

Deep Generative Models

Lecture supplementary

Roman Isachenko



2024, Spring

Outline I

1. Autoregressive models

- Masked Autoencoder (MADE)
- WaveNet
- GatedPixelCNN

2. Variational inference

- ELBO gradient, Log derivative trick
- Mean field approximation

3. VAE-related topics

- Posterior collapse and decoder weakening techniques
- IWAE
- PixelVAE, Hierarchical VAE

4. Normalizing Flows

- Flows intuition
- Residual Flows (planar flows)
- Autoregressive flows
- Inverse gaussian autoregressive flows

Outline II

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

VAE limitations: prior distribution

VAE limitations: posterior distribution

6. Disentanglement

InfoGAN

β -VAE

DIP-VAE

FactorVAE

Challenging Disentanglement Assumptions

7. GANs

DCGAN

Vanishing gradients

Outline III

Improved techniques for training GANs

Adversarial variational Bayes

WGAN

Spectral Normalization GAN

Evolution of GANs

Evaluation of likelihood-free models

8. FFJORD

9. Quantized latents

Vector Quantized VAE-2

Feature Quantized GAN

10. Score matching

Implicit score matching

Outline

1. Autoregressive models

Masked Autoencoder (MADE)
WaveNet
GatedPixelCNN

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

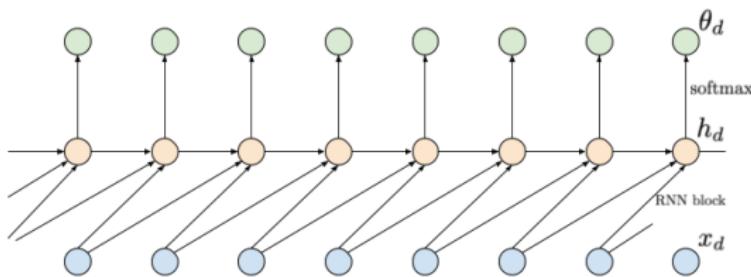
9. Quantized latents

10. Score matching

Autoregressive models

- ▶ Previous model has **limited** memory d . It is insufficient for many modalities (e.g. for images and text).
- ▶ Recurrent NN fixes this problem and potentially could learn long-range dependencies:

$$p(x_j | \mathbf{x}_{1:j-1}, \theta) = p(x_j | \mathbf{h}_j, \theta), \quad \mathbf{h}_j = \text{RNN}(\mathbf{x}_{j-d:j-1}, \mathbf{h}_{j-1})$$

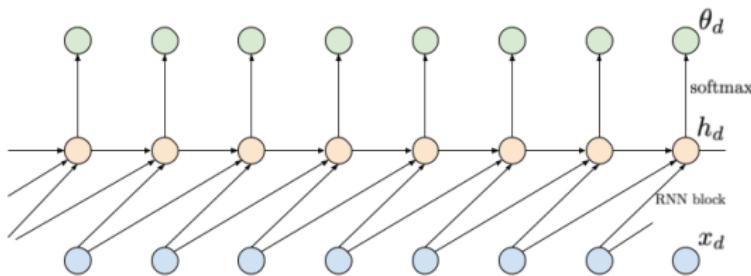


- ▶ Sequential computation of all conditionals $p(x_j | \mathbf{x}_{1:j-1}, \theta)$, hence, the training is slow.
- ▶ RNN suffers from vanishing and exploding gradients.

Autoregressive models: RNN

- ▶ Previous model has **limited** memory d . It is insufficient for many modalities (e.g. for images and text).
- ▶ Recurrent NN fixes this problem and potentially could learn long-range dependencies:

$$p(x_j | \mathbf{x}_{1:j-1}, \theta) = p(x_j | \mathbf{h}_j, \theta), \quad \mathbf{h}_j = \text{RNN}(\mathbf{x}_{j-d:j-1}, \mathbf{h}_{j-1})$$



- ▶ Sequential computation of all conditionals $p(x_j | \mathbf{x}_{1:j-1}, \theta)$, hence, the training is slow.
- ▶ RNN suffers from vanishing and exploding gradients.

Outline

1. Autoregressive models

Masked Autoencoder (MADE)

WaveNet

GatedPixelCNN

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

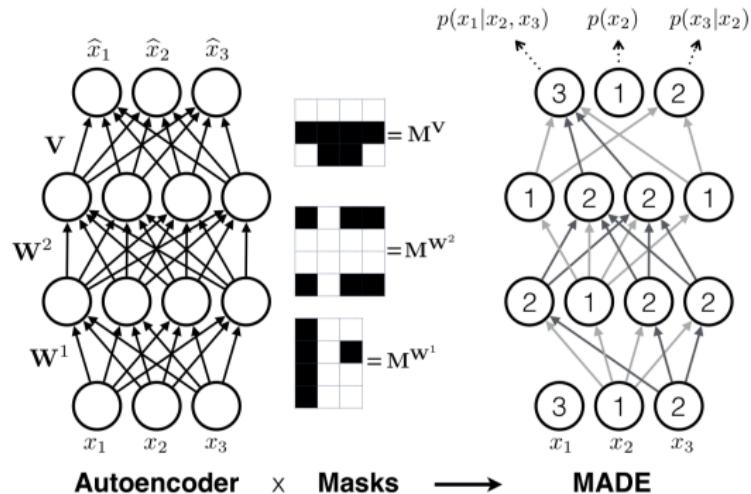
8. FFJORD

9. Quantized latents

10. Score matching

MADE

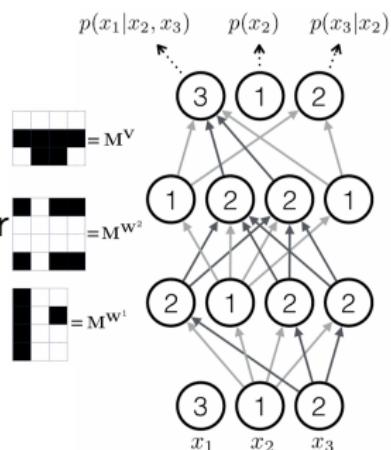
- ▶ Vanila autoencoder is not a generative model.
- ▶ Let mask the weight matrices to make the model generative:
 $\mathbf{W}_M = \mathbf{W} \cdot \mathbf{M}$.



- ▶ The question is how to create matrices \mathbf{M} which produce the autoregressive property?

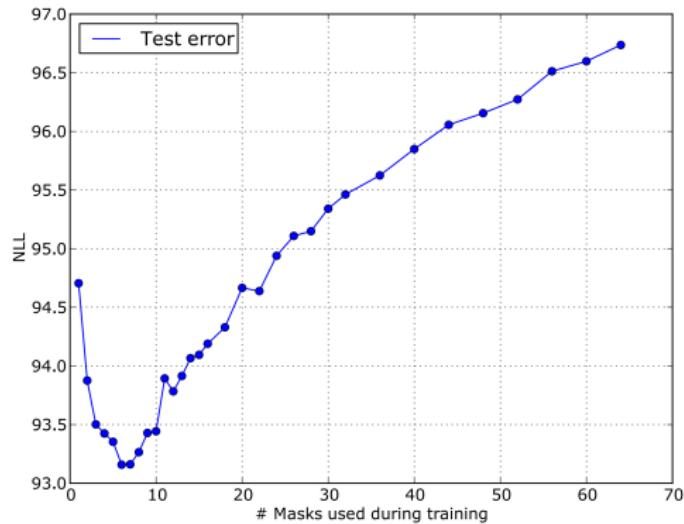
Masks generation

- ▶ Define the ordering of input elements from 1 to m .
- ▶ Assign the random number k from 1 to $m - 1$ to each hidden unit. The number gives the maximum value of input units to which the unit can be connected.
- ▶ Connect each hidden unit with number k with the previous layer units which has the number is **less or equal** than k .
- ▶ Connect each output unit with number k with the previous layer units which has the number is **less** than k .



Possible variations

- ▶ Order agnostic training (missing values in partially observed input vectors can be imputed efficiently);
- ▶ Connectivity-agnostic training (cheap ensembling).



Outline

1. Autoregressive models

Masked Autoencoder (MADE)

WaveNet

GatedPixelCNN

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

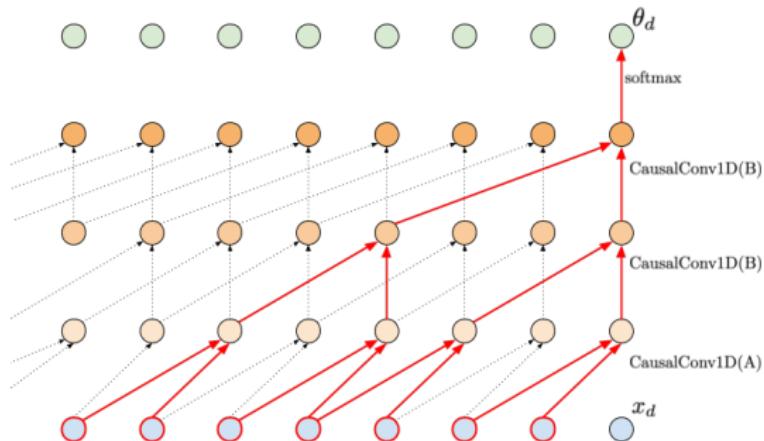
8. FFJORD

9. Quantized latents

10. Score matching

Autoregressive models

- ▶ Convolutions could be used for autoregressive models, but they have to be **causal**.
- ▶ Try to find and understand the difference between Conv A/B.



- ▶ Could learn long-range dependencies.
- ▶ Do not suffer from gradient issues.
- ▶ Easy to estimate probability for given input, but hard generation of new samples (the sequential process).

WaveNet

Goal

Efficient generation of raw audio waveforms with natural sounds.



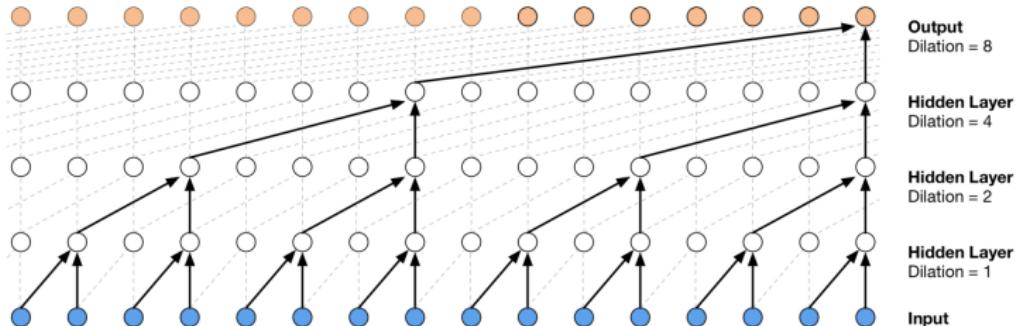
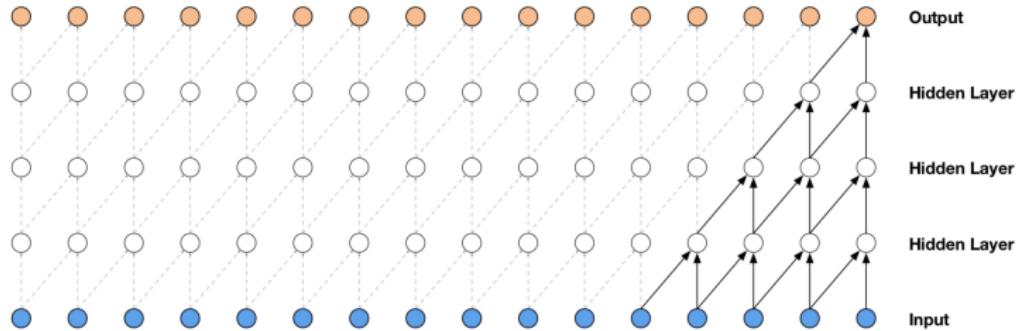
Solution

Autoregressive model

$$p(\mathbf{x}|\theta) = \prod_{t=1}^T p(x_t|\mathbf{x}_{1:t-1}, \theta).$$

- ▶ Each conditional $p(x_t|\mathbf{x}_{1:t-1}, \theta)$ models the distribution for the timestamp t .
- ▶ The model uses **causal** dilated convolutions.

WaveNet



Outline

1. Autoregressive models

Masked Autoencoder (MADE)
WaveNet
GatedPixelCNN

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

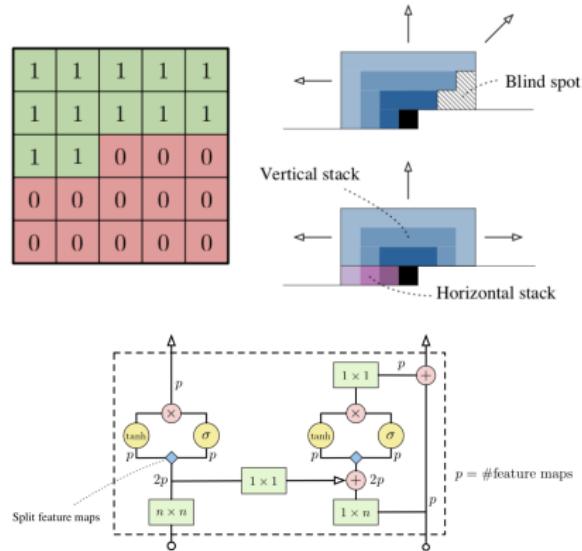
7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

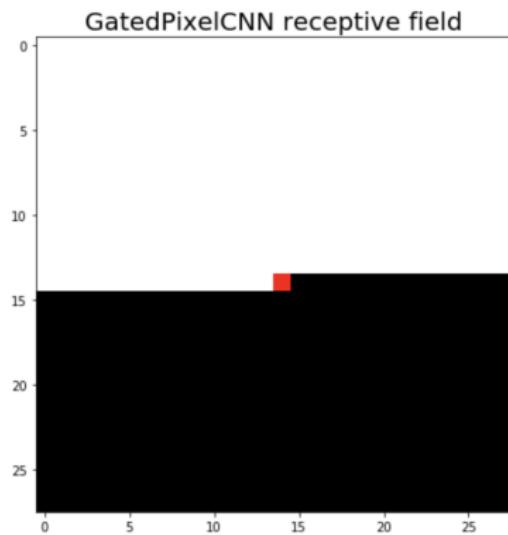
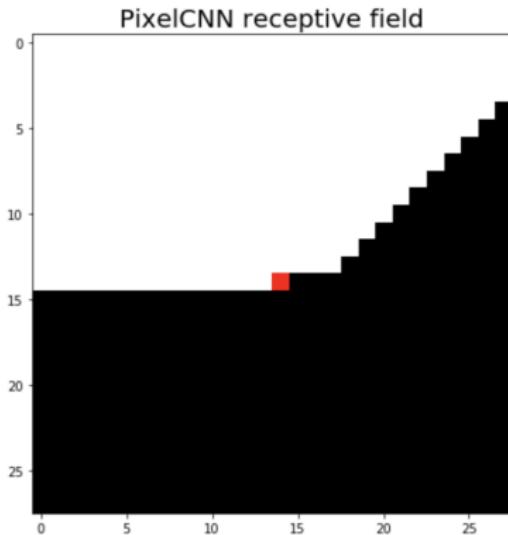
GatedPixelCNN (2016)



Van den Oord A. et al. Conditional image generation with pixelcnn decoders

<https://arxiv.org/pdf/1606.05328.pdf>

GatedPixelCNN (2016)



Van den Oord A. et al. Conditional image generation with pixelcnn decoders
<https://arxiv.org/pdf/1606.05328.pdf>

Extensions

- ▶ **PixelCNN++:** *Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications*
<https://arxiv.org/pdf/1701.05517.pdf>
(mixture of logistics instead of softmax);
- ▶ **PixelSNAIL:** *An Improved Autoregressive Generative Model*
<https://arxiv.org/pdf/1712.09763.pdf>
(self-attention to learn optimal autoregression ordering).

Outline

1. Autoregressive models

2. Variational inference

ELBO gradient, Log derivative trick

Mean field approximation

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

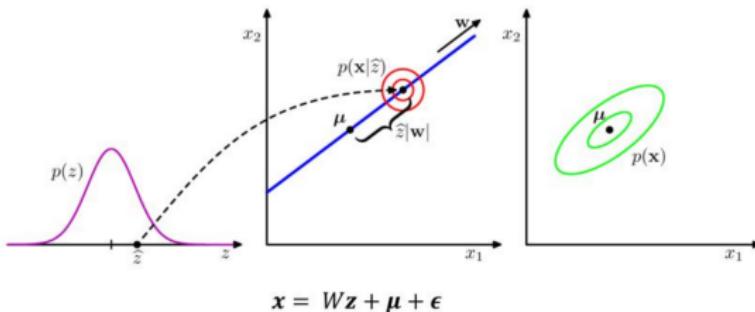
9. Quantized latents

10. Score matching

Latent variable models (LVM)

$$\log p(\mathbf{x}|\theta) = \log \int p(\mathbf{x}|\mathbf{z}, \theta)p(\mathbf{z})d\mathbf{z} \rightarrow \max_{\theta}$$

PCA projects original data \mathbf{X} onto a low dimensional latent space while maximizing the variance of the projected data.



- ▶ $p(\mathbf{x}|\mathbf{z}, \theta) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I})$
- ▶ $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$
- ▶ $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I})$
- ▶ $p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu}), \sigma^2 \mathbf{M})$, where $\mathbf{M} = \mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}$

Outline

1. Autoregressive models

2. Variational inference

ELBO gradient, Log derivative trick

Mean field approximation

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

ELBO gradient (E-step, $\nabla_{\phi}\mathcal{L}(\phi, \theta)$)

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_q \log p(\mathbf{X}|\mathbf{Z}, \theta) - KL(q(\mathbf{Z}|\mathbf{X}, \phi) || p(\mathbf{Z})) \rightarrow \max_{\phi, \theta} .$$

Difference from M-step: density function $q(\mathbf{z}|\mathbf{x}, \phi)$ depends on the parameters ϕ , it is impossible to use Monte-Carlo estimation:

$$\nabla_{\phi}\mathcal{L}(\phi, \theta) = \int \nabla_{\phi} q(\mathbf{Z}|\mathbf{X}, \phi) \log p(\mathbf{X}|\mathbf{Z}, \theta) d\mathbf{Z} - \nabla_{\phi} KL$$

Log-derivative trick

$$\nabla_{\xi} q(\eta|\xi) = q(\eta|\xi) \left(\frac{\nabla_{\xi} q(\eta|\xi)}{q(\eta|\xi)} \right) = q(\eta|\xi) \nabla_{\xi} \log q(\eta|\xi).$$

$$\nabla_{\phi} q(\mathbf{Z}|\mathbf{X}, \phi) = q(\mathbf{Z}|\mathbf{X}, \phi) \nabla_{\phi} \log q(\mathbf{Z}|\mathbf{X}, \phi).$$

ELBO gradient (E-step, $\nabla_{\phi}\mathcal{L}(\phi, \theta)$)

$$\begin{aligned}\nabla_{\phi}\mathcal{L}(\phi, \theta) &= \int \nabla_{\phi}q(\mathbf{Z}|\mathbf{X}, \phi) \log p(\mathbf{X}|\mathbf{Z}, \theta) d\mathbf{Z} - \nabla_{\phi}KL = \\ &= \int q(\mathbf{Z}|\mathbf{X}, \phi) [\nabla_{\phi} \log q(\mathbf{Z}|\mathbf{X}, \phi) \log p(\mathbf{X}|\mathbf{Z}, \theta)] d\mathbf{Z} - \nabla_{\phi}KL\end{aligned}$$

After applying log-reparametrization trick, we are able to use Monte-Carlo estimation:

$$\begin{aligned}\nabla_{\phi}\mathcal{L}(\phi, \theta) &\approx n \nabla_{\phi} \log q(\mathbf{z}_i^* | \mathbf{x}_i, \phi) \log p(\mathbf{x}_i | \mathbf{z}_i^*, \theta) - \nabla_{\phi}KL, \\ \mathbf{z}_i^* &\sim q(\mathbf{z}_i | \mathbf{x}_i, \phi).\end{aligned}$$

Problem

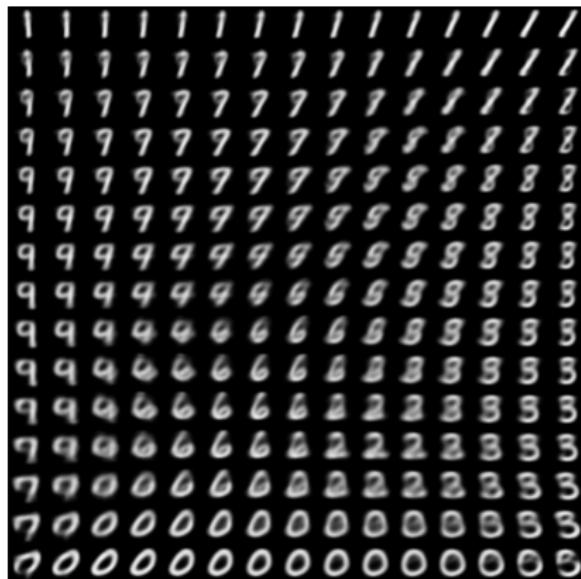
Unstable solution with huge variance.

Solution

Reparametrization trick

Variational Autoencoder

Generated images for latent objects \mathbf{z} sampled from prior $\mathcal{N}(0, \mathbf{I})$



Outline

1. Autoregressive models

2. Variational inference

ELBO gradient, Log derivative trick

Mean field approximation

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

VAE as Bayesian model

Posterior distribution

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})}$$

ELBO

$$\begin{aligned}\log p(\theta|\mathbf{X}) &= \log p(\mathbf{X}|\theta) + \log p(\theta) - \log p(\mathbf{X}) \\ &= \mathcal{L}(q, \theta) + KL(q||p) + \log p(\theta) - \log p(\mathbf{X}) \\ &\geq [\mathcal{L}(q, \theta) + \log p(\theta)] - \log p(\mathbf{X}).\end{aligned}$$

EM-algorithm

► E-step

$$q(\mathbf{z}) = \arg \max_q \mathcal{L}(q, \theta^*) = \arg \min_q KL(q||p) = p(\mathbf{z}|\mathbf{x}, \theta^*);$$

► M-step

$$\theta^* = \arg \max_{\theta} [\mathcal{L}(q, \theta) + \log p(\theta)].$$

Bayesian framework

Bayes theorem

$$p(\mathbf{t}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{t})p(\mathbf{t})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mathbf{t})p(\mathbf{t})}{\int p(\mathbf{x}|\mathbf{t})p(\mathbf{t})d\mathbf{t}}$$

- ▶ \mathbf{x} – observed variables, \mathbf{t} – unobserved variables (latent variables/parameters);
- ▶ $p(\mathbf{x}|\mathbf{t})$ – likelihood;
- ▶ $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{t})p(\mathbf{t})d\mathbf{t}$ – evidence;
- ▶ $p(\mathbf{t})$ – prior distribution, $p(\mathbf{t}|\mathbf{x})$ – posterior distribution.

Meaning

We have unobserved variables \mathbf{t} and some prior knowledge about them $p(\mathbf{t})$. Then, the data \mathbf{x} has been observed. Posterior distribution $p(\mathbf{t}|\mathbf{x})$ summarizes the knowledge after the observations.

Variational Lower Bound

We have set of objects $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$. The goal is to perform Bayesian inference on the unobserved variables $\mathbf{T} = \{\mathbf{t}_i\}_{i=1}^n$.

Evidence Lower Bound (ELBO)

$$\begin{aligned}\log p(\mathbf{X}) &= \log \frac{p(\mathbf{X}, \mathbf{T})}{p(\mathbf{T}|\mathbf{X})} = \\&= \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{p(\mathbf{T}|\mathbf{X})} d\mathbf{T} = \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})q(\mathbf{T})}{p(\mathbf{T}|\mathbf{X})q(\mathbf{T})} d\mathbf{T} = \\&= \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{q(\mathbf{T})} d\mathbf{T} + \int q(\mathbf{T}) \log \frac{q(\mathbf{T})}{p(\mathbf{T}|\mathbf{X})} d\mathbf{T} = \\&= \mathcal{L}(q) + KL(q(\mathbf{T})||p(\mathbf{T}|\mathbf{X})) \geq \mathcal{L}(q).\end{aligned}$$

We would like to maximize lower bound $\mathcal{L}(q)$.

Mean field approximation

Independence assumption

$$q(\mathbf{T}) = \prod_{i=1}^k q_i(\mathbf{T}_i), \quad \mathbf{T} = [\mathbf{T}_1, \dots, \mathbf{T}_k], \quad \mathbf{T}_j = \{\mathbf{t}_{ij}\}_{i=1}^n, \quad \mathbf{t}_i = \{\mathbf{T}_{ij}\}_{j=1}^k.$$

Block coordinate optimization of ELBO for $q_j(\mathbf{T}_j)$

$$\begin{aligned} \mathcal{L}(q) &= \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{q(\mathbf{T})} d\mathbf{T} = \int \left[\prod_{i=1}^k q_i(\mathbf{T}_i) \right] \log \frac{p(\mathbf{X}, \mathbf{T})}{\left[\prod_{i=1}^k q_i(\mathbf{T}_i) \right]} \prod_{i=1}^k d\mathbf{T}_i = \\ &= \int \left[\prod_{i=1}^k q_i \right] \log p(\mathbf{X}, \mathbf{T}) \prod_{i=1}^k d\mathbf{T}_i - \sum_{i=1}^k \int \left[\prod_{j=1}^k q_j \right] \log q_i \prod_{j=1}^k d\mathbf{T}_j = \\ &= \int q_j \left[\int \log p(\mathbf{X}, \mathbf{T}) \prod_{i \neq j} q_i d\mathbf{T}_i \right] d\mathbf{T}_j - \\ &\quad - \int q_j \log q_j d\mathbf{T}_j + \text{const}(q_j) \rightarrow \max_{q_j} \end{aligned}$$

Mean field approximation

Block coordinate optimization of ELBO for $q_j(\mathbf{T}_j)$

$$\begin{aligned}\mathcal{L}(q) &= \int q_j \left[\int \log p(\mathbf{X}, \mathbf{T}) \prod_{i \neq j} q_i d\mathbf{T}_i \right] d\mathbf{T}_j - \int q_j \log q_j d\mathbf{T}_j + \text{const}(q_j) = \\ &= \int q_j \log \hat{p}(\mathbf{X}, \mathbf{T}_j) d\mathbf{T}_j - \int q_j \log q_j d\mathbf{T}_j + \text{const}(q_j) \rightarrow \max_{q_j}.\end{aligned}$$

Here we introduce

$$\log \hat{p}(\mathbf{X}, \mathbf{T}_j) = \int \log p(\mathbf{X}, \mathbf{T}) \prod_{i \neq j} q_i d\mathbf{T}_i = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}(q_j)$$

Final ELBO derivation for $q_j(\mathbf{T}_j)$

$$\begin{aligned}\mathcal{L}(q) &= \int q_j(\mathbf{T}_j) \log \hat{p}(\mathbf{X}, \mathbf{T}_j) d\mathbf{T}_j - \int q_j(\mathbf{T}_j) \log q_j(\mathbf{T}_j) d\mathbf{T}_j + \text{const}(q_j) = \\ &\quad \int q_j(\mathbf{T}_j) \log \frac{\hat{p}(\mathbf{X}, \mathbf{T}_j)}{q_j(\mathbf{T}_j)} d\mathbf{T}_j + \text{const}(q_j) = \\ &= -KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \rightarrow \max_{q_j}.\end{aligned}$$

Mean field approximation

Independence assumption

$$q(\mathbf{T}) = \prod_{i=1}^k q_i(\mathbf{T}_i), \quad \mathbf{T} = [\mathbf{T}_1, \dots, \mathbf{T}_k], \quad \mathbf{T}_j = \{\mathbf{t}_{ij}\}_{i=1}^n.$$

ELBO

$$\mathcal{L}(q) = -KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \rightarrow \max_{q_j}.$$

Solution

$$q_j(\mathbf{T}_j) = \text{const} \cdot \hat{p}(\mathbf{X}, \mathbf{T}_j)$$

$$\log \hat{p}(\mathbf{X}, \mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

$$\log q_j(\mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

Mean field approximation

ELBO

$$\mathcal{L}(q) = -KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \rightarrow \max_{q_j}.$$

Solution

$$\log q_j(\mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

Assumptions:

- ▶ $\mathbf{T} = [\mathbf{T}_1, \mathbf{T}_2] = [\mathbf{Z}, \boldsymbol{\theta}]$, $q(\mathbf{T}) = q(\mathbf{T}_1) \cdot q(\mathbf{T}_2) = q(\mathbf{Z}) \cdot q(\boldsymbol{\theta})$.
- ▶ restrict a class of probability distributions for $\boldsymbol{\theta}$ to Dirac delta functions:

$$q_2 = q(\mathbf{T}_2) = q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*).$$

Under the restrictions the exact solution for q_2 is not reached (KL can be greater than 0).

