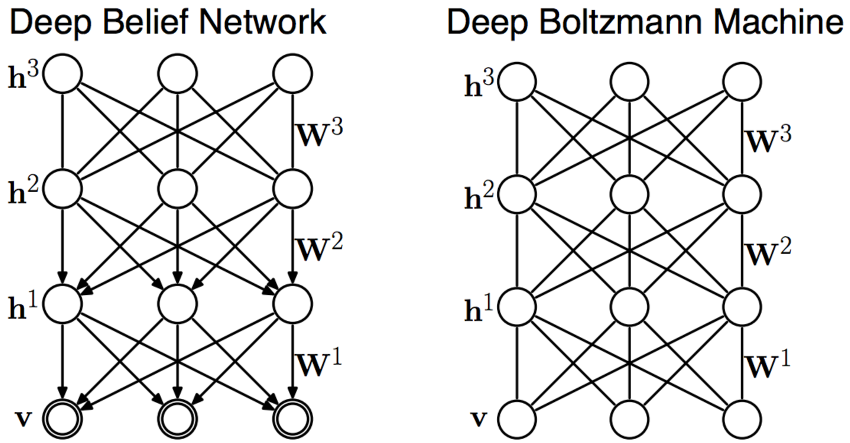
Resurgence of deep generative models

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



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]

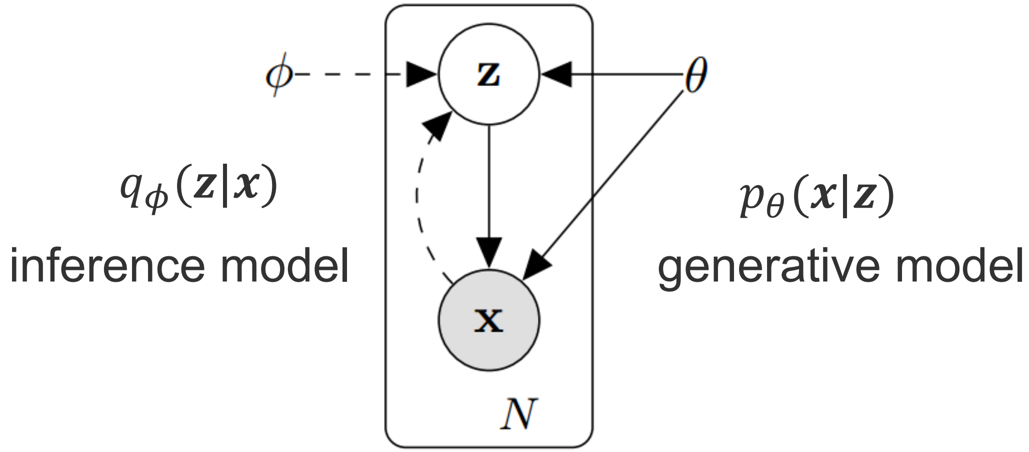
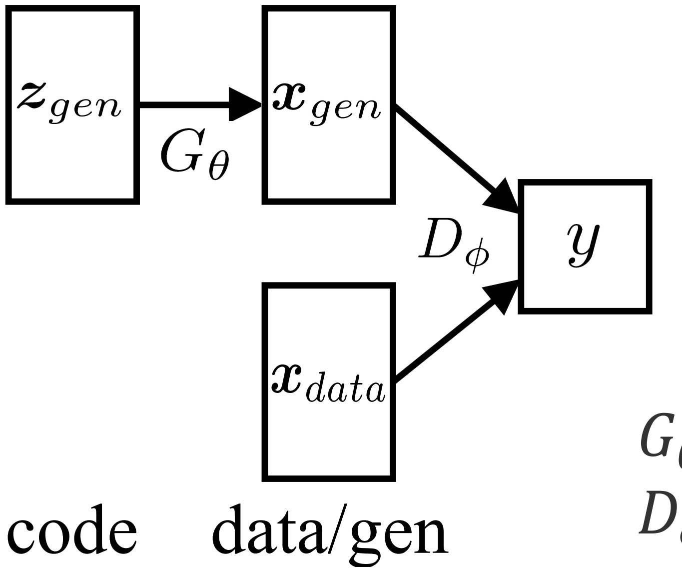


Figure courtesy: Kingma & Welling, 2014

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