Mean field approximation

General solution

$$\log q_j(\mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

Solution for $q_1 = q(\mathbf{Z})$

$$\begin{aligned}\log q(\mathbf{Z}) &= \int q(\boldsymbol{\theta}) \log p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const} = \\ &= \int \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \log p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const} = \\ &= \log p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^*) + \text{const.}\end{aligned}$$

EM-algorithm (E-step)

$$q(\mathbf{Z}) = \arg \max_q \mathcal{L}(q, \boldsymbol{\theta}^*) = \arg \min_q KL(q||p) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^*).$$

Mean field approximation

ELBO

$$\mathcal{L}(q) = -KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \rightarrow \max_{q_j}.$$

ELBO maximization w.r.t. $q_2 = q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$

$$\begin{aligned}\mathcal{L}(q_1, q_2) &= -KL(q(\boldsymbol{\theta}) || \hat{p}(\mathbf{X}, \boldsymbol{\theta})) + \text{const}(\boldsymbol{\theta}^*) \\ &= \int q(\boldsymbol{\theta}) \log \frac{\hat{p}(\mathbf{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} + \text{const}(\boldsymbol{\theta}^*) \\ &= \int q(\boldsymbol{\theta}) \log \hat{p}(\mathbf{X}, \boldsymbol{\theta}) d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \log q(\boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const}(\boldsymbol{\theta}^*) \\ &= \int \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \log \hat{p}(\mathbf{X}, \boldsymbol{\theta}) d\boldsymbol{\theta} + \text{const}(\boldsymbol{\theta}^*) \rightarrow \max_{\boldsymbol{\theta}^*}\end{aligned}$$

Mean field approximation

ELBO maximization w.r.t. $q_2 = q(\theta) = \delta(\theta - \theta^*)$

$$\begin{aligned}\mathcal{L}(q_1, q_2) &= \int \delta(\theta - \theta^*) \log \hat{p}(\mathbf{X}, \theta) d\theta + \text{const} = \log \hat{p}(\mathbf{X}, \theta^*) + \text{const} \\ &= \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const} = \mathbb{E}_{q_1} \log p(\mathbf{X}, \mathbf{Z}, \theta^*) + \text{const} \\ &= \int q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z} | \theta^*) d\mathbf{Z} + \log p(\theta^*) + \text{const} \rightarrow \max_{\theta^*}\end{aligned}$$

EM-algorithm (M-step)

$$\begin{aligned}\mathcal{L}(q, \theta) &= \int q(\mathbf{Z}) \log \frac{p(\mathbf{X}, \mathbf{Z} | \theta)}{q(\mathbf{Z})} d\mathbf{Z} \\ &= \int q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z} | \theta) d\mathbf{Z} + \text{const} \rightarrow \max_{\theta}\end{aligned}$$

Mean field approximation

Solution

$$\log q_j(\mathbf{T}_j) = \mathbb{E}_{i \neq j} \log p(\mathbf{X}, \mathbf{T}) + \text{const}$$

EM algorithm (special case)

- ▶ Initialize θ^* ;
- ▶ E-step

$$q(\mathbf{Z}) = \arg \max_q \mathcal{L}(q, \theta^*) = \arg \min_q KL(q||p) = p(\mathbf{Z}|\mathbf{X}, \theta^*);$$

- ▶ M-step
$$\theta^* = \arg \max_{\theta} \mathcal{L}(q, \theta);$$
- ▶ Repeat E-step and M-step until convergence.

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
 - Posterior collapse and decoder weakening techniques
 - IWAE
 - PixelVAE, Hierarchical VAE
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
 - Posterior collapse and decoder weakening techniques
 - IWAE
 - PixelVAE, Hierarchical VAE
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

VAE limitations

- ▶ Poor generative distribution (decoder)

$$p(\mathbf{x}|\mathbf{z}, \theta) = \mathcal{N}(\mathbf{x}|\mu_\theta(\mathbf{z}), \sigma_\theta^2(\mathbf{z})) \quad \text{or} \quad = \text{Softmax}(\pi_\theta(\mathbf{z})).$$

- ▶ Loose lower bound

$$\log p(\mathbf{x}|\theta) - \mathcal{L}(q, \theta) = (?).$$

- ▶ Poor prior distribution

$$p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I}).$$

- ▶ Poor variational posterior distribution (encoder)

$$q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \sigma_\phi^2(\mathbf{x})).$$

Posterior collapse: toy example

Let define latent variable model in the following way:

$$p(\mathbf{x}|\theta) = \int p(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z}, \theta)p(\mathbf{z})d\mathbf{z}$$

- ▶ prior distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$;
- ▶ probabilistic model $p(\mathbf{x}|\mathbf{z}, \theta) = \mathcal{N}(\mathbf{x}|\mu_\theta(\mathbf{z}), \sigma_\theta(\mathbf{z}))$ (diagonal covariance);
- ▶ variational posterior $q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x}))$ (diagonal covariance).

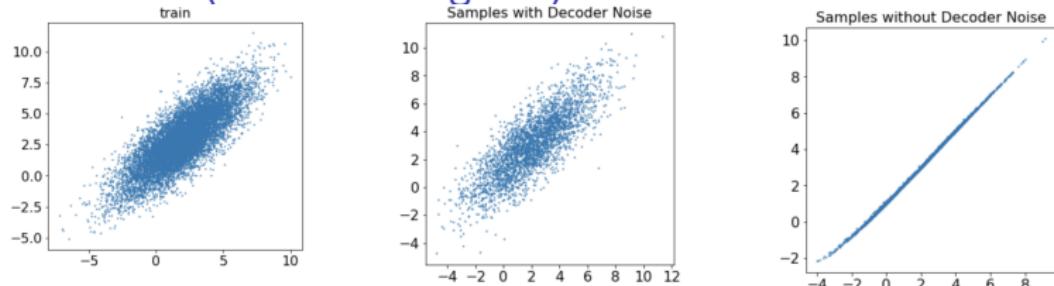
Let data distribution is $\pi(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Possible cases:

- ▶ covariance matrix $\boldsymbol{\Sigma}$ is diagonal (univariate case);
- ▶ covariance matrix $\boldsymbol{\Sigma}$ is **not** diagonal (multivariate case).

What is the difference?

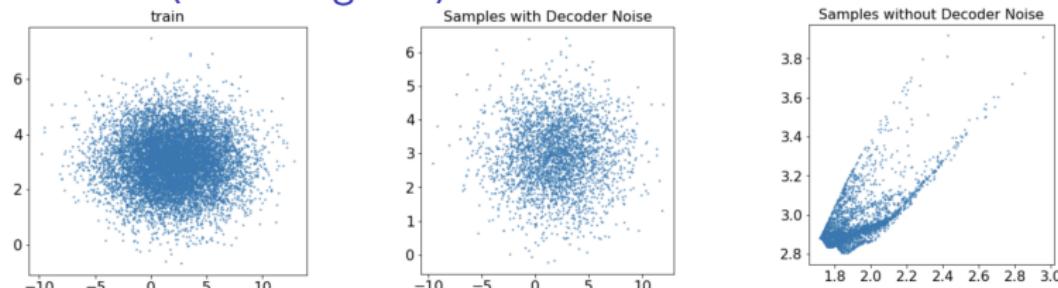
Posterior collapse: toy example + VLAE

Multivariate (Σ is non-diagonal)



The encoder uses latent variables to model data.

Univariate (Σ is diagonal)



Latent variables are not used, since the decoder could model the data without the encoder.

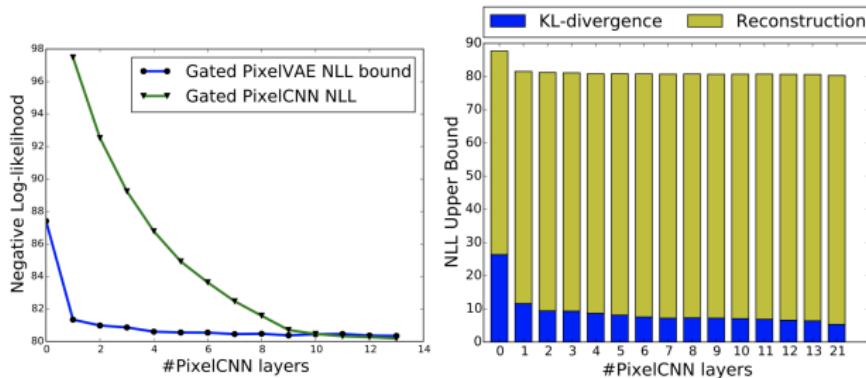
PixelVAE

Autoregressive decoder

$$p(\mathbf{x}|\mathbf{z}, \theta) = \prod_{j=1}^m p(x_j|\mathbf{x}_{1:j-1}, \mathbf{z}, \theta)$$

- ▶ Global structure is captured by latent variables.
- ▶ Local statistics are captured by limited receptive field autoregressive model.

MNIST results



Variational Lossy AutoEncoder

Lossy code via explicit information placement

$$p(\mathbf{x}|\mathbf{z}, \theta) = \prod_{i=1}^m p(x_i|\mathbf{z}, \mathbf{x}_{\text{WindowAround}(i)}, \theta).$$

- ▶ $\text{WindowAround}(i)$ restricts the receptive field (it forbids to represent arbitrarily complex distribution over \mathbf{x} without dependence on \mathbf{z}).
- ▶ Local statistics of 2D images (texture) will be modeled by a small local window.
- ▶ Global structural information (shapes) is long-range dependency that can only be communicated through latent code \mathbf{z} .

Variational Lossy AutoEncoder

- ▶ Can VLAE learn lossy codes that encode global statistics?
- ▶ Does using AF priors improves upon using IAF posteriors as predicted by theory?
- ▶ Does using autoregressive decoding distributions improve density estimation performance?

CIFAR10

MNIST

Model	NLL Test
Normalizing flows (Rezende & Mohamed, 2015)	85.10
DRAW (Gregor et al., 2015)	< 80.97
Discrete VAE (Rollefson, 2016)	81.01
PixelRNN (van den Oord et al., 2016a)	79.20
IAF VAE (Kingma et al., 2016)	79.88
AF VAE	79.30
VLAE	79.03

Method	bits/dim \leq
<i>Results with tractable likelihood models:</i>	
Uniform distribution [1]	8.00
Multivariate Gaussian [1]	4.70
NICE [2]	4.48
Deep GMMS [3]	4.00
Real NVP [4]	3.49
PixelCNN [1]	3.14
Gated PixelCNN [5]	3.03
PixelRNN [1]	3.00
PixelCNN++ [6]	2.92
<i>Results with variationally trained latent-variable models:</i>	
Deep Diffusion [7]	5.40
Convolutional DRAW [8]	3.58
ResNet VAE with IAF [9]	3.11
ResNet VLAE	3.04
DenseNet VLAE	2.95

Posterior collapse

LVM

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z}) d\mathbf{z}$$

ELBO objective

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - KL(q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})||p(\mathbf{z})).$$

More powerful $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ leads to more powerful generative model $p(\mathbf{x}|\boldsymbol{\theta})$.

Extreme cast

$$p(\mathbf{x}|\boldsymbol{\theta}) \in \mathcal{P} = \{p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) | \forall \mathbf{z}, \boldsymbol{\theta}\}.$$

If the decoder $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ is too powerful (it could model $p(\mathbf{x}|\boldsymbol{\theta})$), then ELBO avoids paying any cost $KL(q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})||p(\mathbf{z}))$ ($q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) \approx p(\mathbf{z})$), the variational posterior $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})$ will not carry any information about \mathbf{x} , the latent variables \mathbf{z} becomes irrelevant.

Autoregressive VAE decoder

How to make the generative model $p(\mathbf{x}|\mathbf{z}, \theta)$ more powerful?

PixelVAE/VLAE

$$p(\mathbf{x}|\mathbf{z}, \theta) = \prod_{j=1}^m p(x_j | \mathbf{x}_{1:j-1}, \mathbf{z}, \theta)$$

- ▶ Global structure is captured by latent variables \mathbf{z} .
- ▶ Local statistics are captured by limited receptive field of autoregressive context $\mathbf{x}_{1:j-1}$.

PixelVAE/VLAE models use the autoregressive PixelCNN decoder model with small number of layers to limit receptive field.

Decoder weakening techniques

How to force the model encode information about \mathbf{x} into \mathbf{z} ?

KL annealing

$$\mathcal{L}(\phi, \theta, \beta) = \mathbb{E}_{q(z|x, \phi)} \log p(x|z, \theta) - \beta \cdot KL(q(z|x, \phi) || p(z))$$

Start training with $\beta = 0$, increase it until $\beta = 1$ during training.

Free bits

$$\mathcal{L}(\phi, \theta, \lambda) = \mathbb{E}_{q(z|x, \phi)} \log p(x|z, \theta) - \max(\lambda, KL(q(z|x, \phi) || p(z))).$$

It ensures the use of less than λ bits of information and results in $KL(q(z|x, \phi) || p(z)) \geq \lambda$.

Bowman S. R. et al. *Generating Sentences from a Continuous Space*, 2015

Kingma D. P. et al. *Improving Variational Inference with Inverse Autoregressive Flow*, 2016

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

Posterior collapse and decoder weakening techniques

IWAE

PixelVAE, Hierarchical VAE

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

VAE limitations

- ▶ Poor generative distribution (decoder)

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}|\mu_{\boldsymbol{\theta}}(\mathbf{z}), \sigma_{\boldsymbol{\theta}}^2(\mathbf{z})) \quad \text{or} \quad = \text{Softmax}(\pi_{\boldsymbol{\theta}}(\mathbf{z})).$$

- ▶ **Loose lower bound**

$$\log p(\mathbf{x}|\boldsymbol{\theta}) - \mathcal{L}(q, \boldsymbol{\theta}) = (?).$$

- ▶ Poor prior distribution

$$p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I}).$$

- ▶ Poor variational posterior distribution (encoder)

$$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{z}|\mu_{\boldsymbol{\phi}}(\mathbf{x}), \sigma_{\boldsymbol{\phi}}^2(\mathbf{x})).$$

Importance sampling

LVM

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\theta}) &= \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z} = \int \left[\frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})} \right] q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) d\mathbf{z} \\ &= \int f(\mathbf{x}, \mathbf{z}) q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) d\mathbf{z} = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})} f(\mathbf{x}, \mathbf{z}) \end{aligned}$$

Here $f(\mathbf{x}, \mathbf{z}) = \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})}$.

ELBO: derivation 1

$$\begin{aligned} \log p(\mathbf{x}|\boldsymbol{\theta}) &= \log \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})} f(\mathbf{x}, \mathbf{z}) \geq \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})} \log f(\mathbf{x}, \mathbf{z}) = \\ &= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})} \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})} = \mathcal{L}(q, \boldsymbol{\theta}). \end{aligned}$$

$f(\mathbf{x}, \mathbf{z})$ could be any function that satisfies $p(\mathbf{x}|\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z} \sim q} f(\mathbf{x}, \mathbf{z})$. Could we choose better $f(\mathbf{x}, \mathbf{z})$?

Importance Weighted Autoencoders (IWAE)

$$p(\mathbf{x}|\theta) = \int \left[\frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x}, \phi)} \right] q(\mathbf{z}|\mathbf{x}, \phi) d\mathbf{z} = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \phi)} f(\mathbf{x}, \mathbf{z})$$

Let define

$$f(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_K) = \frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k|\theta)}{q(\mathbf{z}_k|\mathbf{x}, \phi)}$$

$$\mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} f(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_K) = p(\mathbf{x}|\theta)$$

ELBO

$$\begin{aligned} \log p(\mathbf{x}|\theta) &= \log \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x})} f(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_K) \geq \\ &\geq \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \log f(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_K) = \\ &= \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \log \left[\frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k|\theta)}{q(\mathbf{z}_k|\mathbf{x}, \phi)} \right] = \mathcal{L}_K(q, \theta). \end{aligned}$$

Importance Weighted Autoencoders (IWAE)

VAE objective

$$\log p(\mathbf{x}|\theta) \geq \mathcal{L}(q, \theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x}, \phi)} \rightarrow \max_{q, \theta}$$

$$\mathcal{L}(q, \theta) = \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \left(\frac{1}{K} \sum_{k=1}^K \log \frac{p(\mathbf{x}, \mathbf{z}_k|\theta)}{q(\mathbf{z}_k|\mathbf{x}, \phi)} \right) \rightarrow \max_{q, \theta}.$$

IWAE objective

$$\mathcal{L}_K(q, \theta) = \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \log \left(\frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k|\theta)}{q(\mathbf{z}_k|\mathbf{x}, \phi)} \right) \rightarrow \max_{q, \theta}.$$

If $K = 1$, these objectives coincide.

Importance Weighted Autoencoders (IWAE)

Theorem

1. $\log p(\mathbf{x}|\theta) \geq \mathcal{L}_K(q, \theta) \geq \mathcal{L}_M(q, \theta)$, for $K \geq M$;
2. $\log p(\mathbf{x}|\theta) = \lim_{K \rightarrow \infty} \mathcal{L}_K(q, \theta)$ if $\frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x}, \phi)}$ is bounded.

If $K > 1$ the bound could be tighter.

$$\mathcal{L}(q, \theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x}, \phi)};$$

$$\mathcal{L}_K(q, \theta) = \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \log \left(\frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k|\theta)}{q(\mathbf{z}_k|\mathbf{x}, \phi)} \right).$$

- ▶ $\mathcal{L}_1(q, \theta) = \mathcal{L}(q, \theta)$;
- ▶ $\mathcal{L}_\infty(q, \theta) = \log p(\mathbf{x}|\theta)$.
- ▶ Which $q^*(\mathbf{z}|\mathbf{x}, \phi)$ gives $\mathcal{L}(q^*, \theta) = \log p(\mathbf{x}|\theta)$?

IWAE

Theorem

1. $\log p(\mathbf{x}|\theta) \geq \mathcal{L}_K(q, \theta) \geq \mathcal{L}_M(q, \theta)$, for $K \geq M$;
2. $\log p(\mathbf{x}|\theta) = \lim_{K \rightarrow \infty} \mathcal{L}_K(q, \theta)$ if $\frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})}$ is bounded.

Proof of 1.

$$\begin{aligned}\mathcal{L}_K(q, \theta) &= \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K} \log \left(\frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k | \theta)}{q(\mathbf{z}_k | \mathbf{x})} \right) = \\ &= \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K} \log \mathbb{E}_{k_1, \dots, k_M} \left(\frac{1}{M} \sum_{m=1}^M \frac{p(\mathbf{x}, \mathbf{z}_{k_m} | \theta)}{q(\mathbf{z}_{k_m} | \mathbf{x})} \right) \geq \\ &\geq \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K} \mathbb{E}_{k_1, \dots, k_M} \log \left(\frac{1}{M} \sum_{m=1}^M \frac{p(\mathbf{x}, \mathbf{z}_{k_m} | \theta)}{q(\mathbf{z}_{k_m} | \mathbf{x})} \right) = \\ &= \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_M} \log \left(\frac{1}{M} \sum_{m=1}^M \frac{p(\mathbf{x}, \mathbf{z}_m | \theta)}{q(\mathbf{z}_m | \mathbf{x})} \right) = \mathcal{L}_M(q, \theta)\end{aligned}$$

$$\frac{a_1 + \dots + a_K}{K} = \mathbb{E}_{k_1, \dots, k_M} \frac{a_{k_1} + \dots + a_{k_M}}{M}, \quad k_1, \dots, k_M \sim U[1, K]$$

Theorem

1. $\log p(\mathbf{x}|\theta) \geq \mathcal{L}_K(q, \theta) \geq \mathcal{L}_M(q, \theta)$, for $K \geq M$;
2. $\log p(\mathbf{x}|\theta) = \lim_{K \rightarrow \infty} \mathcal{L}_K(q, \theta)$ if $\frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})}$ is bounded.

Proof of 2.

Consider r.v. $\xi_K = \frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k|\theta)}{q(\mathbf{z}_k|\mathbf{x})}$.

If summands are bounded, then (from the strong law of large numbers)

$$\xi_K \xrightarrow[K \rightarrow \infty]{\text{a.s.}} \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})} = p(\mathbf{x}|\theta).$$

Hence $\mathcal{L}_K(q, \theta) = \mathbb{E} \log \xi_K$ converges to $\log p(\mathbf{x}|\theta)$ as $K \rightarrow \infty$.

Importance Weighted Autoencoders (IWAE)

Theorem

1. $\log p(\mathbf{x}|\theta) \geq \mathcal{L}_K(q, \theta) \geq \mathcal{L}_M(q, \theta)$, for $K \geq M$;
2. $\log p(\mathbf{x}|\theta) = \lim_{K \rightarrow \infty} \mathcal{L}_K(q, \theta)$ if $\frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x}, \phi)}$ is bounded.

If $K > 1$ the bound could be tighter.

$$\mathcal{L}(q, \theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x}, \phi)};$$

$$\mathcal{L}_K(q, \theta) = \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \log \left(\frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k|\theta)}{q(\mathbf{z}_k|\mathbf{x}, \phi)} \right).$$

- ▶ $\mathcal{L}_1(q, \theta) = \mathcal{L}(q, \theta)$;
- ▶ $\mathcal{L}_\infty(q, \theta) = \log p(\mathbf{x}|\theta)$.
- ▶ Which $q^*(\mathbf{z}|\mathbf{x}, \phi)$ gives $\mathcal{L}(q^*, \theta) = \log p(\mathbf{x}|\theta)$?
- ▶ Which $q^*(\mathbf{z}|\mathbf{x}, \phi)$ gives $\mathcal{L}(q^*, \theta) = \mathcal{L}_K(q, \theta)$?

Importance Weighted Autoencoders (IWAE)

Theorem

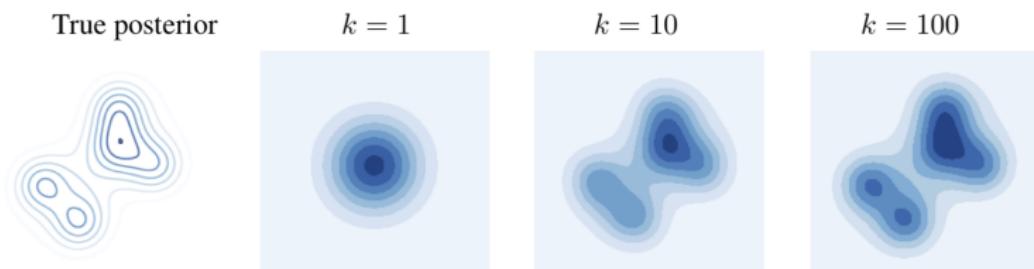
$\mathcal{L}(q^*, \theta) = \mathcal{L}_K(q, \theta)$ for the following variational distribution

$$q^*(\mathbf{z}|\mathbf{x}, \phi) = \mathbb{E}_{\mathbf{z}_2, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x})} q_{IW}(\mathbf{z}|\mathbf{x}, \mathbf{z}_{2:K}),$$

where

$$q_{IW}(\mathbf{z}|\mathbf{x}, \mathbf{z}_{2:K}) = \frac{\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})}}{\frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k)}{q(\mathbf{z}_k|\mathbf{x})}} q(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{\frac{1}{K} \left(\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} + \sum_{k=2}^K \frac{p(\mathbf{x}, \mathbf{z}_k)}{q(\mathbf{z}_k|\mathbf{x})} \right)}.$$

IWAE posterior



How to determine whether all VAE latent variables are informative?

$$A_i = \text{cov}_{\mathbf{x}} (\mathbb{E}_{q(z_i|\mathbf{x})}[z_i]) > 0.01 \Leftrightarrow z_i \text{ is active}$$

# stoch. layers	k	MNIST				OMNIGLOT			
		VAE		IWAE		VAE		IWAE	
		NLL	active units	NLL	active units	NLL	active units	NLL	active units
1	1	86.76	19	86.76	19	108.11	28	108.11	28
	5	86.47	20	85.54	22	107.62	28	106.12	34
	50	86.35	20	84.78	25	107.80	28	104.67	41
2	1	85.33	16+5	85.33	16+5	107.58	28+4	107.56	30+5
	5	85.01	17+5	83.89	21+5	106.31	30+5	104.79	38+6
	50	84.78	17+5	82.90	26+7	106.30	30+5	103.38	44+7

Importance Weighted Autoencoders (IWAE)

Objective

$$\mathcal{L}_K(q, \theta) = \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \log \left(\frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k | \theta)}{q(\mathbf{z}_k | \mathbf{x}, \phi)} \right) \rightarrow \max_{\phi, \theta} .$$

Gradient

$$\Delta_K = \nabla_{\theta, \phi} \log \left(\frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k | \theta)}{q(\mathbf{z}_k | \mathbf{x}, \phi)} \right), \quad \mathbf{z}_k \sim q(\mathbf{z} | \mathbf{x}, \phi).$$

Theorem

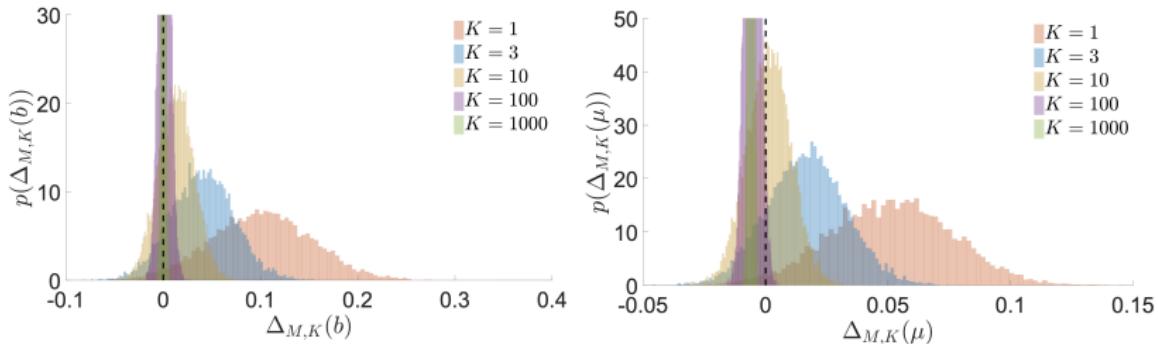
$$\text{SNR}_K = \frac{\mathbb{E}[\Delta_K]}{\sigma(\Delta_K)}; \quad \text{SNR}_K(\theta) = O(\sqrt{K}); \quad \text{SNR}_K(\phi) = O\left(\sqrt{\frac{1}{K}}\right).$$

Hence, increasing K vanishes gradient signal of inference network $q(\mathbf{z} | \mathbf{x}, \phi)$.

Importance Weighted Autoencoders (IWAE)

Theorem

$$\text{SNR}_K = \frac{\mathbb{E}[\Delta_K]}{\sigma(\Delta_K)}; \quad \text{SNR}_K(\theta) = O(\sqrt{K}); \quad \text{SNR}_K(\phi) = O\left(\sqrt{\frac{1}{K}}\right).$$

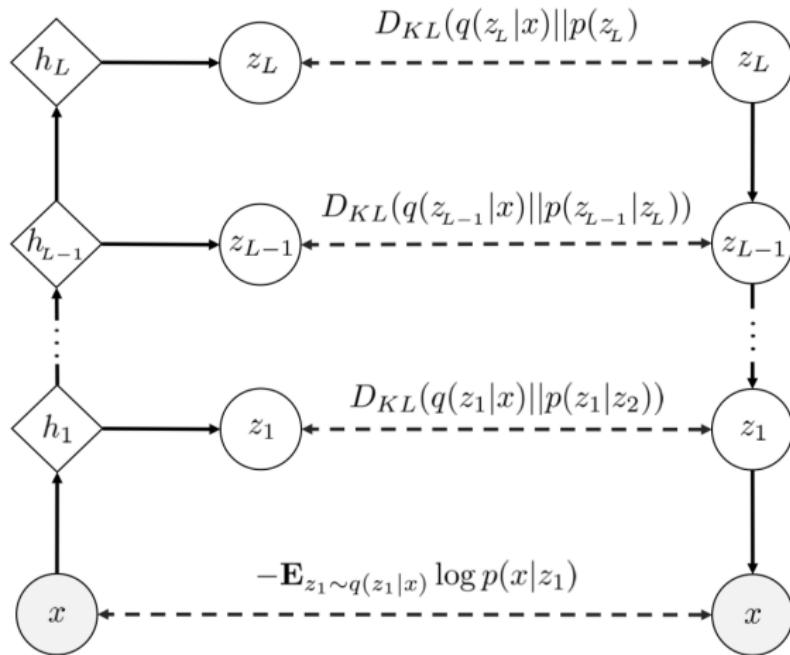


- ▶ IWAE makes the variational bound tighter and extends the class of variational distributions.
- ▶ Gradient signal becomes really small, training is complicated.
- ▶ IWAE is a standard quality measure for VAE models.

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
 - Posterior collapse and decoder weakening techniques
 - IWAE
 - PixelVAE, Hierarchical VAE
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

Hierarchical VAE



PixelVAE, 2016

Hierarchical decomposition

$$p(\mathbf{z}_1, \dots, \mathbf{z}_L) = p(\mathbf{z}_L)p(\mathbf{z}_{L-1}|\mathbf{z}_L)\dots p(\mathbf{z}_1, \mathbf{z}_2);$$
$$q(\mathbf{z}_1, \dots, \mathbf{z}_L|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x})\dots q(\mathbf{z}_L|\mathbf{x}).$$

ELBO

$$\begin{aligned}\mathcal{L}(q, \theta) &= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \theta) - KL(q(\mathbf{z}_1, \dots, \mathbf{z}_L|\mathbf{x})||p(\mathbf{z}_1, \dots, \mathbf{z}_L)) \\ &= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \theta) - \int \prod_{j=1}^L q(\mathbf{z}_j|\mathbf{x}) \sum_{i=1}^L \log \frac{q(\mathbf{z}_i|\mathbf{x})}{p(\mathbf{z}_i|\mathbf{z}_{i+1})} d\mathbf{z}_1 \dots d\mathbf{z}_L \\ &= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \theta) - \sum_{i=1}^L \int \prod_{j=1}^L q(\mathbf{z}_j|\mathbf{x}) \log \frac{q(\mathbf{z}_i|\mathbf{x})}{p(\mathbf{z}_i|\mathbf{z}_{i+1})} d\mathbf{z}_1 \dots d\mathbf{z}_L \\ &= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \theta) - \sum_{i=1}^L \int q(\mathbf{z}_{i+1}|\mathbf{x}) q(\mathbf{z}_i|\mathbf{x}) \log \frac{q(\mathbf{z}_i|\mathbf{x})}{p(\mathbf{z}_i|\mathbf{z}_{i+1})} d\mathbf{z}_i d\mathbf{z}_{i+1} \\ &= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}_1, \theta) - \sum_{i=1}^L \mathbb{E}_{q(\mathbf{z}_{i+1}|\mathbf{x})} [KL(q(\mathbf{z}_i|\mathbf{x})||p(\mathbf{z}_i|\mathbf{z}_{i+1}))]\end{aligned}$$

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

Flows intuition

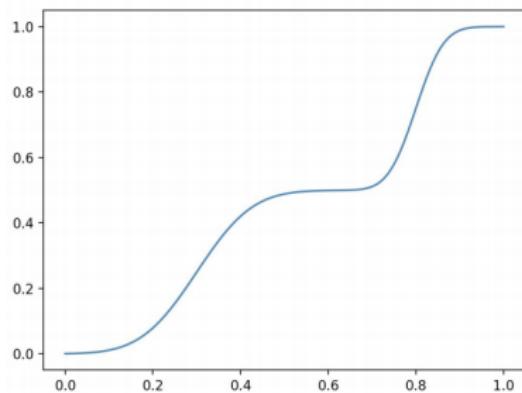
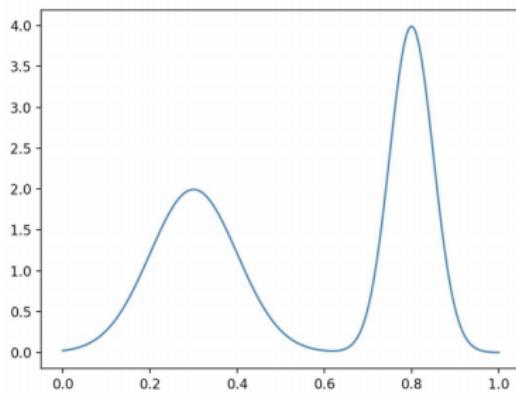
Let ξ be a random variable with density $p(\xi)$. Then

$$\eta = F(\xi) = \int_{-\infty}^{\xi} p(t)dt \sim U[0, 1].$$

$$P(\eta < y) = P(F(\xi) < y) = P(\xi < F^{-1}(y)) = F(F^{-1}(y)) = y$$

Hence

$$\eta \sim U[0, 1]; \quad \xi = F^{-1}(\eta) \quad \Rightarrow \quad \xi \sim p(\xi).$$



Flows intuition

- ▶ Let $z \sim p(z)$ is a random variable with base distribution $p(z) = U[0, 1]$.
- ▶ Let $x \sim p(x)$ is a random variable with complex distribution $p(x)$ and cdf $F(x)$.
- ▶ Then noise variable z can be transformed to x using inverse cdf F^{-1} ($x = F^{-1}(z)$).

How to transform random variable z which has a distribution different from uniform to x ?

- ▶ Let $z \sim p(z)$ is a random variable with base distribution $p(z)$ and cdf $G(z)$.
- ▶ Then $z_0 = G(z)$ has base distribution $p(z_0) = U[0, 1]$.
- ▶ Let $x \sim p(x)$ is a random variable with complex distribution $p(x)$ and cdf $F(x)$.
- ▶ Then noise variable z can be transformed to x using cdf G and inverse cdf F^{-1} ($x = F^{-1}(z_0) = F^{-1}(G(z))$).

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

Residual Flows

Matrix determinant lemma

$$\det(\mathbf{I}_m + \mathbf{V}\mathbf{W}^T) = \det(\mathbf{I}_d + \mathbf{W}^T\mathbf{V}), \quad \text{where } \mathbf{V}, \mathbf{W} \in \mathbb{R}^{m \times d}.$$

Planar flow

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{z} + \mathbf{v} \sigma(\mathbf{w}^T \mathbf{z} + b).$$

Here $\boldsymbol{\theta} = \{\mathbf{v}, \mathbf{w}, b\}$, $\sigma(\cdot)$ is a smooth element-wise non-linearity.

$$\left| \det \left(\frac{\partial g(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}} \right) \right| = \left| \det (\mathbf{I} + \sigma'(\mathbf{w}^T \mathbf{z} + b) \mathbf{v} \mathbf{w}^T) \right| = \left| 1 + \sigma'(\mathbf{w}^T \mathbf{z} + b) \mathbf{w}^T \mathbf{v} \right|$$

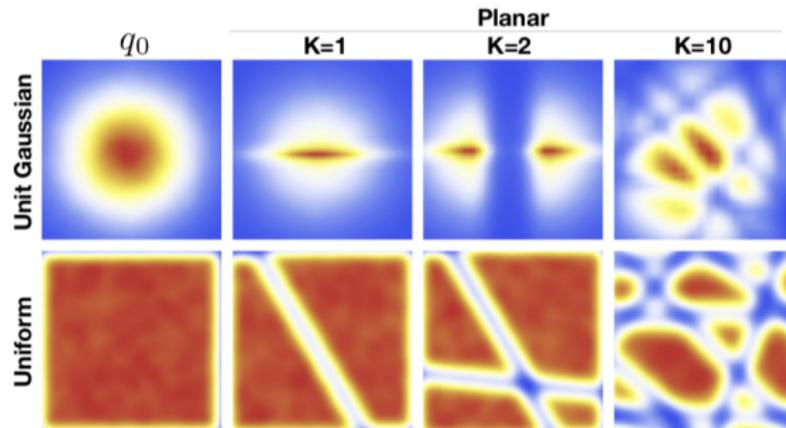
The transformation is invertible, for example, if

$$\sigma = \tanh; \quad \sigma'(\mathbf{w}^T \mathbf{z} + b) \mathbf{w}^T \mathbf{v} \geq -1.$$

Residual Flows

Expressiveness of planar flows

$$\mathbf{z}_K = g_1 \circ \cdots \circ g_K(\mathbf{z}); \quad g_k = g(\mathbf{z}_k, \theta_k) = \mathbf{z}_k + \mathbf{v}_k \sigma(\mathbf{w}_k^T \mathbf{z}_k + b_k).$$



Sylvester flow: planar flow extension

$$g(\mathbf{z}, \theta) = \mathbf{z} + \mathbf{V} \sigma(\mathbf{W}^T \mathbf{z} + \mathbf{b}).$$

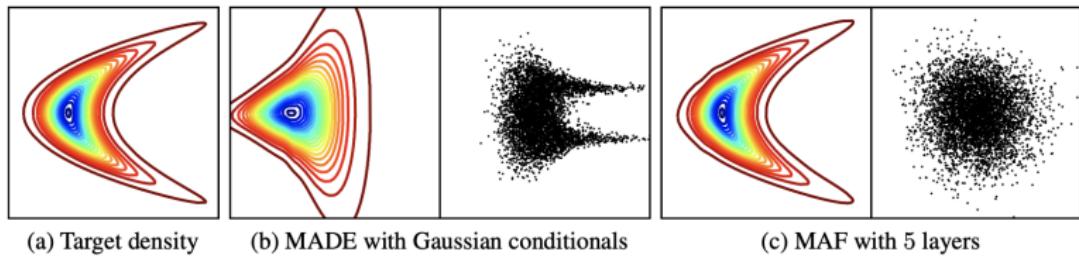
Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015
Berg R. et al. Sylvester normalizing flows for variational inference, 2018

Masked autoregressive flow (MAF)

Gaussian autoregressive model

$$p(\mathbf{x}|\theta) = \prod_{j=1}^m p(x_j|\mathbf{x}_{1:j-1}, \theta) = \prod_{j=1}^m \mathcal{N}(x_j | \mu_j(\mathbf{x}_{1:j-1}), \sigma_j^2(\mathbf{x}_{1:j-1})).$$

We could use MADE for the conditionals. Samples from the base distribution could be an indicator of how good the flow was fitted.



MAF is just a stacked MADE model with different ordering.

- ▶ Parallel density estimation.
- ▶ Sequential sampling.

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

Autoregressive flows

$$x_j = \tau(z_j, c(\mathbf{z}_{1:j-1})) \Leftrightarrow z_j = \tau^{-1}(x_j, c(\mathbf{z}_{1:j-1}))$$

- ▶ $\tau(\cdot, \cdot)$ – coupling law (invertible by first argument, differentiable).
- ▶ $c(\cdot)$ – coupling function (do not need to be invertible, could be neural network).

Coupling law $\tau(\cdot, \cdot)$

- ▶ $\tau(x, c) = x + c$ – additive;
- ▶ $\tau(x, c) = x \odot c_1 + c_2$ – affine.

What is the Jacobian for the additive/affine coupling law?

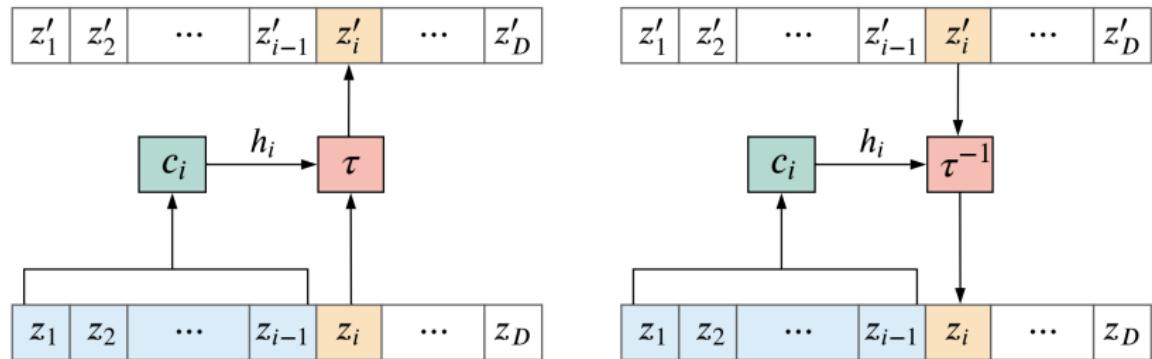
Jacobian

$$\det \left(\frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) = \prod_{j=1}^m \frac{\partial x_j}{\partial z_j} = \prod_{j=1}^m \frac{\partial \tau(z_j, c(\mathbf{z}_{1:j-1}))}{\partial z_j}$$

Autoregressive flows

Forward and inverse transforms

$$x_j = \tau(z_j, c(\mathbf{z}_{1:j-1})) \Leftrightarrow z_j = \tau^{-1}(x_j, c(\mathbf{z}_{1:j-1}))$$



- ▶ Forward transform is **not sequential**.
- ▶ Inverse transform is **sequential**.

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

Inverse gaussian autoregressive flow (IAF)

Let's use the following reparametrization: $\tilde{\sigma} = \frac{1}{\sigma}$; $\tilde{\mu} = -\frac{\mu}{\sigma}$.

Gaussian autoregressive flow

$$x_j = \sigma_j(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_j(\mathbf{x}_{1:j-1}) = (z_j - \tilde{\mu}_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\tilde{\sigma}_j(\mathbf{x}_{1:j-1})}$$

$$z_j = (x_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})} = \tilde{\sigma}_j(\mathbf{x}_{1:j-1}) \cdot x_j + \tilde{\mu}_j(\mathbf{x}_{1:j-1}).$$

Let's just swap \mathbf{z} and \mathbf{x} .

Inverse gaussian autoregressive flow

$$\mathbf{x} = g(\mathbf{z}, \theta) \quad \Rightarrow \quad \textcolor{violet}{x_j} = \tilde{\sigma}_j(\textcolor{teal}{\mathbf{z}_{1:j-1}}) \cdot \textcolor{teal}{z_j} + \tilde{\mu}_j(\textcolor{teal}{\mathbf{z}_{1:j-1}})$$

$$\mathbf{z} = f(\mathbf{x}, \theta) \quad \Rightarrow \quad \textcolor{teal}{z_j} = (\textcolor{violet}{x_j} - \tilde{\mu}_j(\textcolor{teal}{\mathbf{z}_{1:j-1}})) \cdot \frac{1}{\tilde{\sigma}_j(\textcolor{teal}{\mathbf{z}_{1:j-1}})}.$$

Inverse gaussian autoregressive flow (IAF)

Gaussian autoregressive flow: $g(\mathbf{z}, \theta)$

$$x_j = \sigma_j(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_j(\mathbf{x}_{1:j-1}).$$

Inverse transform: $f(\mathbf{x}, \theta)$

$$z_j = (x_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})};$$

$$z_j = \tilde{\sigma}_j(\mathbf{x}_{1:j-1}) \cdot x_j + \tilde{\mu}_j(\mathbf{x}_{1:j-1}).$$

Inverse gaussian autoregressive flow:
 $g(\mathbf{z}, \theta)$

$$x_j = \tilde{\sigma}_j(\mathbf{z}_{1:j-1}) \cdot z_j + \tilde{\mu}_j(\mathbf{z}_{1:j-1}).$$

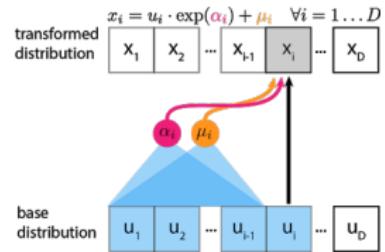
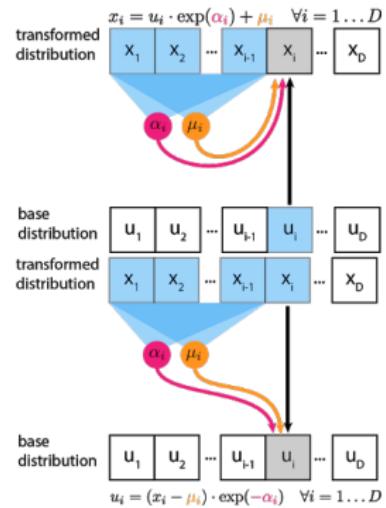


image credit: <https://blog.evjang.com/2018/01/nf2.html>

Inverse gaussian autoregressive flow (IAF)

Inverse gaussian autoregressive flow

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad \textcolor{violet}{x}_j = \tilde{\sigma}_j(\mathbf{z}_{1:j-1}) \cdot \textcolor{teal}{z}_j + \tilde{\mu}_j(\mathbf{z}_{1:j-1})$$

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad \textcolor{teal}{z}_j = (\textcolor{violet}{x}_j - \tilde{\mu}_j(\mathbf{z}_{1:j-1})) \cdot \frac{1}{\tilde{\sigma}_j(\mathbf{z}_{1:j-1})}.$$

Reverse KL for NF

$$KL(p||\pi) = \mathbb{E}_{p(\mathbf{z})} [\log p(\mathbf{z}) - \log |\det(\mathbf{J}_g)| - \log \pi(g(\mathbf{z}, \boldsymbol{\theta}))]$$

- ▶ We need to be able to compute $g(\mathbf{z}, \boldsymbol{\theta})$ and its Jacobian.
- ▶ We need to be able to sample from the density $p(\mathbf{z})$ (do not need to evaluate it) and to evaluate(!) $\pi(\mathbf{x})$.
- ▶ We don't need to think about computing the function $f(\mathbf{x}, \boldsymbol{\theta})$.

Gaussian autoregressive NF

Gaussian AR NF

$$\mathbf{x} = g(\mathbf{z}, \theta) \Rightarrow \mathbf{x}_j = \sigma_j(\mathbf{x}_{1:j-1}) \cdot \mathbf{z}_j + \mu_j(\mathbf{x}_{1:j-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \theta) \Rightarrow \mathbf{z}_j = (\mathbf{x}_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})}.$$

- ▶ Sampling is sequential, density estimation is parallel.
- ▶ Forward KL is a natural loss.

Inverse gaussian AR NF

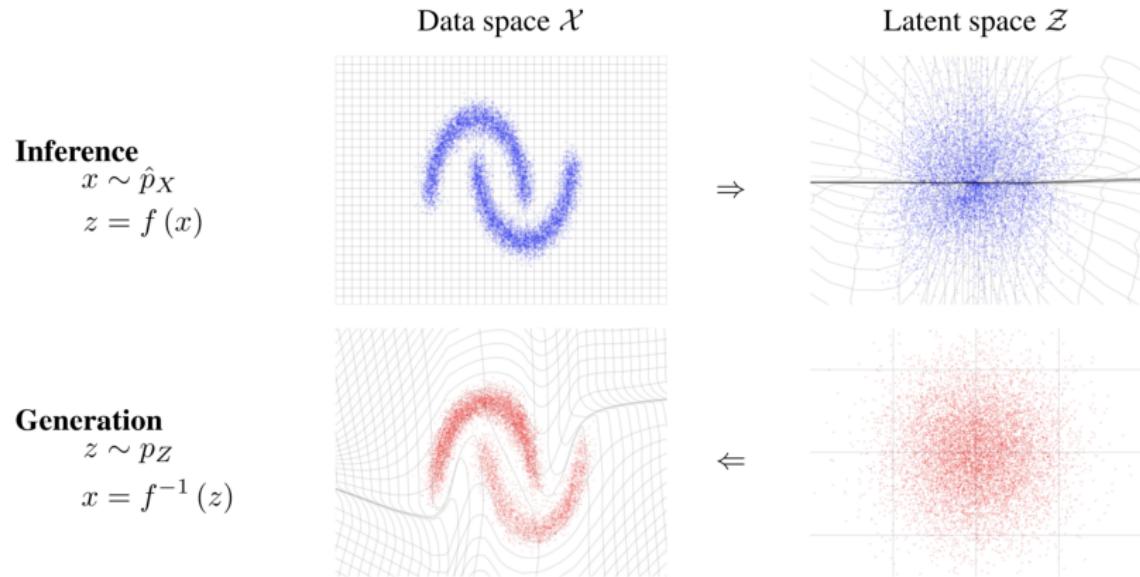
$$\mathbf{x} = g(\mathbf{z}, \theta) \Rightarrow \mathbf{x}_j = \tilde{\sigma}_j(\mathbf{z}_{1:j-1}) \cdot \mathbf{z}_j + \tilde{\mu}_j(\mathbf{z}_{1:j-1})$$

$$\mathbf{z} = f(\mathbf{x}, \theta) \Rightarrow \mathbf{z}_j = (\mathbf{x}_j - \tilde{\mu}_j(\mathbf{z}_{1:j-1})) \cdot \frac{1}{\tilde{\sigma}_j(\mathbf{z}_{1:j-1})}.$$

- ▶ Sampling is parallel, density estimation is sequential.
- ▶ Reverse KL is a natural loss.

Autoregressive flows

Gaussian AR NF and inverse gaussian AR NF are mutually interchangeable.



Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

MAF/IAF pros and cons

MAF

- ▶ Sampling is slow.
- ▶ Likelihood evaluation is fast.

IAF

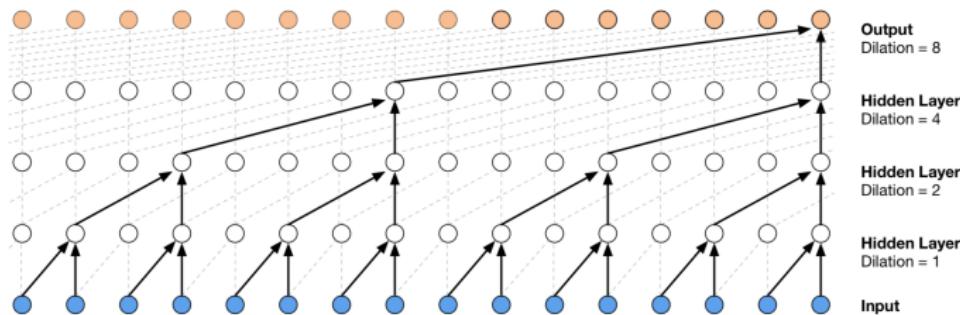
- ▶ Sampling is fast.
- ▶ Likelihood evaluation is slow.

How to take the best of both worlds?

WaveNet

Autoregressive model with caused dilated convolutions for raw audio waveforms generation.

$$p(\mathbf{x}|\theta) = \prod_{t=1}^T p(x_t|\mathbf{x}_{1:t-1}, \theta).$$



AF vs IAF vs RealNVP

MADE/AF

$$\mathbf{x} = \sigma(\mathbf{z}) \odot \mathbf{z} + \mu(\mathbf{x}).$$

Estimating the density $p(\mathbf{x}|\theta)$ - 1 pass, sampling - m passes.

IAF

$$\mathbf{x} = \tilde{\sigma}(\mathbf{z}) \odot \mathbf{z} + \tilde{\mu}(\mathbf{z}).$$

Estimating the density $p(\mathbf{x}|\theta)$ - m passes, sampling - 1 pass.

RealNVP

$$\begin{cases} \mathbf{x}_{1:d} &= \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} &= \mathbf{z}_{d:m} \odot c_1(\mathbf{z}_{1:d}, \theta) + c_2(\mathbf{z}_{1:d}, \theta). \end{cases}$$

AF vs IAF vs RealINVP

RealINVP

$$\begin{cases} \mathbf{x}_{1:d} = \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} = \mathbf{z}_{d:m} \odot c_1(\mathbf{z}_{1:d}, \theta) + c_2(\mathbf{z}_{1:d}, \theta). \end{cases}$$

- ▶ Calculating the density $p(\mathbf{x}|\theta)$ - 1 pass.
- ▶ Sampling - 1 pass.

RealINVP is a special case of AF and IAF:

AF

$$\begin{cases} \mu_j = 0, \sigma_j = 1, j = 1, \dots, d; \\ \mu_j, \sigma_j - \text{functions of } \mathbf{x}_{1:d}, j = d + 1, \dots, m. \end{cases}$$

IAF

$$\begin{cases} \tilde{\mu}_j = 0, \tilde{\sigma}_j = 1, j = 1, \dots, d; \\ \tilde{\mu}_j, \tilde{\sigma}_j - \text{functions of } \mathbf{z}_{1:d}, j = d + 1, \dots, m. \end{cases}$$

Linear flows

RealNVP

$$\begin{cases} \mathbf{z}_{1:d} = \mathbf{x}_{1:d}; \\ \mathbf{z}_{d:m} = \tau(\mathbf{x}_{d:m}, c(\mathbf{x}_{1:d})); \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} = \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} = \tau^{-1}(\mathbf{z}_{d:m}, c(\mathbf{z}_{1:d})). \end{cases}$$

- ▶ First step is a **split** operator which decouples a variable into 2 subparts: \mathbf{x}_1 and \mathbf{x}_2 (usually channel-wise).
- ▶ We should **permute** components between different layers.

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}$$

In general, we need $O(m^3)$ to invert matrix.

Invertibility

- ▶ Diagonal matrix $O(m)$.
- ▶ Triangular matrix $O(m^2)$.
- ▶ It is impossible to parametrize all invertible matrices.

Parallel WaveNet

- ▶ 24kHz instead of 16kHz using increased dilated convolution filter size from 2 to 3.
- ▶ 16-bit signals with mixture of logistics instead of 8-bit signal with 256-way categorical distribution.

Probability density distillation

1. Train usual WaveNet (MAF) via MLE (teacher network).
2. Train IAF WaveNet (student network), which attempts to match the probability of its own samples under the distribution learned by the teacher.

Student objective

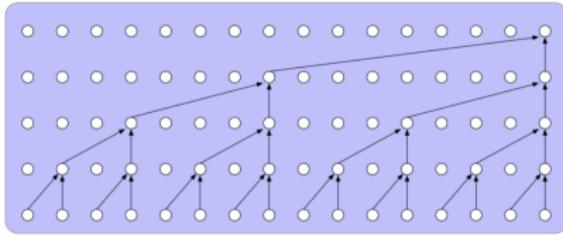
$$KL(p_s || p_t) = H(p_s, p_t) - H(p_s).$$

More than 1000x speed-up relative to original WaveNet!

Parallel WaveNet

WaveNet Teacher

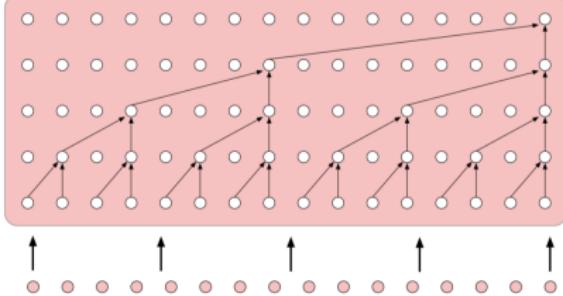
Linguistic features \dashrightarrow



Teacher Output
 $P(x_i|x_{<i})$

WaveNet Student

Linguistic features \dashrightarrow



Generated Samples
 $x_i = g(z_i|z_{<i})$

Student Output
 $P(x_i|z_{<i})$

Input noise
 z_i

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

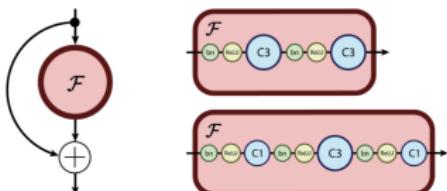
7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

- ▶ Modern neural networks are trained via backpropagation.
- ▶ Residual networks are state of the art in image classification.
- ▶ Backpropagation requires storing the network activations.



Problem

Storing the activations imposes an increasing memory burden.

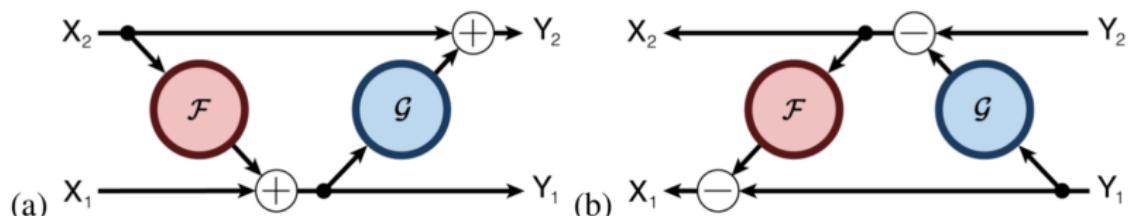
GPUs have limited memory capacity, leading to constraints often exceeded by state-of-the-art architectures (with thousand layers).

NICE

$$\begin{cases} \mathbf{z}_1 = \mathbf{x}_1; \\ \mathbf{z}_2 = \mathbf{x}_2 + \mathcal{F}(\mathbf{x}_1, \theta); \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_1 = \mathbf{z}_1; \\ \mathbf{x}_2 = \mathbf{z}_2 - \mathcal{F}(\mathbf{z}_1, \theta). \end{cases}$$

RevNet

$$\begin{cases} \mathbf{y}_1 = \mathbf{x}_1 + \mathcal{F}(\mathbf{x}_2, \theta); \\ \mathbf{y}_2 = \mathbf{x}_2 + \mathcal{G}(\mathbf{y}_1, \theta); \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_2 = \mathbf{y}_2 - \mathcal{F}(\mathbf{y}_1, \theta); \\ \mathbf{x}_1 = \mathbf{y}_1 - \mathcal{G}(\mathbf{x}_2, \theta). \end{cases}$$



Architecture	CIFAR-10 [15]		CIFAR-100 [15]	
	ResNet	RevNet	ResNet	RevNet
32 (38)	7.14%	7.24%	29.95%	28.96%
110	5.74%	5.76%	26.44%	25.40%
164	5.24%	5.17%	23.37%	23.69%

- ▶ If the network contains non-reversible blocks (poolings, strides), activations for these blocks should be stored.
- ▶ To avoid storing activations in the modern frameworks, the backward pass should be manually redefined.

Hypothesis

The success of deep convolutional networks is based on progressively discarding uninformative variability about the input with respect to the problem at hand.

- ▶ It is difficult to recover images from their hidden representations.
- ▶ Information bottleneck principle: an optimal representation must reduce the MI between an input and its representation to reduce uninformative variability + maximize the MI between the output and its representation to preserve each class from collapsing onto other classes.

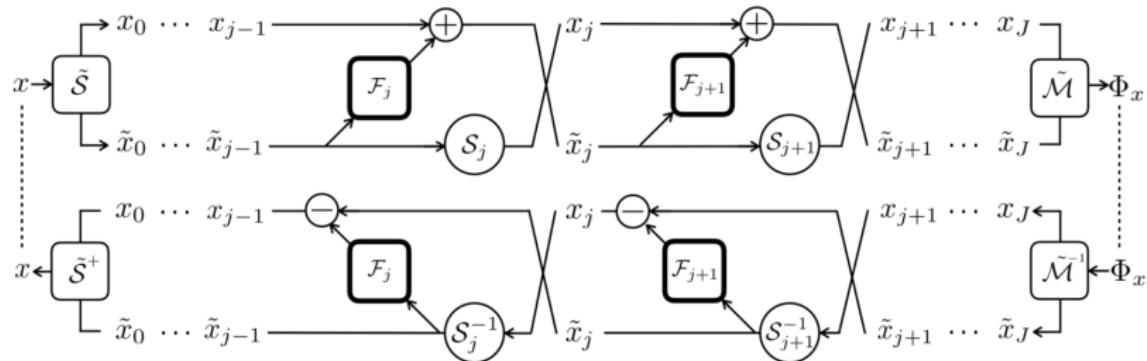
Hypothesis

The success of deep convolutional networks is based on progressively discarding uninformative variability about the input with respect to the problem at hand.

Idea

Build a cascade of homeomorphic layers (i-RevNet), a network that can be fully inverted up to the final projection onto the classes, i.e. no information is discarded.

i-RevNet, 2018



Architecture	Injective	Bijective	Top-1 error	Parameters
ResNet	-	-	24.7	26M
RevNet	-	-	25.2	28M
<i>i</i> -RevNet (a)	yes	-	24.7	181M
<i>i</i> -RevNet (b)	yes	yes	26.7	29M

Flow-based variational dequantization

$$\log P(\mathbf{x}|\theta) \geq \int p(\epsilon) \log \left(\frac{p(\mathbf{x} + g(\epsilon, \mathbf{x}, \lambda))}{p(\epsilon) \cdot |\det \mathbf{J}_g|^{-1}} \right) d\epsilon.$$

Table 1. Unconditional image modeling results in bits/dim

Model family	Model	CIFAR10	ImageNet 32x32	ImageNet 64x64
Non-autoregressive	RealNVP (Dinh et al., 2016)	3.49	4.28	—
	Glow (Kingma & Dhariwal, 2018)	3.35	4.09	3.81
	IAF-VAE (Kingma et al., 2016)	3.11	—	—
	Flow++ (ours)	3.08	3.86	3.69
Autoregressive	Multiscale PixelCNN (Reed et al., 2017)	—	3.95	3.70
	PixelCNN (van den Oord et al., 2016b)	3.14	—	—
	PixelRNN (van den Oord et al., 2016b)	3.00	3.86	3.63
	Gated PixelCNN (van den Oord et al., 2016c)	3.03	3.83	3.57
	PixelCNN++ (Salimans et al., 2017)	2.92	—	—
	Image Transformer (Parmar et al., 2018)	2.90	3.77	—
	PixelSNAIL (Chen et al., 2017)	2.85	3.80	3.52

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

Discretization of continuous distribution

Model discretization through CDF

$$F(\mathbf{x}|\boldsymbol{\theta}) = \int_{-\infty}^{\mathbf{x}} p(\mathbf{x}'|\boldsymbol{\theta}) d\mathbf{x}'; \quad P(\mathbf{y}|\boldsymbol{\theta}) = F(\mathbf{y} + 0.5|\boldsymbol{\theta}) - F(\mathbf{y} - 0.5|\boldsymbol{\theta})$$

Mixture of logistic distributions

$$p(x|\mu, s) = \frac{\exp^{-(x-\mu)/s}}{s(1 + \exp^{-(x-\mu)/s})^2}; \quad p(x|\boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{s}) = \sum_{k=1}^K \pi_k p(x|\mu_k, s_k).$$

PixelCNN++

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{j=1}^m p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta}); \quad p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta}) = \sum_{k=1}^K \pi_k p(x|\mu_k, s_k).$$

Here, $\pi_k = \pi_{k,\boldsymbol{\theta}}(\mathbf{x}_{1:j-1})$, $\mu_k = \mu_{k,\boldsymbol{\theta}}(\mathbf{x}_{1:j-1})$, $s_k = s_{k,\boldsymbol{\theta}}(\mathbf{x}_{1:j-1})$.

For the pixel edge cases of 0, replace $y - 0.5$ by $-\infty$, and for 255 replace $y + 0.5$ by $+\infty$.

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

Flows intuition

Residual Flows (planar flows)

Autoregressive flows

Inverse gaussian autoregressive flows

Parallel WaveNet

RevNet, i-RevNet

Discretization of continuous distribution

Dequantization of discrete data

5. ELBO surgery

6. Disentanglement

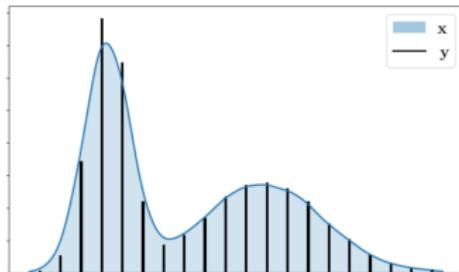
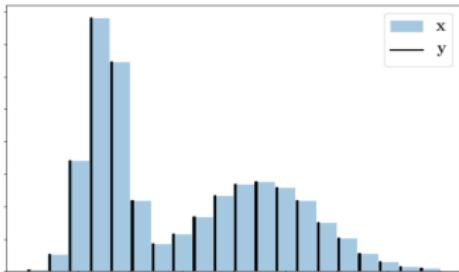
7. GANs

8. FFJORD

9. Quantized latents

10. Score matching

Variational dequantization



- ▶ $p(x|\theta)$ assign uniform density to unit hypercubes $\mathbf{y} + U[0, 1]$ (left fig).
- ▶ Smooth dequantization is more natural (right fig).
- ▶ Neural network density models are smooth function approximators.

Introduce variational dequantization noise distribution $q(\mathbf{u}|\mathbf{y})$, which tells what kind of noise we have to add to our discrete data. Treat it as an approximate posterior as in VAE model.

Variational dequantization

Variational lower bound

$$\begin{aligned}\log P(\mathbf{y}|\theta) &= \left[\log \int q(\mathbf{u}|\mathbf{y}) \frac{p(\mathbf{y} + \mathbf{u}|\theta)}{q(\mathbf{u}|\mathbf{y})} d\mathbf{u} \right] \geq \\ &\geq \int q(\mathbf{u}|\mathbf{y}) \log \frac{p(\mathbf{y} + \mathbf{u}|\theta)}{q(\mathbf{u}|\mathbf{y})} d\mathbf{u} = \mathcal{L}(q, \theta).\end{aligned}$$

Uniform dequantization is a special case of variational dequantization ($q(\mathbf{u}|\mathbf{y}) = U[0, 1]$).

Flow++: flow-based variational dequantization

Let $\mathbf{u} = g_\lambda(\epsilon, \mathbf{y})$ is a flow model with base distribution $\epsilon \sim p(\epsilon)$:

$$q(\mathbf{u}|\mathbf{y}) = p(f_\lambda(\mathbf{u}, \mathbf{y})) \cdot \left| \det \frac{\partial f_\lambda(\mathbf{u}, \mathbf{y})}{\partial \mathbf{u}} \right|.$$

$$\log P(\mathbf{y}|\theta) \geq \mathcal{L}(\lambda, \theta) = \int p(\epsilon) \log \left(\frac{p(\mathbf{y} + g_\lambda(\epsilon, \mathbf{y})|\theta)}{p(\epsilon) \cdot |\det \mathbf{J}_g|^{-1}} \right) d\epsilon.$$

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
 - VAE limitations: prior distribution
 - VAE limitations: posterior distribution
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

ELBO interpretations

$$\log p(\mathbf{x}|\theta) = \mathcal{L}(q, \theta) + KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z}|\mathbf{x}, \theta)).$$

$$\mathcal{L}(q, \theta) = \int q(\mathbf{z}|\mathbf{x}, \phi) \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x}, \phi)} d\mathbf{z}.$$

- ▶ Evidence minus posterior KL

$$\mathcal{L}(q, \theta) = \log p(\mathbf{x}|\theta) - KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z}|\mathbf{x}, \theta)).$$

- ▶ Average negative energy plus entropy

$$\begin{aligned}\mathcal{L}(q, \theta) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} [\log p(\mathbf{x}, \mathbf{z}|\theta) - \log q(\mathbf{z}|\mathbf{x}, \phi)] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}, \mathbf{z}|\theta) + \mathbb{H}[q(\mathbf{z}|\mathbf{x}, \phi)].\end{aligned}$$

- ▶ Average reconstruction minus KL to prior

$$\begin{aligned}\mathcal{L}(q, \theta) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} [\log p(\mathbf{x}|\mathbf{z}, \theta) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x}, \phi)] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \theta) - KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z})).\end{aligned}$$

ELBO surgery, 2016

$$\mathcal{L}(q, \theta) = \int q(\mathbf{Z}|\mathbf{X}) \log \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z}|\mathbf{X})} d\mathbf{Z}.$$

ELBO interpretations

- ▶ Evidence minus posterior KL

$$\mathcal{L}(q, \theta) = \log p(\mathbf{X}|\theta) - KL(q(\mathbf{Z}|\mathbf{X})||p(\mathbf{Z}|\mathbf{X}, \theta)).$$

- ▶ Average negative energy plus entropy

$$\mathcal{L}(q, \theta) = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X})} p(\mathbf{X}, \mathbf{Z}|\theta) + \mathbb{H}[q(\mathbf{Z}|\mathbf{X})].$$

- ▶ Average term-by-term reconstruction minus KL to prior

$$\mathcal{L}(q, \theta) = \frac{1}{n} \sum_{i=1}^n [\mathbb{E}_{q(\mathbf{z}_i|\mathbf{x}_i)} \log p(\mathbf{x}_i|\mathbf{z}_i, \theta) - KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i))].$$

ELBO surgery, 2016

$$\mathcal{L}(q, \theta) = \frac{1}{n} \sum_{i=1}^n [\mathbb{E}_{q(\mathbf{z}_i|\mathbf{x}_i)} \log p(\mathbf{x}_i|\mathbf{z}_i, \theta) - KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i))].$$

Theorem

$$\frac{1}{n} \sum_{i=1}^n KL(q(\mathbf{z}_i|\mathbf{x}_i)||p(\mathbf{z}_i)) = KL(q(\mathbf{z})||p(\mathbf{z})) + \mathbb{I}_{q(i,\mathbf{z})}[i, \mathbf{z}],$$

where i is treated as random variable:

$$q(i, \mathbf{z}) = q(i)q(\mathbf{z}|i); \quad p(i, \mathbf{z}) = p(i)p(\mathbf{z}); \quad q(i) = p(i) = \frac{1}{n}; \quad q(\mathbf{z}|i) = q(\mathbf{z}|\mathbf{x}_i).$$

$$q(\mathbf{z}) = \sum_{i=1}^n q(i, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^n q(\mathbf{z}|\mathbf{x}_i); \quad \mathbb{I}_{q(i,\mathbf{z})}[i, \mathbf{z}] = \mathbb{E}_{q(i,\mathbf{z})} \log \frac{q(i, \mathbf{z})}{q(i)q(\mathbf{z})}.$$

ELBO surgery, 2016

Theorem

$$\frac{1}{n} \sum_{i=1}^n KL(q(\mathbf{z}_i | \mathbf{x}_i) || p(\mathbf{z}_i)) = KL(q(\mathbf{z}) || p(\mathbf{z})) + \mathbb{I}_{q(i, \mathbf{z})}[i, \mathbf{z}].$$

Proof

$$\begin{aligned}\frac{1}{n} \sum_{i=1}^n KL(q(\mathbf{z}_i | \mathbf{x}_i) || p(\mathbf{z}_i)) &= \sum_{i=1}^n \int q(i) q(\mathbf{z}|i) \log \frac{q(\mathbf{z}|i)}{p(\mathbf{z})} d\mathbf{z} = \\&= \sum_{i=1}^n \int q(i, \mathbf{z}) \log \frac{q(i, \mathbf{z})}{p(\mathbf{z})p(i)} d\mathbf{z} = \int \sum_{i=1}^n q(i, \mathbf{z}) \log \frac{q(\mathbf{z})q(i|\mathbf{z})}{p(\mathbf{z})p(i)} d\mathbf{z} = \\&= \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z} + \int \sum_{i=1}^n q(i|\mathbf{z})q(\mathbf{z}) \log \frac{q(i|\mathbf{z})}{p(i)} d\mathbf{z} = \\&= KL(q(\mathbf{z}) || p(\mathbf{z})) - \mathbb{E}_{q(\mathbf{z})} \mathbb{H}[q(i|\mathbf{z})] + \log n.\end{aligned}$$

ELBO surgery, 2016

Theorem

$$\frac{1}{n} \sum_{i=1}^n KL(q(\mathbf{z}_i | \mathbf{x}_i) || p(\mathbf{z}_i)) = KL(q(\mathbf{z}) || p(\mathbf{z})) + \mathbb{I}_{q(i, \mathbf{z})}[i, \mathbf{z}].$$

Proof (continued)

$$\frac{1}{n} \sum_{i=1}^n KL(q(\mathbf{z}_i | \mathbf{x}_i) || p(\mathbf{z}_i)) = KL(q(\mathbf{z}) || p(\mathbf{z})) - \mathbb{E}_{q(\mathbf{z})} \mathbb{H}[q(i|\mathbf{z})] + \log n$$

$$\begin{aligned}\mathbb{I}_{q(i, \mathbf{z})}[i, \mathbf{z}] &= \mathbb{E}_{q(i, \mathbf{z})} \log \frac{q(i, \mathbf{z})}{q(i)q(\mathbf{z})} = \mathbb{E}_{q(\mathbf{z})} \mathbb{E}_{q(i|\mathbf{z})} \log \frac{q(i|\mathbf{z})q(\mathbf{z})}{q(i)q(\mathbf{z})} = \\ &= \mathbb{E}_{q(\mathbf{z})} \mathbb{E}_{q(i|\mathbf{z})} \log \frac{q(i|\mathbf{z})}{q(i)} = -\mathbb{E}_{q(\mathbf{z})} \mathbb{H}[q(i|\mathbf{z})] + \log n.\end{aligned}$$

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
 - VAE limitations: prior distribution
 - VAE limitations: posterior distribution
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

Learnable VAE prior

Optimal prior

$$KL(q_{\text{agg}}(\mathbf{z}) || p(\mathbf{z})) = 0 \Leftrightarrow p(\mathbf{z}) = q_{\text{agg}}(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^n q(\mathbf{z} | \mathbf{x}_i).$$

Mixture of Gaussians

$$p(\mathbf{z} | \boldsymbol{\lambda}) = \sum_{k=1}^K w_k \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2), \quad \boldsymbol{\lambda} = \{w_k, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k\}_{k=1}^K.$$

Variational Mixture of posteriors (VampPrior)

$$p(\mathbf{z} | \boldsymbol{\lambda}) = \frac{1}{K} \sum_{k=1}^K q(\mathbf{z} | \mathbf{u}_k),$$

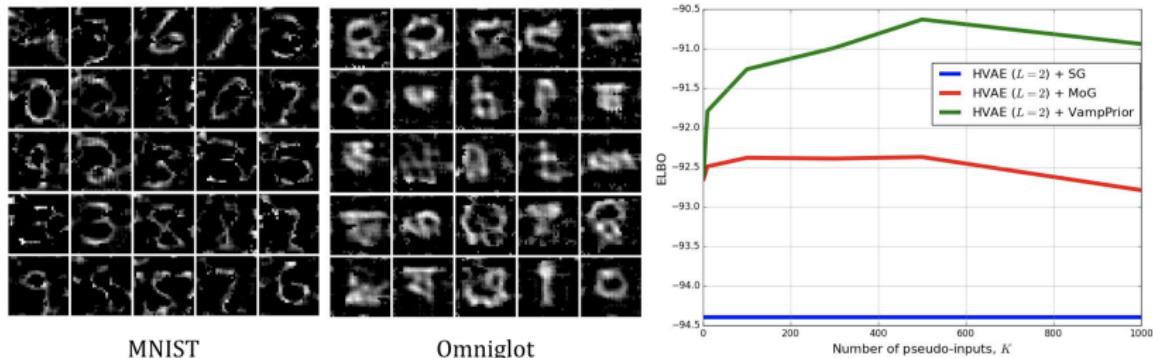
where $\boldsymbol{\lambda} = \{\mathbf{u}_1, \dots, \mathbf{u}_K\}$ are trainable pseudo-inputs.

- ▶ Multimodal \Rightarrow prevents over-regularization;.
- ▶ $K \ll n \Rightarrow$ prevents from potential overfitting + less expensive to train.

VampPrior

- ▶ Do we really need the multimodal prior?
- ▶ Is it beneficial to couple the prior with the variational posterior or the MoG prior is enough?

Results



Top row: generated images by PixelHVAE + VampPrior for chosen pseudo-input in the left top corner.

Bottom row: pseudo-inputs for different datasets.

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
 - VAE limitations: prior distribution
 - VAE limitations: posterior distribution
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

Normalizing Flows in VAE posterior

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(z|x, \phi)} [\log p(x|z, \theta) + \log p(z) - \log q(z|x, \phi)]$$

Let apply NF to VAE posterior!

Assume $q(z|x, \phi)$ (VAE encoder) is a base distribution for a flow model.

Flow model in latent space

$$\log q(z^*|x, \phi, \lambda) = \log q(z|x, \phi) + \log \left| \det \left(\frac{dz}{dz^*} \right) \right|$$

$$z^* = f(z, \lambda) = g^{-1}(z, \lambda)$$

- ▶ Encoder outputs base distribution $q(z|x, \phi)$.
- ▶ Flow model $z^* = f(z, \lambda)$ transforms the base distribution $q(z|x, \phi)$ to the distribution $q(z^*|x, \phi, \lambda)$.
- ▶ Distribution $q(z^*|x, \phi, \lambda)$ is used as a variational distribution for ELBO maximization.
- ▶ Here ϕ – encoder parameters, λ – flow parameters.

Normalizing Flows in VAE posterior

ELBO with flow-based VAE posterior

$$\begin{aligned}\mathcal{L}(\phi, \theta, \lambda) &= \mathbb{E}_{q(z^*|x, \phi, \lambda)} \log p(x|z^*, \theta) - KL(q(z^*|x, \phi, \lambda)||p(z^*)) = \\ &= \mathbb{E}_{q(z^*|x, \phi, \lambda)} [\log p(x|z^*, \theta) + \log p(z^*) - \log q(z^*|x, \phi, \lambda)] = \\ &= \mathbb{E}_{q(z^*|x, \phi, \lambda)} \left[\log p(x|z^*, \theta) + \log p(z^*) - \right. \\ &\quad \left. - (\log q(g(z^*, \lambda)|x, \phi) + \log |\det(\mathbf{J}_g)|) \right].\end{aligned}$$

KL term in ELBO is **reverse** KL divergence with respect to λ .

- ▶ RealNVP with coupling layers.
- ▶ Inverse autoregressive flow (slow $f(z, \lambda)$, fast $g(z^*, \lambda)$).
- ▶ Is it OK to use AF for VAE posterior?

Normalizing Flows in VAE posterior

Theorem (flow KL duality, Lecture 5)

$$KL(\pi(\mathbf{x}) || p(\mathbf{x}|\theta)) = KL(p(\mathbf{z}|\theta) || p(\mathbf{z})).$$

ELBO with flow-based VAE posterior

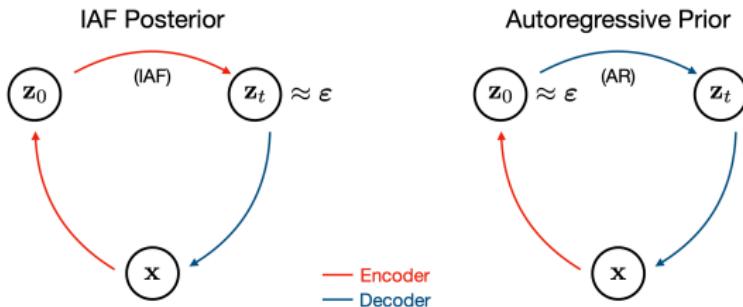
$$\begin{aligned}\mathcal{L}(\phi, \theta, \lambda) &= \mathbb{E}_{q(\mathbf{z}^*|\mathbf{x}, \phi, \lambda)} \log p(\mathbf{x}|\mathbf{z}^*, \theta) - KL(q(\mathbf{z}^*|\mathbf{x}, \phi, \lambda) || p(\mathbf{z}^*)) \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{f}(\mathbf{z}, \lambda), \theta) - KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z}|\lambda)).\end{aligned}$$

(Here we use Flow KL duality theorem and LOTUS trick.)

ELBO with flow-based VAE prior

$$\mathcal{L}(\phi, \theta, \lambda) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \theta) - KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z}|\lambda))$$

Flows-based VAE prior vs posterior



- ▶ Flow-based posterior decoder path: $\mathbf{z} \sim p(\mathbf{z})$, $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \theta)$.
- ▶ Flow-based prior decoder path: $\mathbf{z}^* \sim p(\mathbf{z}^*)$, $\mathbf{z} = f(\mathbf{z}^*, \lambda)$, $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \theta)$.

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
 - InfoGAN
 - β -VAE
 - DIP-VAE
 - FactorVAE
 - Challenging Disentanglement Assumptions
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
 - InfoGAN
 - β -VAE
 - DIP-VAE
 - FactorVAE
 - Challenging Disentanglement Assumptions
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

InfoGAN

GAN objective

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

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

Latent vector \mathbf{z} is not imposed to be disentangled.

InfoGAN decomposes input vector:

- ▶ \mathbf{z} – incompressible noise;
- ▶ \mathbf{c} – structured latent code $p(\mathbf{c}) = \prod_{j=1}^d p(c_j)$.

Information-theoretic regularization

$$\max I(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$

Information in the latent code \mathbf{c} should not be lost in the
generation process.

Chen X. et al. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, 2016

InfoGAN

Objective

$$\min_G \max_D V(G, D) - \lambda I(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$

Variational Information Maximization

$$\begin{aligned} I(\mathbf{c}, G(\mathbf{z}, \mathbf{c})) &= H(\mathbf{c}) - H(\mathbf{c}|G(\mathbf{z}, \mathbf{c})) = \\ &= H(\mathbf{c}) + \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} [\mathbb{E}_{\mathbf{c}' \sim p(\mathbf{c}|\mathbf{x})} \log p(\mathbf{c}'|\mathbf{x})] = \\ &= H(\mathbf{c}) + \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} KL(p(\mathbf{c}'|\mathbf{x}) || q(\mathbf{z}'|\mathbf{x})) + \\ &\quad + \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} \mathbb{E}_{\mathbf{c}' \sim p(\mathbf{c}|\mathbf{x})} \log q(\mathbf{c}'|\mathbf{x}) \geq \\ &\geq H(\mathbf{c}) + \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} \mathbb{E}_{\mathbf{c}' \sim p(\mathbf{c}|\mathbf{x})} \log q(\mathbf{c}'|\mathbf{x}) = \\ &\quad H(\mathbf{c}) + \mathbb{E}_{\mathbf{c} \sim p(\mathbf{c})} \mathbb{E}_{\mathbf{x} \sim G(\mathbf{z}, \mathbf{c})} \log q(\mathbf{c}|\mathbf{x}) \end{aligned}$$

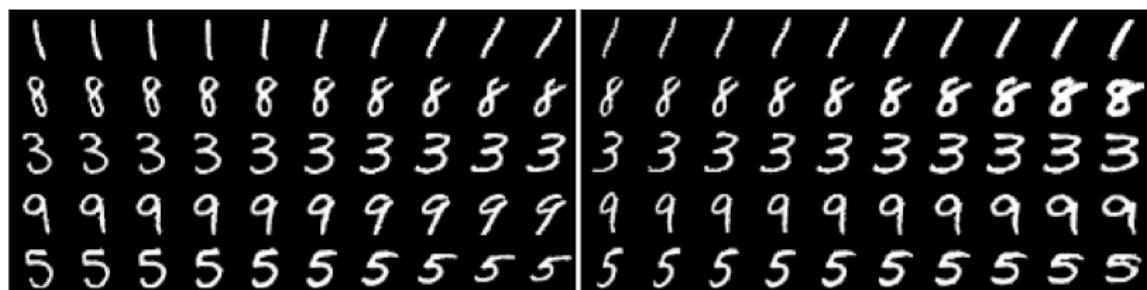
InfoGAN

Latent codes on MNIST



(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)



(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

InfoGAN

Latent codes on 3D Faces



(a) Azimuth (pose)

(b) Elevation



(c) Lighting

(d) Wide or Narrow

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
 - InfoGAN
 - β -VAE
 - DIP-VAE
 - FactorVAE
 - Challenging Disentanglement Assumptions
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

Disentangled representations

Representation learning is looking for an interpretable representation of the independent data generative factors.

Disentanglement informal definition

Every single latent unit are sensitive to changes in a single generative factor, while being invariant to changes in other factors.

ELBO objective

$$\mathcal{L}(q, \theta, \beta) = \mathbb{E}_{q(z|x)} \log p(x|z, \theta) - \beta \cdot KL(q(z|x)||p(z)).$$

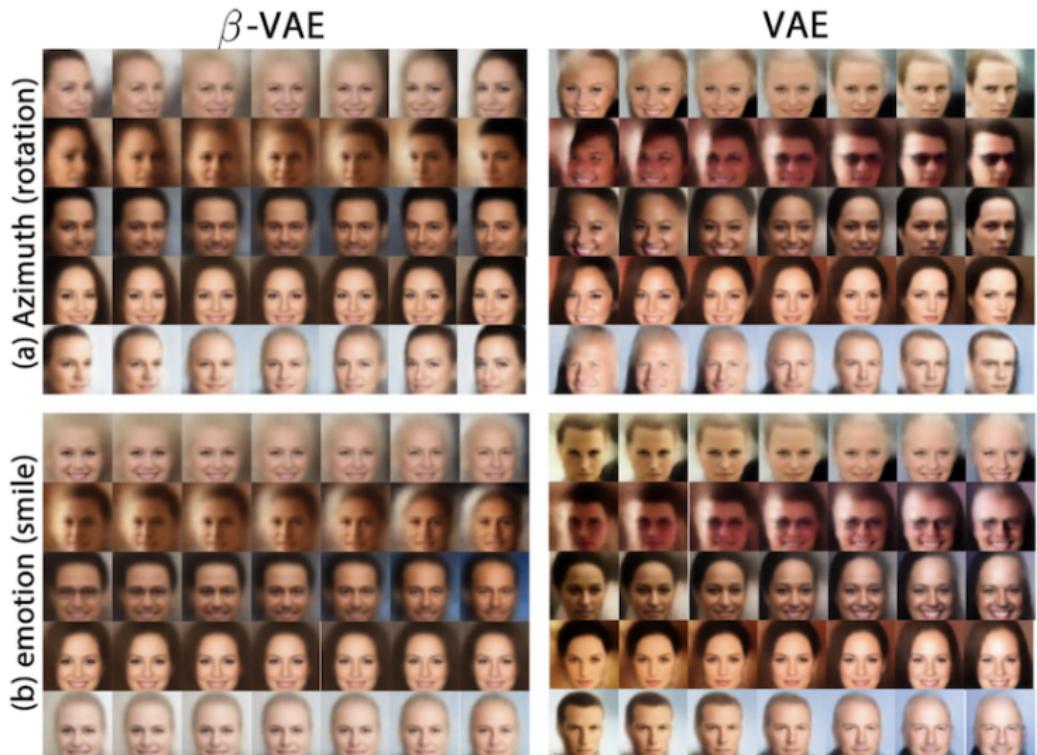
What do we get at $\beta = 1$?

Constrained optimization

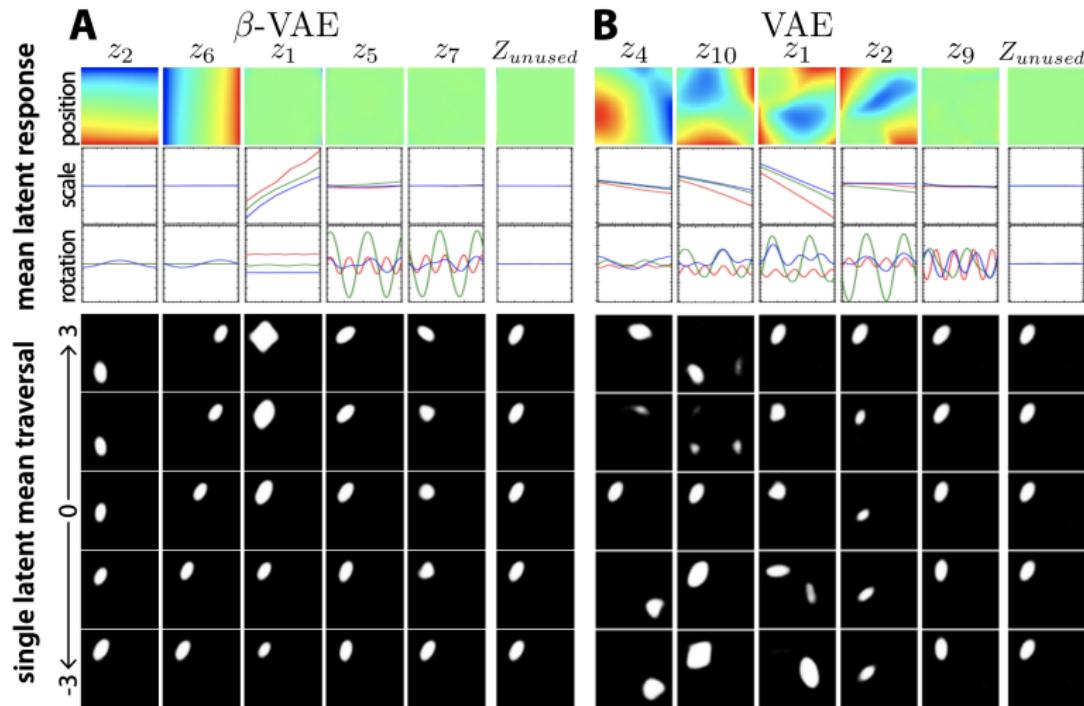
$$\max_{q, \theta} \mathbb{E}_{q(z|x)} \log p(x|z, \theta), \quad \text{subject to } KL(q(z|x)||p(z)) < \epsilon.$$

Note: It leads to poorer reconstructions and a loss of high frequency details.

β -VAE samples



β -VAE analysis



β -VAE

ELBO

$$\mathcal{L}(q, \theta, \beta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}, \theta) - \beta \cdot KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})).$$

ELBO surgery

$$\frac{1}{n} \sum_{i=1}^n \mathcal{L}_i(q, \theta, \beta) = \underbrace{\frac{1}{n} \sum_{i=1}^n \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i)} \log p(\mathbf{x}_i|\mathbf{z}, \theta)}_{\text{Reconstruction loss}} - \underbrace{\beta \cdot \mathbb{I}_q[\mathbf{x}, \mathbf{z}]}_{\text{MI}} - \underbrace{\beta \cdot KL(q_{\text{agg}}(\mathbf{z})||p(\mathbf{z}))}_{\text{Marginal KL}}$$

Minimization of MI

- ▶ It is not necessary and not desirable for disentanglement.
- ▶ It hurts reconstruction.

β -VAE

Disentangling metric

1. Generate two sets of objects

$$\mathbf{x}_{li} \sim \text{Sim}(\mathbf{v}_{li}, \mathbf{w}_{li}); \quad \mathbf{x}_{lj} \sim \text{Sim}(\mathbf{v}_{lj}, \mathbf{w}_{lj}); \quad y_{ij} \sim U[1, d].$$

$$\mathbf{v}_{li} \sim p(\mathbf{v}); \quad \mathbf{v}_{lj} \sim p(\mathbf{v}) ([v_{li}]_y = [v_{lj}]_y); \quad \mathbf{w}_{li}, \mathbf{w}_{lj} \sim p(\mathbf{w}).$$

2. Find representations

$$q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x})|\sigma^2(\mathbf{x})); \quad \mathbf{z}_{li} = \mu(\mathbf{x}_{li}); \quad \mathbf{z}_{lj} = \mu(\mathbf{x}_{lj}).$$

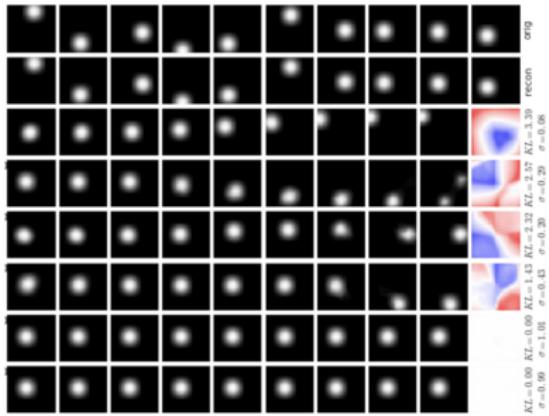
3. Use accuracy of classifier $p(y|\mathbf{z}_{\text{diff}})$ with a low VC-dimension as metric of disentanglement

$$\mathbf{z}_{\text{diff}} = \frac{1}{L} \sum_{l=1}^L |\mathbf{z}_{li} - \mathbf{z}_{lj}|.$$

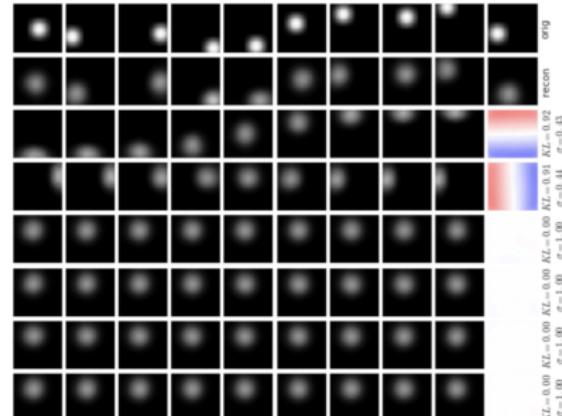
β -VAE

- ▶ **Top row:** original images.
 - ▶ **Second row:** the corresponding reconstructions.
 - ▶ **Remaining rows:** latent traversals ordered by KL divergence with the prior.
 - ▶ **Heatmaps:** latent activations for each 2D position.

$$\beta = 1$$



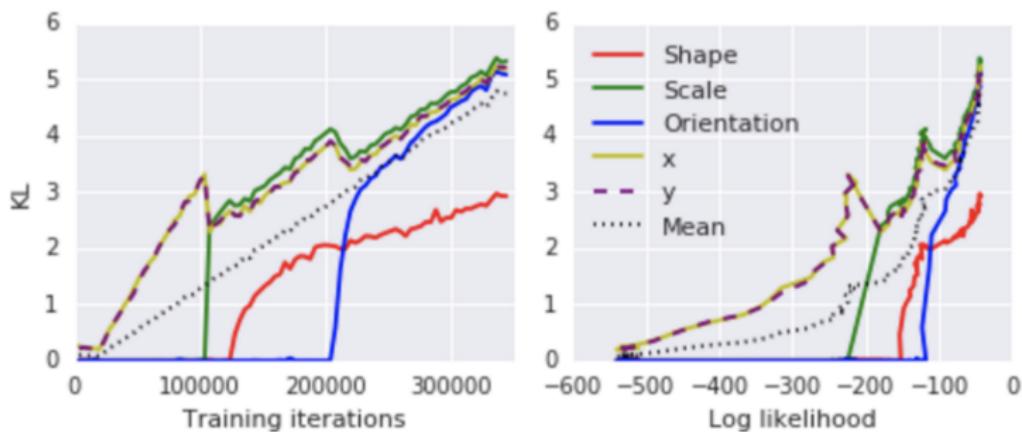
$$\beta = 150$$



β -VAE

Controlled encoding capacity

$$\mathcal{L}(q, \theta, \beta) = \mathbb{E}_{q(z|x)} \log p(x|z, \theta) - [KL(q(z|x)||p(z)) - C].$$

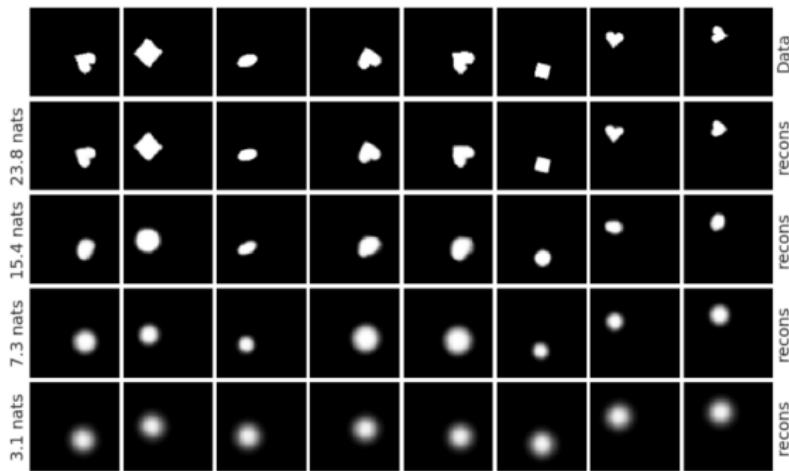


The early capacity is allocated to positional latents only, followed by a scale latent, then shape and orientation latents.

β -VAE

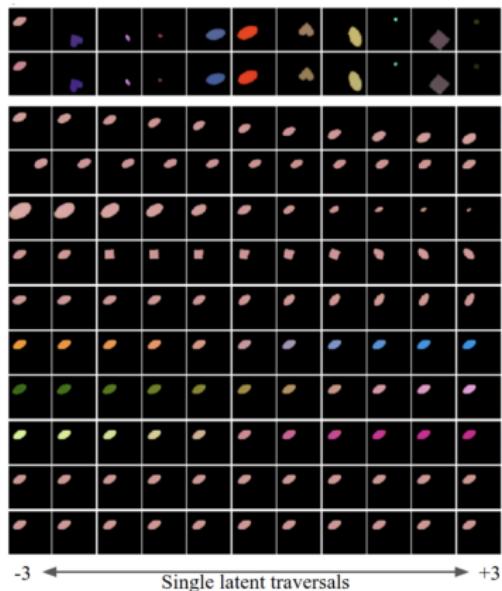
Controlled encoding capacity

$$\mathcal{L}(q, \theta, \beta) = \mathbb{E}_{q(z|x)} \log p(x|z, \theta) - |KL(q(z|x)||p(z)) - C|.$$

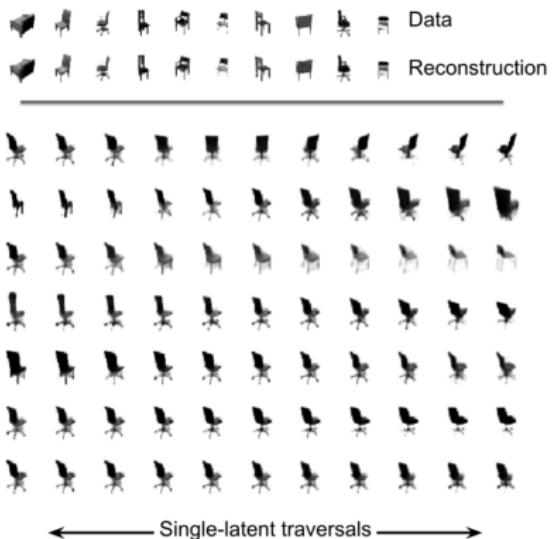


β -VAE

(a) Coloured dSprites



(b) 3D Chairs



Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
 - InfoGAN
 - β -VAE
 - DIP-VAE
 - FactorVAE
 - Challenging Disentanglement Assumptions
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

DIP-VAE: disentangled posterior

Disentangled aggregated variational posterior

$$q_{\text{agg}}(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^n q(\mathbf{z}|\mathbf{x}) = \prod_{j=1}^d q_{\text{agg}}(z_j)$$

DIP-VAE objective

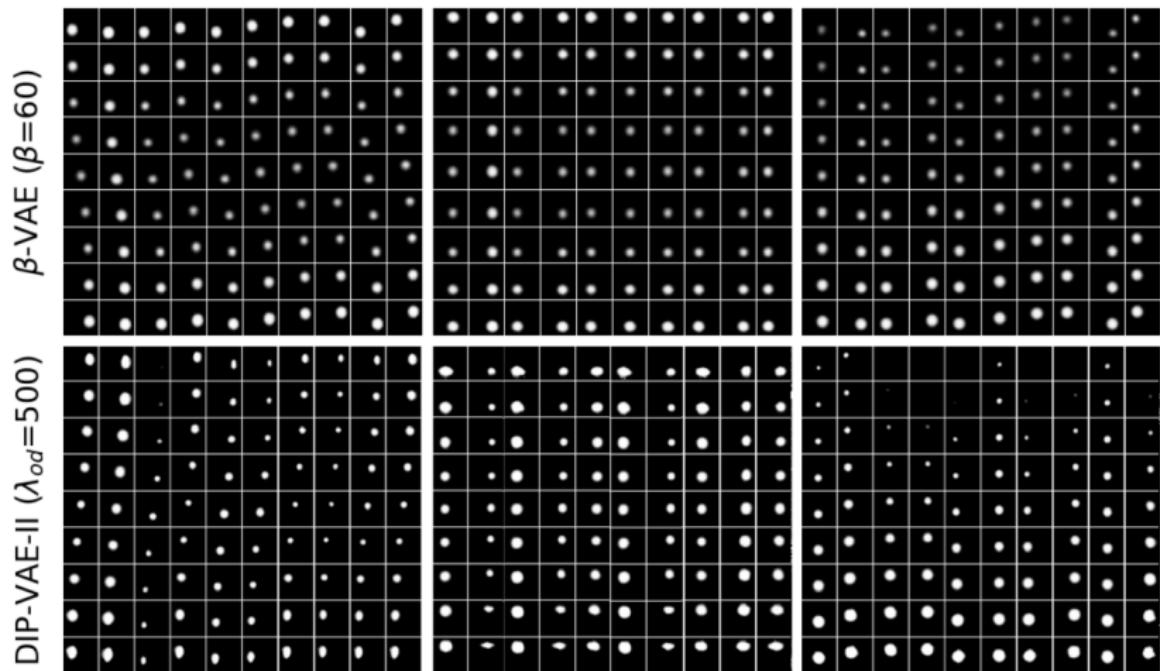
$$\begin{aligned}\mathcal{L}_{\text{DIP}}(q, \theta) &= \frac{1}{n} \sum_{i=1}^n \mathcal{L}_i(q, \theta) - \lambda \cdot KL(q_{\text{agg}}(\mathbf{z}) || p(\mathbf{z})) = \\ &= \underbrace{\frac{1}{n} \sum_{i=1}^n [\mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i)} \log p(\mathbf{x}_i|\mathbf{z}, \theta)]}_{\text{Reconstruction loss}} - \underbrace{\mathbb{I}_q[\mathbf{x}, \mathbf{z}]}_{\text{MI}} - \underbrace{(1 + \lambda) \cdot KL(q_{\text{agg}}(\mathbf{z}) || p(\mathbf{z}))}_{\text{Marginal KL}}\end{aligned}$$

Marginal KL term is intractable. \Rightarrow Let match the moments of $q_{\text{agg}}(\mathbf{z})$ and $p(\mathbf{z})$:

$$\text{cov}_{q_{\text{agg}}(\mathbf{z})}(\mathbf{z}) = \mathbb{E}_{q_{\text{agg}}(\mathbf{z})} \left[(\mathbf{z} - \mathbb{E}_{q_{\text{agg}}(\mathbf{z})}(\mathbf{z}))(\mathbf{z} - \mathbb{E}_{q_{\text{agg}}(\mathbf{z})}(\mathbf{z}))^T \right].$$

DIP-VAE: analysis

Reconstructions become better.



Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. **Disentanglement**
 - InfoGAN
 - β -VAE
 - DIP-VAE
 - FactorVAE**
 - Challenging Disentanglement Assumptions
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

FactorVAE

Disentangled aggregated variational posterior

$$q(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^n q(\mathbf{z}|\mathbf{x}) = \prod_{j=1}^d q(z_j)$$

Total correlation regularizer

$$\min KL(q(\mathbf{z}) || \prod_{j=1}^d q(z_j))$$

FactorVAE objective

$$\min_{\phi, \theta} \mathcal{L}(\phi, \theta) - \gamma \cdot KL(q(\mathbf{z}) || \prod_{j=1}^d q(z_j))$$

- ▶ The last term is intractable.
- ▶ FactorVAE uses density ratio trick for estimation.

FactorVAE

Consider two distributions $q_1(\mathbf{x})$, $q_2(\mathbf{x})$ and probabilistic model

$$p(\mathbf{x}|y) = \begin{cases} q_1(\mathbf{x}), & \text{if } y = 1, \\ q_2(\mathbf{x}), & \text{if } y = 0, \end{cases} \quad y \sim \text{Bern}(0.5).$$

Density ratio trick

$$\begin{aligned} \frac{q_1(\mathbf{x})}{q_2(\mathbf{x})} &= \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} = \frac{p(y=1|\mathbf{x})p(\mathbf{x})}{p(y=1)} \Big/ \frac{p(y=0|\mathbf{x})p(\mathbf{x})}{p(y=0)} = \\ &= \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{1 - p(y=1|\mathbf{x})} = \frac{D(\mathbf{x})}{1 - D(\mathbf{x})} \end{aligned}$$

Here $D(\mathbf{x})$ could be treated as a discriminator a model the output of which is a probability that \mathbf{x} is a sample from $q_1(\mathbf{x})$ rather than from $q_2(\mathbf{x})$.

FactorVAE

FactorVAE objective

$$\min_{\theta, \phi} \text{ELBO}(\theta, \phi) - \gamma \cdot KL(q(\mathbf{z}) || \prod_{j=1}^d q(z_j))$$

Total correlation regularizer

$$\begin{aligned} KL(q(\mathbf{z}) || \prod_{j=1}^d q(z_j)) &= KL(q(\mathbf{z}) || \bar{q}(\mathbf{z})) = \\ &= \mathbb{E}_{q(\mathbf{z})} \log \frac{q(\mathbf{z})}{\bar{q}(\mathbf{z})} \approx \mathbb{E}_{q(\mathbf{z})} \log \frac{D(\mathbf{z})}{1 - D(\mathbf{z})} \end{aligned}$$

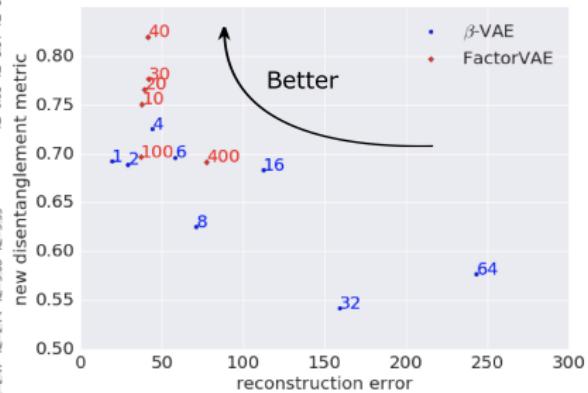
VAE and GAN are trained simultaneously.

FactorVAE

β -VAE ($\beta = 8$)



FactorVAE ($\gamma = 10$)



Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
 - InfoGAN
 - β -VAE
 - DIP-VAE
 - FactorVAE
 - Challenging Disentanglement Assumptions**
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

Challenging disentanglement assumptions

Theorem

Let \mathbf{z} has density $p(\mathbf{z}) = \prod_{i=1}^d p(z_i)$. Then, there exists an **infinite** family of bijective functions $f : \text{supp}(\mathbf{z}) \rightarrow \text{supp}(\mathbf{z})$:

- ▶ $\frac{\partial f_i(\mathbf{z})}{\partial z_j} \neq 0$ for all i and j (\mathbf{z} and $f(\mathbf{z})$ are completely entangled);
- ▶ $P(\mathbf{z} \leq \mathbf{u}) = P(f(\mathbf{z}) \leq \mathbf{u})$ for all $\mathbf{u} \in \text{supp}(\mathbf{z})$.

Consider a generative model with disentangled representation \mathbf{z} .

- ▶ $\exists \hat{\mathbf{z}} = f(\mathbf{z})$ where $\hat{\mathbf{z}}$ is completely entangled with respect to \mathbf{z} .
- ▶ The disentanglement method cannot distinguish between the two equivalent generative models:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int p(\mathbf{x}|\hat{\mathbf{z}})p(\hat{\mathbf{z}})d\hat{\mathbf{z}}.$$

Theorem claims that unsupervised disentanglement learning is impossible for arbitrary generative models with a factorized prior.

Challenging Disentanglement Assumptions

Proof (1)

1. Consider the function $g : \text{supp}(\mathbf{z}) \rightarrow [0, 1]^d$:

$$g_i(\mathbf{u}) = P(z_i \leq u_i), \quad i = 1, \dots, d.$$

- ▶ g is bijective (since $p(\mathbf{z}) = \prod_{i=1}^d p(z_i)$).
 - ▶ $\frac{\partial g_i(\mathbf{u})}{\partial u_i} \neq 0$, for all i and $\frac{\partial g_i(\mathbf{u})}{\partial u_j} = 0$ for all $i \neq j$.
 - ▶ $g(\mathbf{z})$ is an independent d -dimensional uniform distribution.
2. Consider $h : (0, 1]^d \rightarrow \mathbb{R}^d$

$$h_i(\mathbf{u}) = \psi^{-1}(u_i), \quad i = 1, \dots, d.$$

Here ψ denotes the CDF of a standard normal distribution.

- ▶ h is bijective.
- ▶ $\frac{\partial h_i(\mathbf{u})}{\partial u_i} \neq 0$, for all i and $\frac{\partial h_i(\mathbf{u})}{\partial u_j} = 0$ for all $i \neq j$.
- ▶ $h(g(\mathbf{z}))$ is a d -dimensional standard normal distribution.

Challenging Disentanglement Assumptions

Proof (2)

Let $\mathbf{A} \in \mathbb{R}^{d \times d}$ be an arbitrary orthogonal matrix with $A_{ij} \neq 0$ for all i, j . The family of such matrices is infinite.

- ▶ \mathbf{A} is orthogonal, it is invertible and thus defines a bijective linear operator.
- ▶ $\mathbf{A}h(g(\mathbf{z})) \in \mathbb{R}^d$ is hence an independent, multivariate standard normal distribution.
- ▶ $h^{-1}(\mathbf{A}h(g(\mathbf{z}))) \in \mathbb{R}^d$ is an independent d -dimensional uniform distribution.

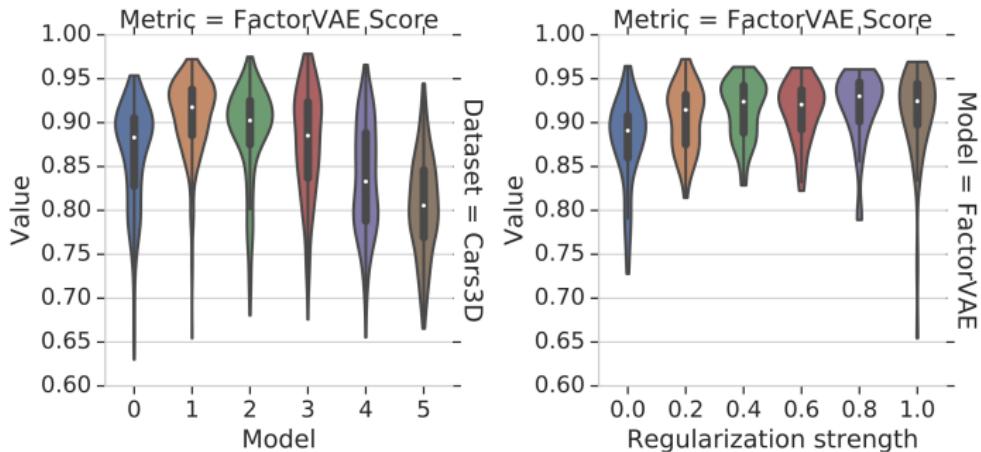
Define $f : \text{supp}(\mathbf{z}) \rightarrow \text{supp}(\mathbf{z})$:

$$f(\mathbf{u}) = g^{-1}(h^{-1}(\mathbf{A}h(g(\mathbf{z}))).$$

By definition $f(\mathbf{z})$ has the same marginal distribution as \mathbf{z} :

$$P(\mathbf{z} \leq \mathbf{u}) = P(f(\mathbf{z}) \leq \mathbf{u}) \text{ and } \frac{\partial f_i(\mathbf{z})}{\partial z_j} \neq 0.$$

Challenging disentanglement assumptions



	Dataset = Noisy-dSprites					
BetaVAE Score (A)	100	80	44	41	46	37
FactorVAE Score (B)	80	100	49	52	25	38
MIG (C)	44	49	100	76	6	42
DCI Disentanglement (D)	41	52	76	100	-8	38
Modularity (E)	46	25	6	-8	100	13
SAP (F)	37	38	42	38	13	100
	(A)	(B)	(C)	(D)	(E)	(F)

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
 - DCGAN
 - Vanishing gradients
 - Improved techniques for training GANs
 - Adversarial variational Bayes
 - WGAN
 - Spectral Normalization GAN
 - Evolution of GANs
 - Evaluation of likelihood-free models
8. FFJORD
9. Quantized latents
10. Score matching

Likelihood-free learning

- ▶ Likelihood is not a perfect quality measure for generative model.
- ▶ Likelihood could be intractable.

Where did we start

We would like to approximate true data distribution $\pi(\mathbf{x})$. Instead of searching true $\pi(\mathbf{x})$ over all probability distributions, learn function approximation $p(\mathbf{x}|\theta) \approx \pi(\mathbf{x})$.

Imagine we have two sets of samples

- ▶ $\mathcal{S}_1 = \{\mathbf{x}_i\}_{i=1}^{n_1} \sim \pi(\mathbf{x})$ – real samples;
- ▶ $\mathcal{S}_2 = \{\mathbf{x}_i\}_{i=1}^{n_2} \sim p(\mathbf{x}|\theta)$ – generated (or fake) samples.

Two sample test

$$H_0 : \pi(\mathbf{x}) = p(\mathbf{x}|\theta), \quad H_1 : \pi(\mathbf{x}) \neq p(\mathbf{x}|\theta)$$

Define test statistic $T(\mathcal{S}_1, \mathcal{S}_2)$. The test statistic is likelihood free.
If $T(\mathcal{S}_1, \mathcal{S}_2) < \alpha$, then accept H_0 , else reject it.

Likelihood-free learning

Two sample test

$$H_0 : \pi(\mathbf{x}) = p(\mathbf{x}|\theta), \quad H_1 : \pi(\mathbf{x}) \neq p(\mathbf{x}|\theta)$$

Desired behaviour

- ▶ $p(\mathbf{x}|\theta)$ minimizes the value of test statistic $T(\mathcal{S}_1, \mathcal{S}_2)$.
- ▶ It is hard to find an appropriate test statistic in high dimensions. $T(\mathcal{S}_1, \mathcal{S}_2)$ could be learnable.

Generative adversarial network (GAN) objective

- ▶ **Generator:** generative model $\mathbf{x} = G(\mathbf{z})$, which makes generated sample more realistic. Here $\mathbf{z} \sim p(\mathbf{z})$, $\mathbf{x} \sim p(\mathbf{x}|\theta)$.
- ▶ **Discriminator:** a classifier $D(\mathbf{x}) \in [0, 1]$, which distinguishes real samples from generated samples.

$$\min_G \max_D \left[\mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\mathbf{x}|\theta)} \log(1 - D(\mathbf{x})) \right]$$

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

DCGAN

Vanishing gradients

Improved techniques for training GANs

Adversarial variational Bayes

WGAN

Spectral Normalization GAN

Evolution of GANs

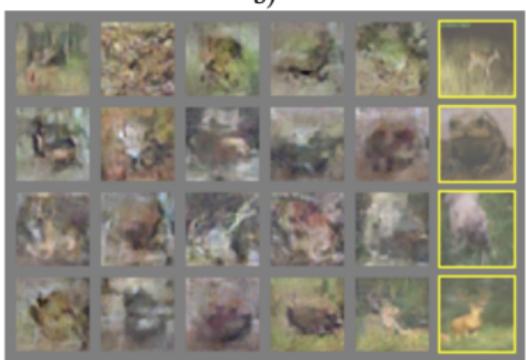
Evaluation of likelihood-free models

8. FFJORD

9. Quantized latents

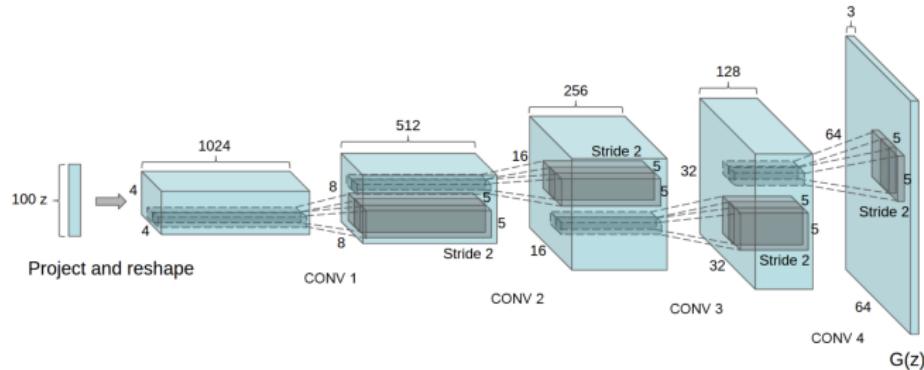
10. Score matching

Vanilla GAN results



Deep Convolutional GAN

Architecture

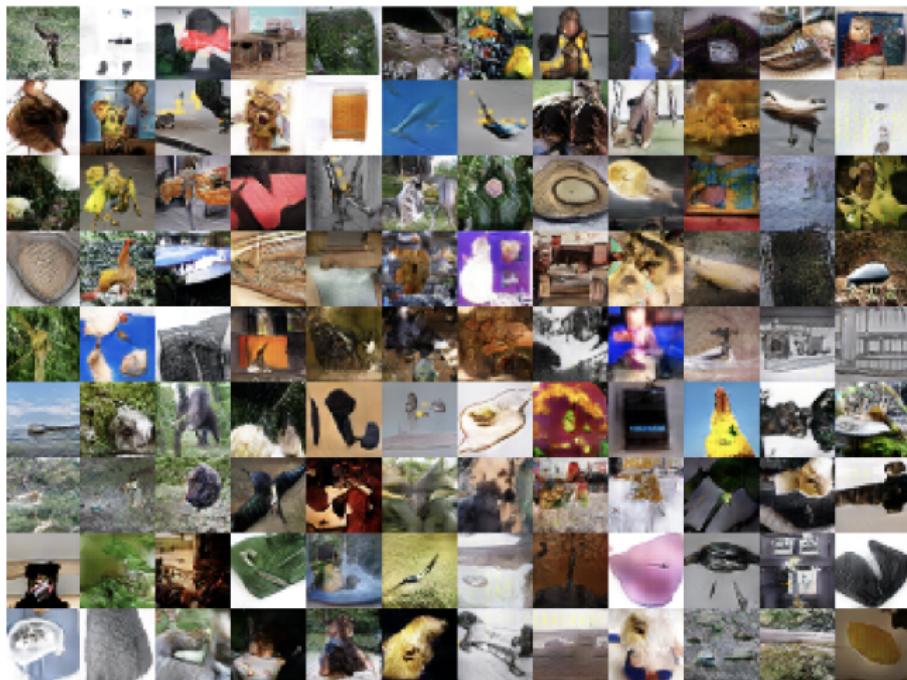


- ▶ Mean-pooling instead of max-pooling.
- ▶ Transposed convolutions in the generator for upsampling.
- ▶ Downsample with strided convolutions and average pooling.
- ▶ ReLU for generator, Leaky-ReLU (0.2) for discriminator.
- ▶ Output nonlinearity: tanh for Generator, sigmoid for discriminator.
- ▶ Batch Normalization used to prevent mode collapse (not applied at the output of G and input of D).
- ▶ Adam: small LR = 2e-4; small momentum: 0.5, batch-size: 128.

Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015

Deep Convolutional GAN

ImageNet samples



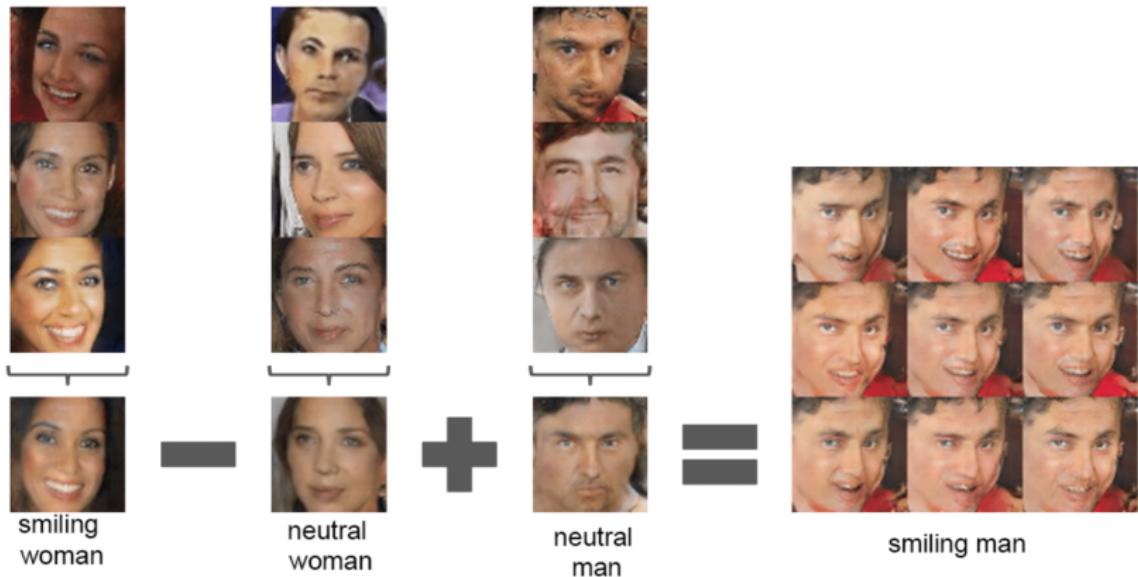
Deep Convolutional GAN

Smooth interpolations



Deep Convolutional GAN

Vector arithmetic



Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

DCGAN

Vanishing gradients

Improved techniques for training GANs

Adversarial variational Bayes

WGAN

Spectral Normalization GAN

Evolution of GANs

Evaluation of likelihood-free models

8. FFJORD

9. Quantized latents

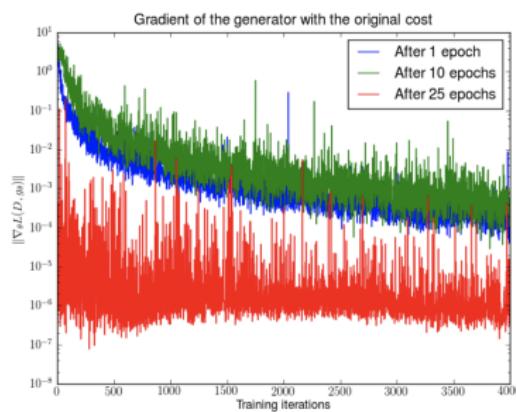
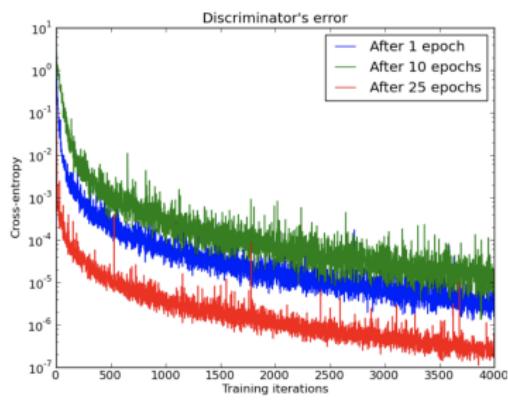
10. Score matching

Vanishing gradients

Objective

$$\min_{\theta} \max_{\phi} [\mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}, \phi) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z}, \theta), \phi))]$$

Early in learning, G is poor, D can reject samples with high confidence. In this case, $\log(1 - D(G(\mathbf{z}, \theta), \phi))$ saturates.



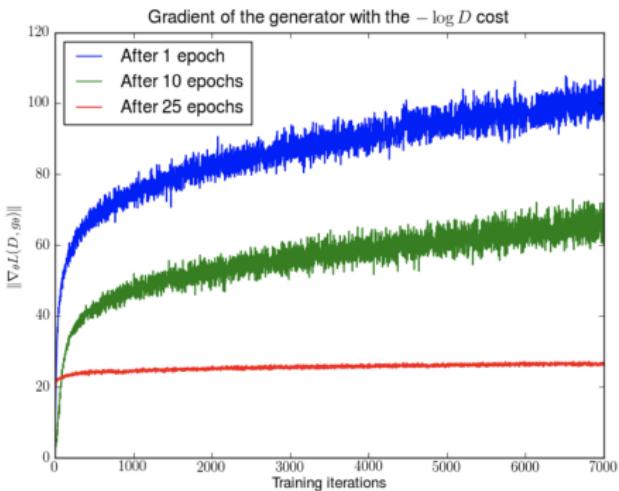
Vanishing gradients

Objective

$$\min_{\theta} \max_{\phi} \left[\mathbb{E}_{\pi(\mathbf{x})} \log D(\mathbf{x}, \phi) + \mathbb{E}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z}, \theta), \phi)) \right]$$

Non-saturating GAN

- ▶ Maximize $\log D(G(z))$ instead of minimizing $\log(1 - D(G(z)))$.
- ▶ Gradients are getting much stronger, but the training is unstable (with increasing mean and variance).



Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
 - DCGAN
 - Vanishing gradients
 - Improved techniques for training GANs**
 - Adversarial variational Bayes
 - WGAN
 - Spectral Normalization GAN
 - Evolution of GANs
 - Evaluation of likelihood-free models
8. FFJORD
9. Quantized latents
10. Score matching

Improved techniques for training GANs

- ▶ Feature matching

$$\mathcal{L}_G = \|\mathbb{E}_{\pi(\mathbf{x})} \mathbf{d}(\mathbf{x}) - \mathbb{E}_{p(\mathbf{z})} \mathbf{d}(G(\mathbf{z}))\|_2^2$$

Here $\mathbf{d}(\mathbf{x})$ – intermediate layer of discriminator. Matching the learned discriminator statistics instead of the output of the discriminator. Helps to avoid the vanishing gradients for sufficiently good discriminator.

- ▶ Historical averaging adds extra loss term for generator and discriminator losses

$$\|\boldsymbol{\theta} - \frac{1}{T} \sum_{t=1}^T \boldsymbol{\theta}_t\|_2^2.$$

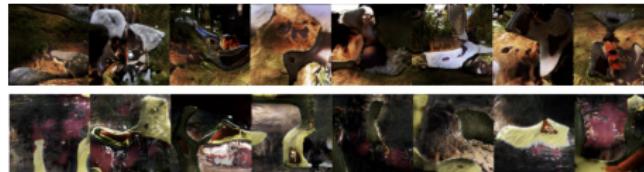
Here $\boldsymbol{\theta}_t$ – value of parameters at the previous step t . It allows to stabilize training procedure.

Improved techniques for training GANs

- ▶ One-sided label smoothing. Instead of using one-hot labels in classification, use $(1 - \alpha)$ for real data (the generated samples are not smoothed).

$$D^*(\mathbf{x}) = \frac{(1 - \alpha)\pi(\mathbf{x})}{\pi(\mathbf{x}) + p(\mathbf{x}|\theta)}$$

- ▶ Virtual batch normalization. BatchNorm makes samples within minibatch are highly correlated.



Use reference fixed batch to compute the normalization statistics. To avoid overfitting construct batch with the reference batch and the current sample.

Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

DCGAN

Vanishing gradients

Improved techniques for training GANs

Adversarial variational Bayes

WGAN

Spectral Normalization GAN

Evolution of GANs

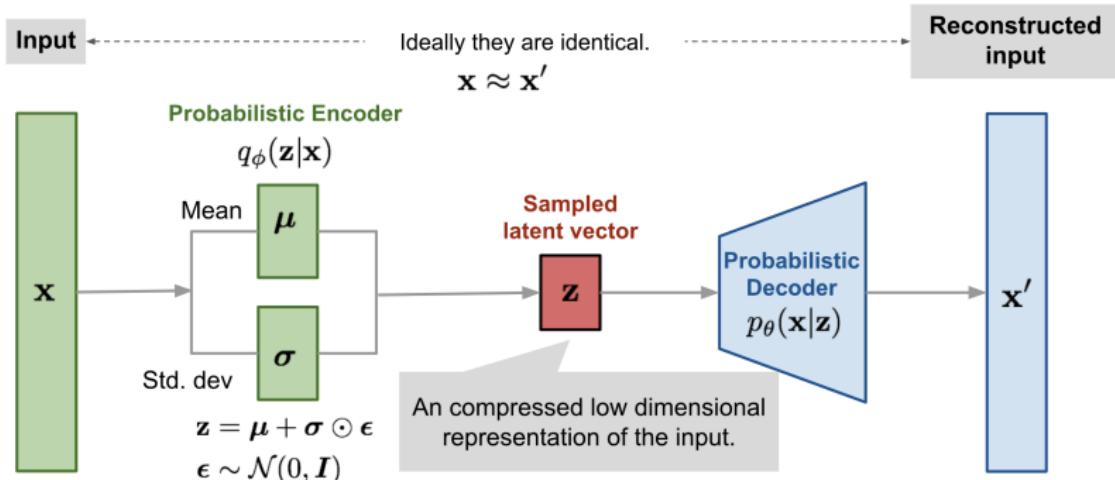
Evaluation of likelihood-free models

8. FFJORD

9. Quantized latents

10. Score matching

VAE recap



- ▶ Encoder $q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x}))$.
- ▶ Variational posterior $q(\mathbf{z}|\mathbf{x}, \phi)$ originally approximates the true posterior $p(\mathbf{z}|\mathbf{x}, \theta)$.
- ▶ Which methods are you already familiar with to make the posterior more flexible?

image credit:

<https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html>

Adversarial Variational Bayes

ELBO objective

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \theta) - KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z})) \rightarrow \max_{\phi, \theta} .$$

What is the problem to make the variational posterior model an implicit model?

- ▶ The first term is the reconstruction loss that needs only samples from $q(\mathbf{z}|\mathbf{x}, \phi)$ to evaluate.
- ▶ Reparametrization trick allows to get gradients of the reconstruction loss

$$\begin{aligned}\nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) f(\mathbf{z}) d\mathbf{z} &= \nabla_{\phi} \int r(\epsilon) f(\mathbf{z}) d\epsilon \\ &= \int r(\epsilon) \nabla_{\phi} f(g_{\phi}(\mathbf{x}, \epsilon)) d\epsilon \approx \nabla_{\phi} f(g_{\phi}(\mathbf{x}, \epsilon^*)),\end{aligned}$$

where $\epsilon^* \sim r(\epsilon)$, $\mathbf{z} = g_{\phi}(\mathbf{x}, \epsilon)$, $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \phi)$.

Adversarial Variational Bayes

ELBO objective

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(z|x, \phi)} \log p(x|z, \theta) - KL(q(z|x, \phi) || p(z)) \rightarrow \max_{\phi, \theta}.$$

What is the problem to make the variational posterior model an implicit model?

- ▶ The second term requires the explicit value of $q(z|x, \phi)$.
- ▶ We could join second and third terms:

$$KL = \mathbb{E}_{q(z|x, \phi)} \log \frac{q(z|x, \phi)}{p(z)} = \mathbb{E}_{q(z|x, \phi)} \log \frac{q(z|x, \phi)\pi(x)}{p(z)\pi(x)}.$$

- ▶ We have to estimate density ratio

$$r(x, z) = \frac{q_1(x, z)}{q_2(x, z)} = \frac{q(z|x, \phi)\pi(x)}{p(z)\pi(x)}.$$

Density ratio trick

Consider two distributions $q_1(\mathbf{x})$, $q_2(\mathbf{x})$ and probabilistic model

$$p(\mathbf{x}|y) = \begin{cases} q_1(\mathbf{x}), & \text{if } y = 1, \\ q_2(\mathbf{x}), & \text{if } y = 0, \end{cases} \quad y \sim \text{Bern}(0.5).$$

Density ratio

$$\begin{aligned} \frac{q_1(\mathbf{x})}{q_2(\mathbf{x})} &= \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} = \frac{p(y=1|\mathbf{x})p(\mathbf{x})}{p(y=1)} \Big/ \frac{p(y=0|\mathbf{x})p(\mathbf{x})}{p(y=0)} = \\ &= \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{1 - p(y=1|\mathbf{x})} = \frac{D(\mathbf{x})}{1 - D(\mathbf{x})} \end{aligned}$$

Here $D(\mathbf{x})$ is a discriminator model the output of which is a probability that \mathbf{x} is a sample from $q_1(\mathbf{x})$ rather than from $q_2(\mathbf{x})$.

$$\max_D [\mathbb{E}_{q_1(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{q_2(\mathbf{x})} \log(1 - D(\mathbf{x}))]$$

Density ratio trick

$$r(\mathbf{x}, \mathbf{z}) = \frac{q_1(\mathbf{x}, \mathbf{z})}{q_2(\mathbf{x}, \mathbf{z})} = \frac{q(\mathbf{z}|\mathbf{x}, \phi)\pi(\mathbf{x})}{p(\mathbf{z})\pi(\mathbf{x})}.$$

Density ratio

$$\frac{q_1(\mathbf{x}, \mathbf{z})}{q_2(\mathbf{x}, \mathbf{z})} = \frac{p(y=1|\mathbf{x}, \mathbf{z})}{1 - p(y=1|\mathbf{x}, \mathbf{z})} = \frac{D(\mathbf{x}, \mathbf{z})}{1 - D(\mathbf{x}, \mathbf{z})}$$

Adversarial Variational Bayes

$$\max_D \left[\mathbb{E}_{\pi(\mathbf{x})} \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log D(\mathbf{x}, \mathbf{z}) + \mathbb{E}_{\pi(\mathbf{x})} \mathbb{E}_{p(\mathbf{z})} \log(1 - D(\mathbf{x}, \mathbf{z})) \right]$$

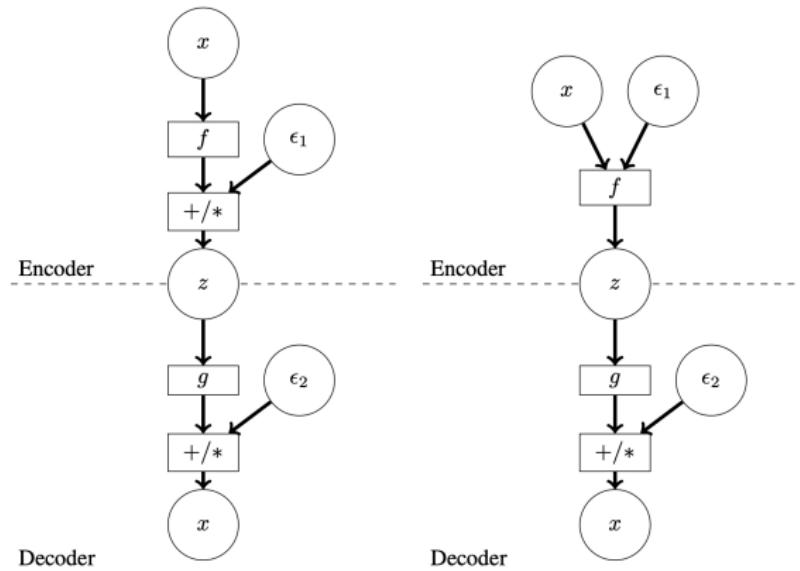
Monte-Carlo estimation for KL divergence:

$$KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z})) \approx \frac{D(\mathbf{x}, \mathbf{z})}{1 - D(\mathbf{x}, \mathbf{z})}.$$

Adversarial Variational Bayes

ELBO objective

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(z|x, \phi)} \left[\log p(x|z, \theta) - \log \frac{q(z|x, \phi)}{p(z)} \right] \rightarrow \max_{\phi, \theta} .$$



Outline

1. Autoregressive models

2. Variational inference

3. VAE-related topics

4. Normalizing Flows

5. ELBO surgery

6. Disentanglement

7. GANs

DCGAN

Vanishing gradients

Improved techniques for training GANs

Adversarial variational Bayes

WGAN

Spectral Normalization GAN

Evolution of GANs

Evaluation of likelihood-free models

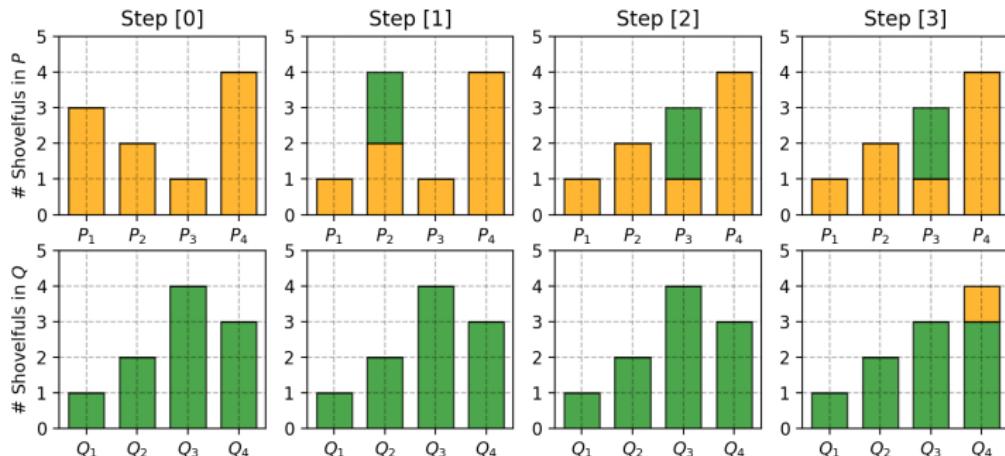
8. FFJORD

9. Quantized latents

10. Score matching

Wasserstein distance (discrete)

A.k.a. **Earth Mover's distance**. The minimum cost of moving and transforming a pile of dirt in the shape of one probability distribution to the shape of the other distribution.



$$W(P, Q) = 2(\text{step 1}) + 2(\text{step 2}) + 1(\text{step 3}) = 5$$

Wasserstein GAN

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α , the learning rate. c , the clipping parameter. m , the batch size.
 n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$ 
12: end while
```

Wasserstein GAN with Gradient Penalty

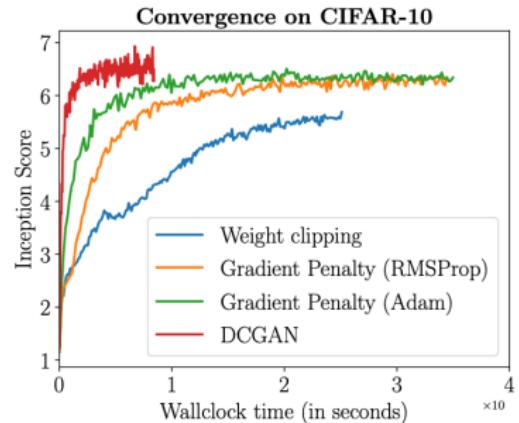
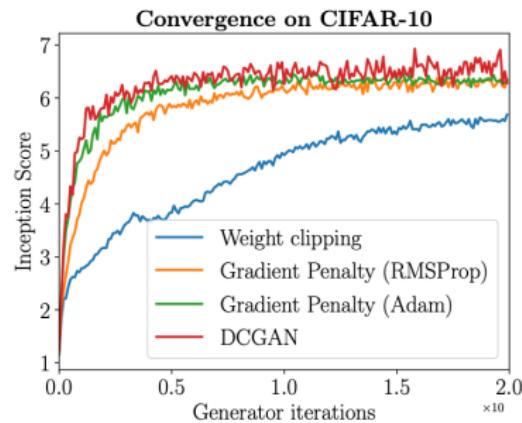
Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ , the number of critic iterations per generator iteration n_{critic} , the batch size m , Adam hyperparameters α, β_1, β_2 .

Require: initial critic parameters w_0 , initial generator parameters θ_0 .

```
1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ , latent variable  $\mathbf{z} \sim p(\mathbf{z})$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{\mathbf{x}} \leftarrow G_\theta(\mathbf{z})$ 
6:        $\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ 
7:        $L^{(i)} \leftarrow D_w(\tilde{\mathbf{x}}) - D_w(\mathbf{x}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:   end for
11:   Sample a batch of latent variables  $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim p(\mathbf{z})$ .
12:    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(\mathbf{z})), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while
```

Wasserstein GAN with Gradient Penalty



WGANGP convergence

Min. score	Only GAN	Only WGANGP	Both succeeded	Both failed
1.0	0	8	192	0
3.0	1	88	110	1
5.0	0	147	42	11
7.0	1	104	5	90
9.0	0	0	0	200

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
 - DCGAN
 - Vanishing gradients
 - Improved techniques for training GANs
 - Adversarial variational Bayes
 - WGAN
 - Spectral Normalization GAN**
 - Evolution of GANs
 - Evaluation of likelihood-free models
8. FFJORD
9. Quantized latents
10. Score matching

Spectral Normalization GAN

Definition

$\|\mathbf{A}\|_2$ is a *spectral norm* of matrix \mathbf{A} :

$$\|\mathbf{A}\|_2 = \max_{\mathbf{h} \neq 0} \frac{\|\mathbf{A}\mathbf{h}\|_2}{\|\mathbf{h}\|_2} = \max_{\|\mathbf{h}\|_2 \leq 1} \|\mathbf{A}\mathbf{h}\|_2 = \sqrt{\lambda_{\max}(\mathbf{A}^T \mathbf{A})},$$

where $\lambda_{\max}(\mathbf{A}^T \mathbf{A})$ is the largest eigenvalue value of $\mathbf{A}^T \mathbf{A}$.

Statement 1

if \mathbf{g} is a K-Lipschitz vector function then

$$\|\mathbf{g}\|_L \leq K = \sup_{\mathbf{x}} \|\nabla \mathbf{g}(\mathbf{x})\|_2.$$

Statement 2

Lipschitz norm of superposition is bounded above by product of Lipschitz norms

$$\|\mathbf{g}_1 \circ \mathbf{g}_2\|_L \leq \|\mathbf{g}_1\|_L \cdot \|\mathbf{g}_2\|_L$$

Spectral Normalization GAN

Let consider the critic $f_\phi(\mathbf{x})$ of the following form:

$$f_\phi(\mathbf{x}) = \mathbf{W}_{K+1} \sigma_K (\mathbf{W}_K \sigma_{K-1} (\dots \sigma_1 (\mathbf{W}_1 \mathbf{x}) \dots)).$$

This feedforward network is a superposition of simple functions.

- ▶ σ_k is a pointwise nonlinearities. We assume that $\|\sigma_k\|_L = 1$ (it holds for ReLU).
- ▶ $\mathbf{g}(\mathbf{x}) = \mathbf{W}^T \mathbf{x}$ is a linear transformation ($\nabla \mathbf{g}(\mathbf{x}) = \mathbf{W}$).

$$\|\mathbf{g}\|_L \leq \sup_{\mathbf{x}} \|\nabla \mathbf{g}(\mathbf{x})\|_2 = \|\mathbf{W}\|_2.$$

Critic spectral norm

$$\|f\|_L \leq \|\mathbf{W}_{K+1}\|_2 \cdot \prod_{k=1}^K \|\sigma_k\|_L \cdot \|\mathbf{W}_k\|_2 = \prod_{k=1}^{K+1} \|\mathbf{W}_k\|_2.$$

If we replace the weights in the critic $f_\phi(\mathbf{x})$ by

$\mathbf{W}_k^{SN} = \mathbf{W}_k / \|\mathbf{W}_k\|_2$, we will get $\|f\|_L \leq 1$.

Spectral Normalization GAN

How to compute $\|\mathbf{W}\|_2 = \sqrt{\lambda_{\max}(\mathbf{W}^T \mathbf{W})}$?

We are not able to apply SVD at each iteration.

Power iteration (PI) method

- ▶ \mathbf{u}_0 – random vector.
- ▶ for $m = 0, \dots, M - 1$: (M is a fixed number of steps)

$$\mathbf{v}_{m+1} = \frac{\mathbf{W}^T \mathbf{u}_m}{\|\mathbf{W}^T \mathbf{u}_m\|}, \quad \mathbf{u}_{m+1} = \frac{\mathbf{W} \mathbf{v}_{m+1}}{\|\mathbf{W} \mathbf{v}_{m+1}\|}.$$

- ▶ approximate the spectral norm

$$\|\mathbf{W}\|_2 = \sqrt{\lambda_{\max}(\mathbf{W}^T \mathbf{W})} \approx \mathbf{u}_M^T \mathbf{W} \mathbf{v}_M.$$

SNGAN gradient update

- ▶ Apply PI method to get approximation of spectral norm.
- ▶ Normalize weights $\mathbf{W}_k^{SN} = \mathbf{W}_k / \|\mathbf{W}_k\|_2$.
- ▶ Apply gradient rule to \mathbf{W} .

Spectral Normalization GAN

Algorithm 1 SGD with spectral normalization

- Initialize $\tilde{\mathbf{u}}_l \in \mathcal{R}^{d_l}$ for $l = 1, \dots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer l :
 1. Apply power iteration method to a unnormalized weight W^l :

$$\tilde{\mathbf{v}}_l \leftarrow (W^l)^T \tilde{\mathbf{u}}_l / \| (W^l)^T \tilde{\mathbf{u}}_l \|_2 \quad (20)$$

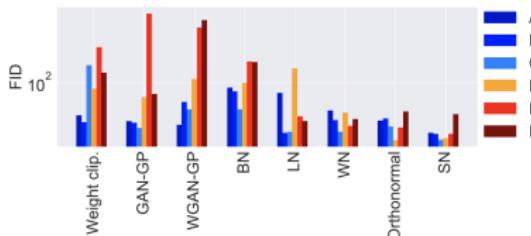
$$\tilde{\mathbf{u}}_l \leftarrow W^l \tilde{\mathbf{v}}_l / \| W^l \tilde{\mathbf{v}}_l \|_2 \quad (21)$$

2. Calculate \bar{W}_{SN} with the spectral norm:

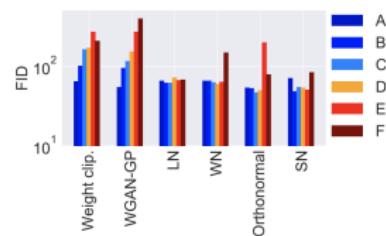
$$\bar{W}_{\text{SN}}^l(W^l) = W^l / \sigma(W^l), \text{ where } \sigma(W^l) = \tilde{\mathbf{u}}_l^T W^l \tilde{\mathbf{v}}_l \quad (22)$$

3. Update W^l with SGD on mini-batch dataset \mathcal{D}_M with a learning rate α :

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}_{\text{SN}}^l(W^l), \mathcal{D}_M) \quad (23)$$



(a) CIFAR-10



(b) STL-10

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
 - DCGAN
 - Vanishing gradients
 - Improved techniques for training GANs
 - Adversarial variational Bayes
 - WGAN
 - Spectral Normalization GAN
 - Evolution of GANs**
 - Evaluation of likelihood-free models
8. FFJORD
9. Quantized latents
10. Score matching

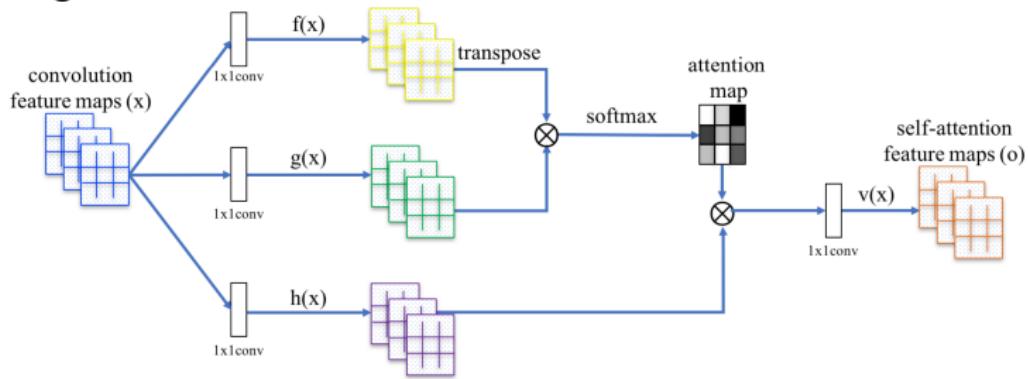
Evolution of GANs



- ▶ **Standard GAN** <https://arxiv.org/abs/1406.2661>
- ▶ **DCGAN** <https://arxiv.org/abs/1511.06434>
- ▶ **CoGAN** <https://arxiv.org/abs/1606.07536>
- ▶ **ProGAN** <https://arxiv.org/abs/1710.10196>
- ▶ **StyleGAN** <https://arxiv.org/abs/1812.04948>

Self-Attention GAN

Convolutional layers process the information in a local neighborhood \Rightarrow inefficient for modeling long-range dependencies in images.

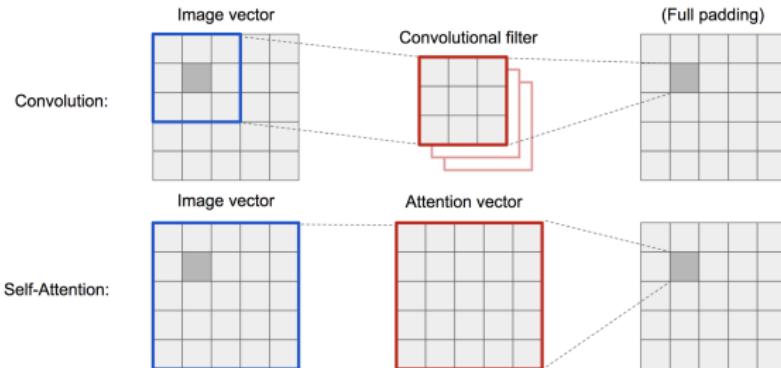


$$\mathbf{f}(\mathbf{x}) = \mathbf{W}_f \mathbf{x}, \quad \mathbf{g}(\mathbf{x}) = \mathbf{W}_g \mathbf{x}, \quad \mathbf{h}(\mathbf{x}) = \mathbf{W}_h \mathbf{x}, \quad \mathbf{v}(\mathbf{x}) = \mathbf{W}_v \mathbf{x}$$

$$s_{ij} = \mathbf{f}(\mathbf{x}_i)^T \mathbf{g}(\mathbf{x}_j), \quad a_{ij} = \frac{\exp s_{ij}}{\sum_{i=1}^N \exp s_{ij}}, \quad \mathbf{o}_j = \mathbf{v} \left(\sum_{i=1}^N a_{ij} \mathbf{h}(\mathbf{x}_i) \right)$$

Self-Attention GAN

Convolution vs Attention



Visualization of attention maps

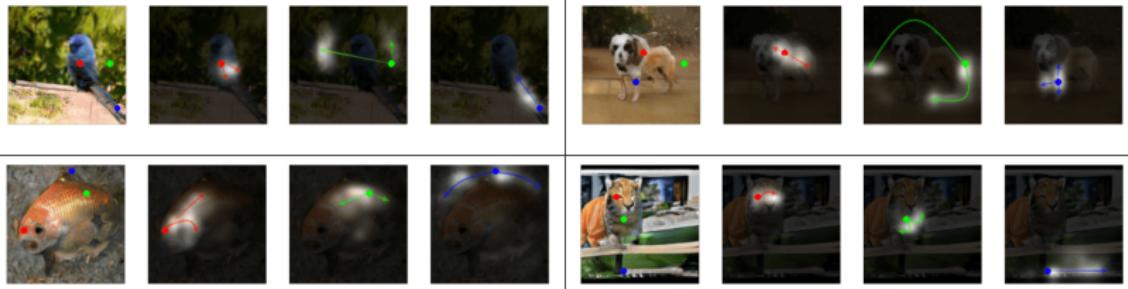


image credit: <https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>
Zhang H. et al. Self-Attention Generative Adversarial Networks, 2018

BigGAN

Batch-size is matter

Batch	Ch.	Param (M)	Shared	Skip- z	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5	SA-GAN Baseline			1000	18.65	52.52
512	64	81.5	X	X	X	1000	15.30	58.77(± 1.18)
1024	64	81.5	X	X	X	1000	14.88	63.03(± 1.42)
2048	64	81.5	X	X	X	732	12.39	76.85(± 3.83)
2048	96	173.5	X	X	X	295(± 18)	9.54(± 0.62)	92.98(± 4.27)

Samples (512x512)



Progressive Growing GAN

Problems with HR image generation

- ▶ Disjoint manifolds \Rightarrow gradient problem.
- ▶ Small minibatch \Rightarrow training instability.

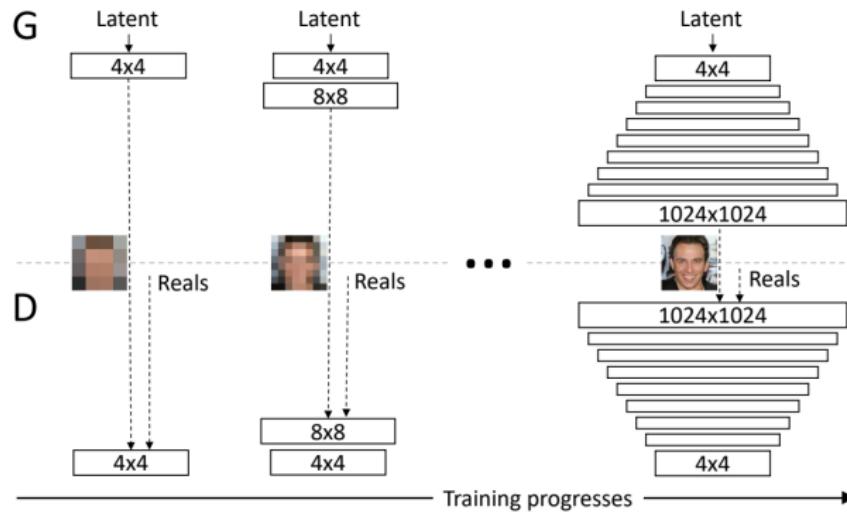
Samples (1024x1024)



Progressive Growing GAN

Grow both the generator and discriminator progressively, new layers will introduce higher-resolution details as the training progresses.

- ▶ Train GAN which generate 4x4 images (2 convs for G and D).
- ▶ Add upsampling layers to G, downsampling layers to D.
- ▶ Train GAN which generate 8x8 images.
- ▶ etc.



Karras T. et al. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*, 2017

StyleGAN

- ▶ Generating of HR images is hard.
- ▶ Progressive growing greatly simplifies the task.
- ▶ The ability to control specific features of the generated image is very limited.

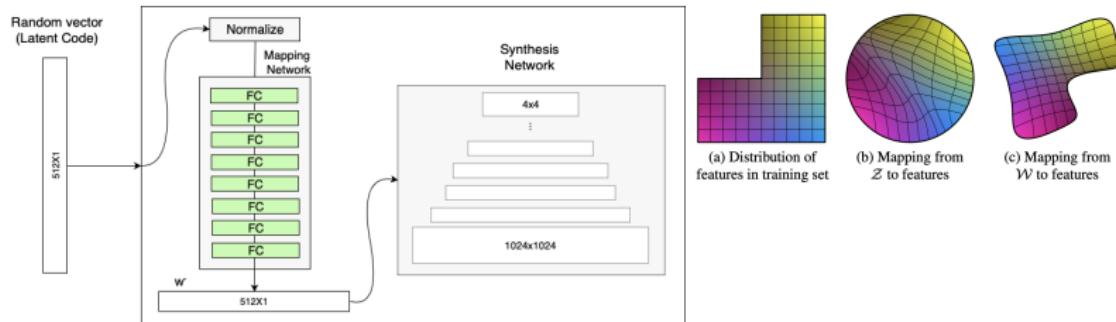
Face image features

- ▶ Coarse (pose, general hair style, face shape). Resolution $4^2 - 8^2$.
- ▶ Middle (finer facial features, hair style, eyes open/closed). Resolution $16^2 - 32^2$.
- ▶ Fine (color scheme (eye, hair and skin) and micro features). Resolution $64^2 - 1024^2$.

StyleGAN

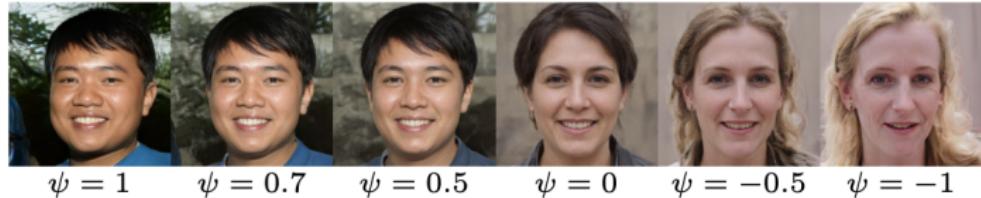
Mapping Network

- ▶ Generator input is likely to be **disentangled**. Each component of input vector \mathbf{z} should be responsible for one generative factor.
- ▶ Mapping network $f : \mathcal{Z} \rightarrow \mathcal{W}$ is used to reduce correlations between components of \mathbf{z} .



StyleGAN

Truncation trick



Samples (1024x1024)

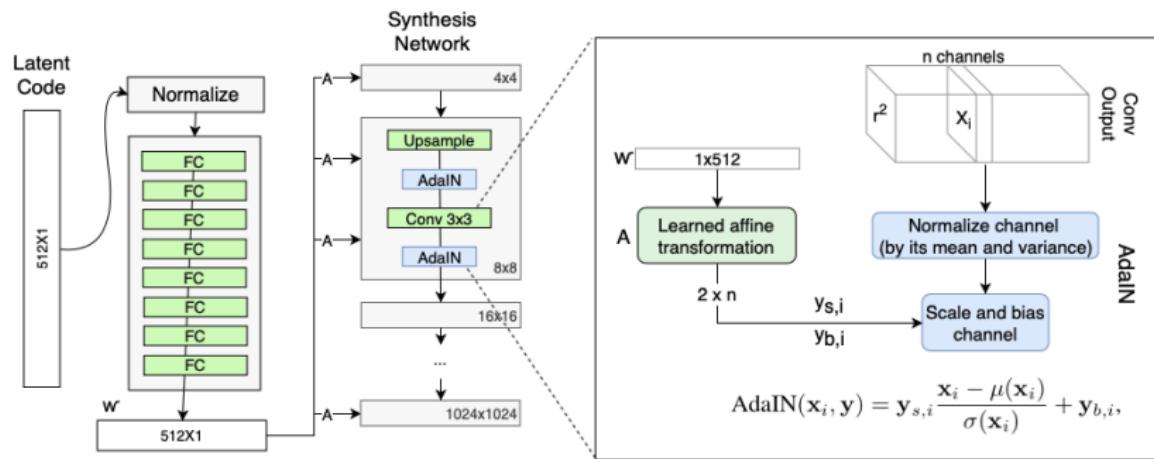


Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

StyleGAN

Step 2: Style modulation

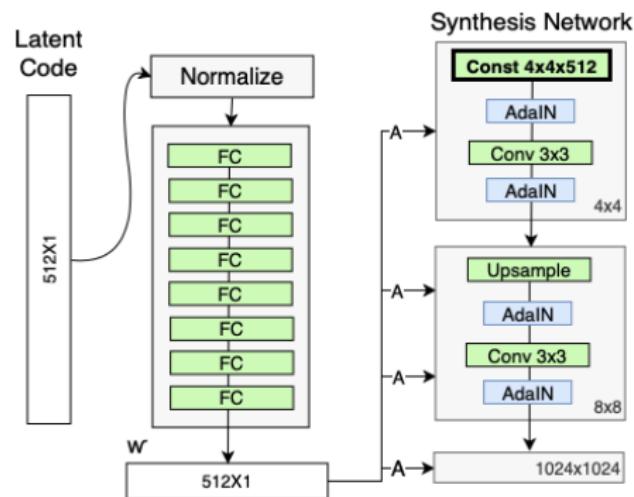
- ▶ Adaptive Instance Normalization transfers the \mathbf{w} vector to the synthesis Network.
- ▶ The module is added to each resolution to define the visual expression of the features.



StyleGAN

Step 3: Remove traditional input

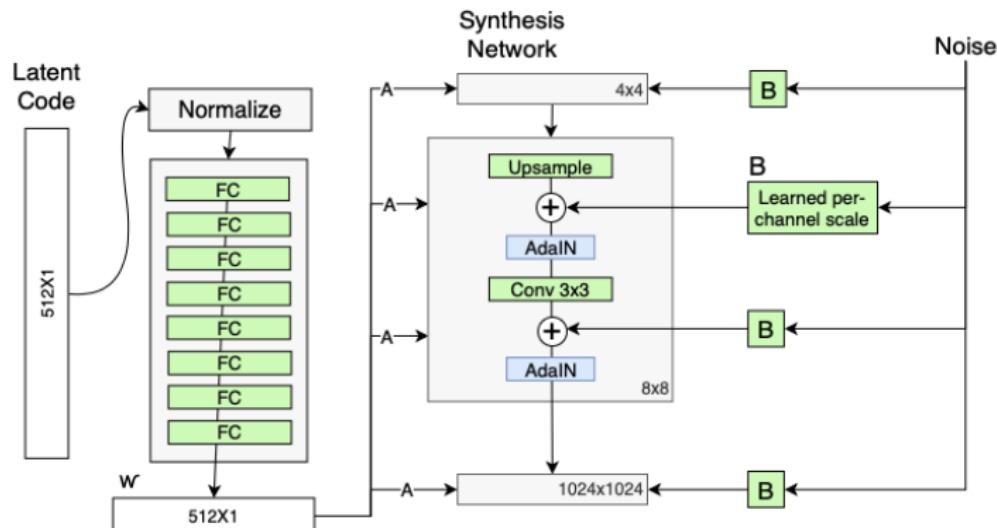
Mapping network provides stochasticity to different stages of the synthesis network. Input of the synthesis network is a trainable vector.



StyleGAN

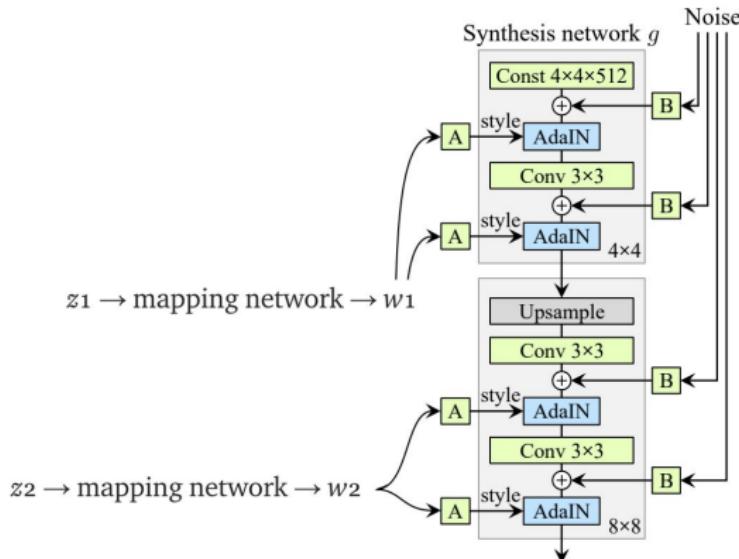
Step 4: Stochastic variation

Inject random noise to add small aspects, such as freckles, exact placement of hairs, wrinkles, features which make the image more realistic and increase the variety of outputs.



StyleGAN

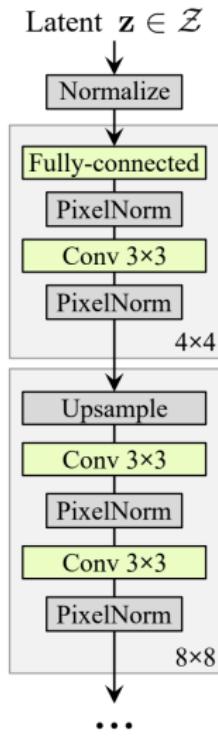
Step 4: Style Mixing



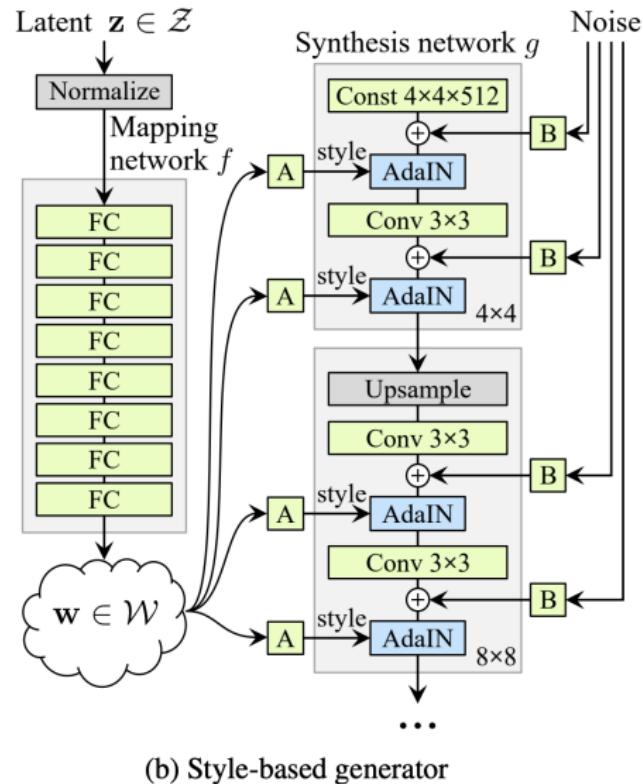
- ▶ Makes different levels of synthesis network to be independent.
- ▶ Allows to couple different styles.

Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

StyleGAN

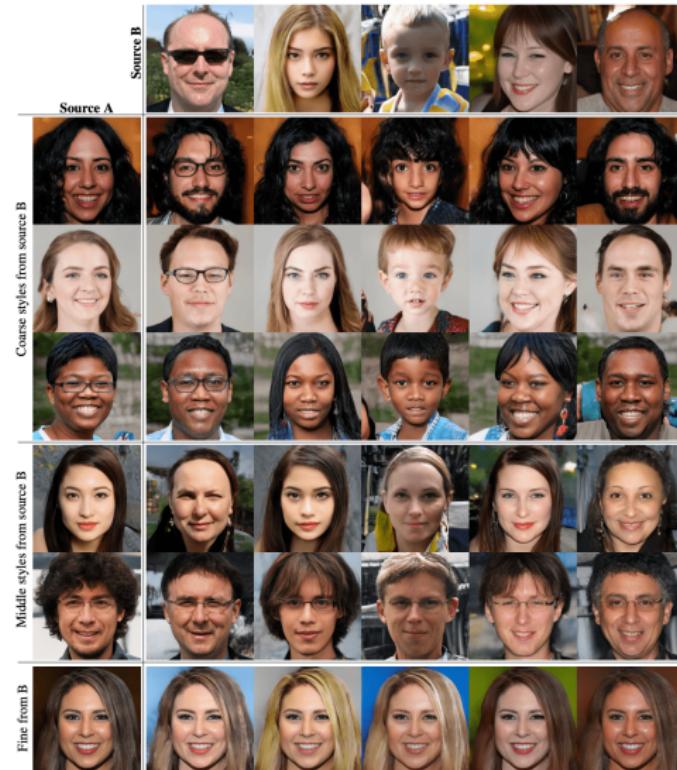


(a) Traditional



(b) Style-based generator

StyleGAN



Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
 - DCGAN
 - Vanishing gradients
 - Improved techniques for training GANs
 - Adversarial variational Bayes
 - WGAN
 - Spectral Normalization GAN
 - Evolution of GANs
 - Evaluation of likelihood-free models**
8. FFJORD
9. Quantized latents
10. Score matching

Evaluation of likelihood-free models

What do we want from samples?

- ▶ **Sharpness.** The conditional distribution $p(y|x)$ should have low entropy (each image x should have distinctly recognizable object).
- ▶ **Diversity.** The marginal distribution $p(y) = \int p(y|x)p(x)dx$ should have high entropy (there should be as many classes generated as possible).

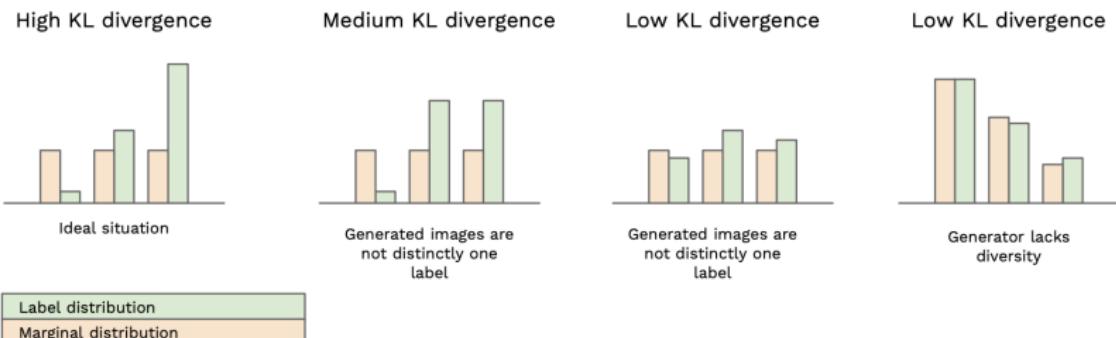


image credit: <https://medium.com/octavian-ai/a-simple-explanation-of-the-inception-score-372dff6a8c7a>

Evaluation of likelihood-free models

What do we want from samples?

- ▶ Sharpness \Rightarrow low $H(y|\mathbf{x}) = - \sum_y \int_{\mathbf{x}} p(y, \mathbf{x}) \log p(y|\mathbf{x}) d\mathbf{x}$.
- ▶ Diversity \Rightarrow high $H(y) = - \sum_y p(y) \log p(y)$.

Inception Score

$$\begin{aligned} IS &= \exp(H(y) - H(y|\mathbf{x})) \\ &= \exp \left(- \sum_y p(y) \log p(y) + \sum_y \int_{\mathbf{x}} p(y, \mathbf{x}) \log p(y|\mathbf{x}) d\mathbf{x} \right) \\ &= \exp \left(\sum_y \int_{\mathbf{x}} p(y, \mathbf{x}) \log \frac{p(y|\mathbf{x})}{p(y)} d\mathbf{x} \right) \\ &= \exp \left(\mathbb{E}_{\mathbf{x}} \sum_y p(y|\mathbf{x}) \log \frac{p(y|\mathbf{x})}{p(y)} \right) = \exp (\mathbb{E}_{\mathbf{x}} KL(p(y|\mathbf{x}) || p(y))) \end{aligned}$$

Evaluation of likelihood-free models

Theorem (informal)

If $\pi(\mathbf{x})$ and $p(\mathbf{x}|\theta)$ has moment generation functions then

$$\pi(\mathbf{x}) = p(\mathbf{x}|\theta) \Leftrightarrow \mathbb{E}_\pi \mathbf{x}^k = \mathbb{E}_p \mathbf{x}^k, \quad \forall k \geq 1.$$

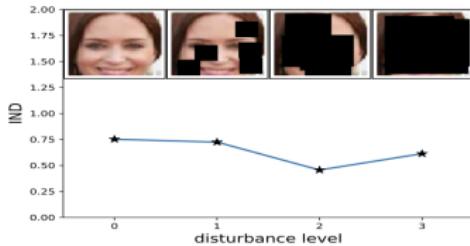
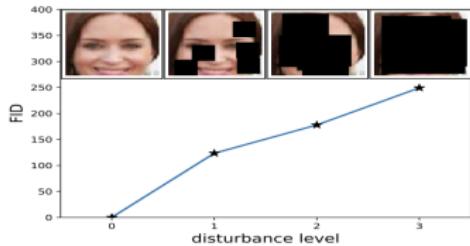
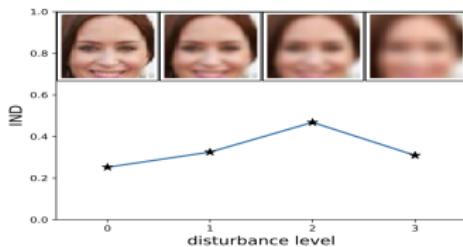
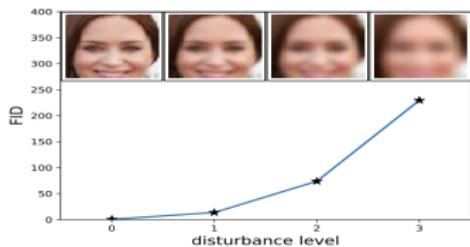
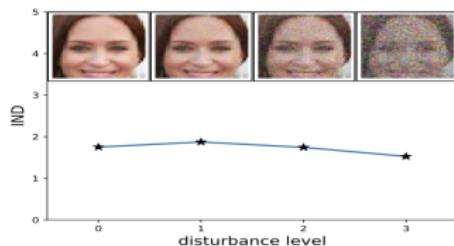
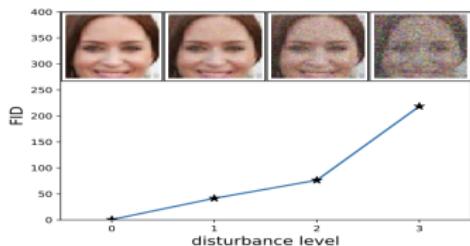
This is intractable to calculate all moments.

Frechet Inception Distance

$$FID(\pi, p) = \|\mathbf{m}_\pi - \mathbf{m}_p\|_2^2 + \text{Tr} \left(\boldsymbol{\Sigma}_\pi + \boldsymbol{\Sigma}_p - 2\sqrt{\boldsymbol{\Sigma}_\pi \boldsymbol{\Sigma}_p} \right)$$

- ▶ Representations are the outputs of the intermediate layer from the pretrained classification model.
- ▶ $\mathbf{m}_\pi, \boldsymbol{\Sigma}_\pi$ are the mean vector and the covariance matrix of feature representations for samples from $\pi(\mathbf{x})$
- ▶ $\mathbf{m}_p, \boldsymbol{\Sigma}_p$ are the mean vector and the covariance matrix of feature representations for samples from $p(\mathbf{x}|\theta)$.

Evaluation of likelihood-free models



Limitations

Inception Score

$$IS = \exp(\mathbb{E}_{\mathbf{x}} KL(p(y|\mathbf{x}) || p(y)))$$

- ▶ If generator produces images with a different set of labels from the classifier training set, IS will be low.
- ▶ If generator produces one image per class, the IS will be perfect (there is no measure of intra-class diversity).

Frechet Inception Distance

$$FID = \|\mathbf{m}_\pi - \mathbf{m}_p\|_2^2 + \text{Tr} \left(\boldsymbol{\Sigma}_\pi + \boldsymbol{\Sigma}_p - 2\sqrt{\boldsymbol{\Sigma}_\pi \boldsymbol{\Sigma}_p} \right)$$

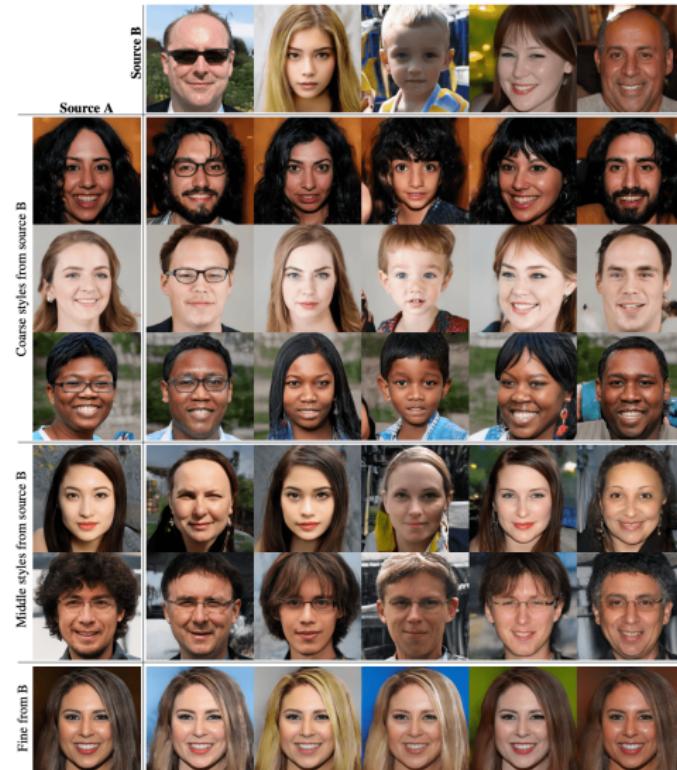
- ▶ Needs a large sample size for evaluation.
- ▶ Calculation of FID is slow.
- ▶ Uses the normality assumption!

Both scores depend on the pretrained classifier $p(y|\mathbf{x})$.

Barratt S., Sharma R. A Note on the Inception Score, 2018

Heusel M. et al. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, 2017

StyleGAN



Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching

Continuous Normalizing Flows

Forward transform + log-density

$$\begin{bmatrix} \mathbf{x} \\ \log p(\mathbf{x}|\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \mathbf{z} \\ \log p(\mathbf{z}) \end{bmatrix} + \int_{t_0}^{t_1} \begin{bmatrix} f(\mathbf{z}(t), \boldsymbol{\theta}) \\ -\text{trace}\left(\frac{\partial f(\mathbf{z}(t), \boldsymbol{\theta})}{\partial \mathbf{z}(t)}\right) \end{bmatrix} dt.$$

- ▶ Discrete-in-time normalizing flows need invertible f . It costs $O(d^3)$ to get determinant of Jacobian.
- ▶ Continuous-in-time flows require only smoothness of f . It costs $O(d^2)$ to get trace of Jacobian.

It is possible to reduce cost from $O(d^2)$ to $O(d)$!

Hutchinson's trace estimator

$$\text{trace}(A) = \mathbb{E}_{p(\epsilon)} \left[\epsilon^T A \epsilon \right]; \quad \mathbb{E}[\epsilon] = 0; \quad \text{Cov}(\epsilon) = I.$$

FFJORD density estimation

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \mathbb{E}_{p(\epsilon)} \int_{t_0}^{t_1} \left[\epsilon^T \frac{\partial f}{\partial \mathbf{z}} \epsilon \right] dt.$$

	Method	One-pass Sampling	Exact log-likelihood	Free-form Jacobian
Variational Autoencoders	Variational Autoencoders	✓	✗	✓
	Generative Adversarial Nets	✓	✗	✓
	Likelihood-based Autoregressive	✗	✓	✗
Change of Variables	Normalizing Flows	✓	✓	✗
	Reverse-NF, MAF, TAN	✗	✓	✗
	NICE, Real NVP, Glow, Planar CNF	✓	✓	✗
	FFJORD	✓	✓	✓

Density estimation (forward KL)

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST	CIFAR10
Real NVP	-0.17	-8.33	18.71	13.55	-153.28	1.06*	3.49*
Glow	-0.17	-8.15	18.92	11.35	-155.07	1.05*	3.35*
FFJORD	-0.46	-8.59	14.92	10.43	-157.40	0.99* (1.05 [†])	3.40*

Flows for variational inference (reverse KL)

	MNIST	Omniglot	Frey Faces	Caltech Silhouettes
IAF	$84.20 \pm .17$	$102.41 \pm .04$	$4.47 \pm .05$	$111.58 \pm .38$
Sylvester	$83.32 \pm .06$	$99.00 \pm .04$	$4.45 \pm .04$	$104.62 \pm .29$
FFJORD	$82.82 \pm .01$	$98.33 \pm .09$	$4.39 \pm .01$	$104.03 \pm .43$

Grathwohl W. et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models, 2018

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
 - Vector Quantized VAE-2
 - Feature Quantized GAN
10. Score matching

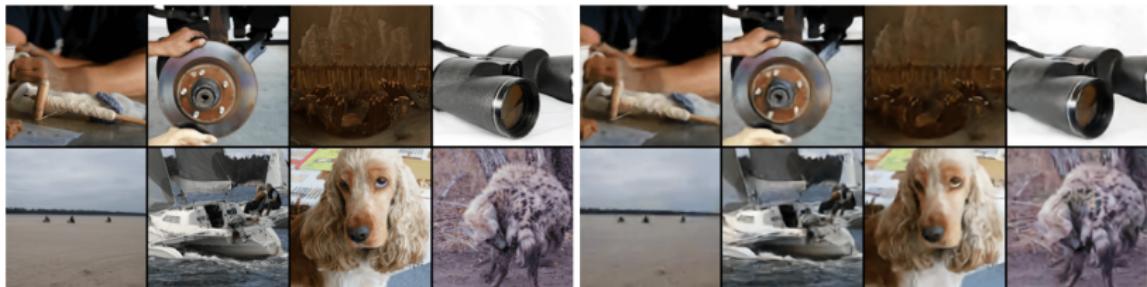
Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
 - Vector Quantized VAE-2
 - Feature Quantized GAN
10. Score matching

Vector Quantized VAE

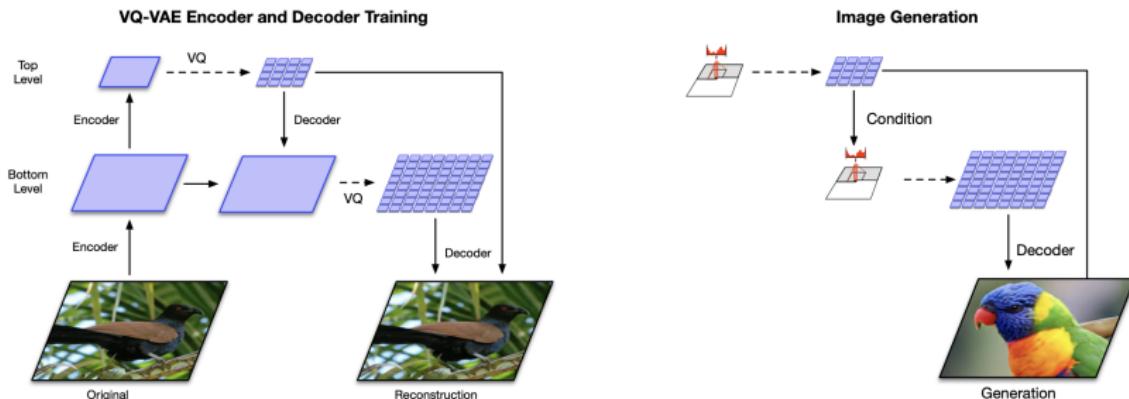
- ▶ The prior distribution over the discrete latents $p(\hat{z})$ is a categorical distribution.
- ▶ It could be made autoregressive by depending on other \hat{z} in the feature map.
- ▶ While training the VQ-VAE, the prior is kept constant and uniform.
- ▶ After training, fit an autoregressive distribution (using PixelCNN) over \hat{z} .

Samples



Vector Quantized VAE-2

- ▶ Use multi-scale hierarchical model.
- ▶ Use autoregressive prior model in each scale of the hierarchy.
- ▶ Improve autoregressive prior (PixelSNAIL with self-attention in bottom layer, PixelCNN++ in bottom layer).
- ▶ Train the encoder and decoder at the first stage, train the priors at the second stage.



Vector Quantized VAE-2

Algorithm 1 VQ-VAE training (stage 1)

Require: Functions E_{top} , E_{bottom} , D , \mathbf{x} (batch of training images)

- 1: $\mathbf{h}_{top} \leftarrow E_{top}(\mathbf{x})$
 ▷ quantize with top codebook eq 1
- 2: $\mathbf{e}_{top} \leftarrow Quantize(\mathbf{h}_{top})$
- 3: $\mathbf{h}_{bottom} \leftarrow E_{bottom}(\mathbf{x}, \mathbf{e}_{top})$
 ▷ quantize with bottom codebook eq 1
- 4: $\mathbf{e}_{bottom} \leftarrow Quantize(\mathbf{h}_{bottom})$
- 5: $\hat{\mathbf{x}} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$
 ▷ Loss according to eq 2
- 6: $\theta \leftarrow Update(\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}))$

Algorithm 2 Prior training (stage 2)

- 1: $\mathbf{T}_{top}, \mathbf{T}_{bottom} \leftarrow \emptyset$ ▷ training set
- 2: **for** $\mathbf{x} \in$ training set **do**
- 3: $\mathbf{e}_{top} \leftarrow Quantize(E_{top}(\mathbf{x}))$
- 4: $\mathbf{e}_{bottom} \leftarrow Quantize(E_{bottom}(\mathbf{x}, \mathbf{e}_{top}))$
- 5: $\mathbf{T}_{top} \leftarrow \mathbf{T}_{top} \cup \mathbf{e}_{top}$
- 6: $\mathbf{T}_{bottom} \leftarrow \mathbf{T}_{bottom} \cup \mathbf{e}_{bottom}$
- 7: **end for**
- 8: $p_{top} = TrainPixelCNN(\mathbf{T}_{top})$
- 9: $p_{bottom} = TrainCondPixelCNN(\mathbf{T}_{bottom}, \mathbf{T}_{top})$

▷ Sampling procedure

- 10: **while** true **do**
- 11: $\mathbf{e}_{top} \sim p_{top}$
- 12: $\mathbf{e}_{bottom} \sim p_{bottom}(\mathbf{e}_{top})$
- 13: $\mathbf{x} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$
- 14: **end while**



h_{top}



h_{top}, h_{middle}



$h_{top}, h_{middle}, h_{bottom}$



Original

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
 Vector Quantized VAE-2
 Feature Quantized GAN
10. Score matching

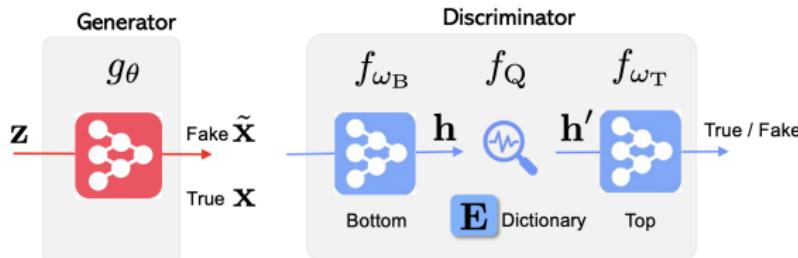
Feature Quantized GAN

- ▶ GAN tries to find Nash equilibrium, minibatch training is unstable. GAN relies heavily on the minibatch statistics.
- ▶ Lots of feature matching strategies were proposed to stabilize the training.

Feature quantized GAN discriminator

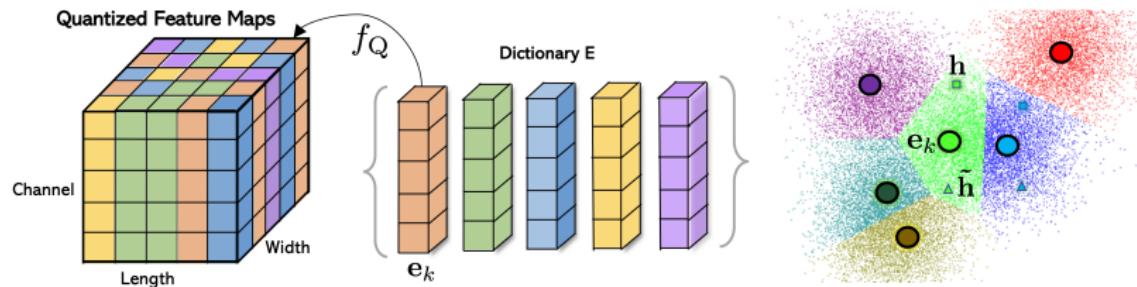
$$D(\mathbf{x}) = f_{\mathbf{w}_T} \circ f_{\mathbf{w}_B}(\mathbf{x}) \quad \Rightarrow \quad D(\mathbf{x}) = f_{\mathbf{w}_T} \circ f_Q \circ f_{\mathbf{w}_B}(\mathbf{x}).$$

Here f_Q is a vector quantization operation.



Feature Quantized GAN

Quantization procedure



Quantized features



Feature Quantized GAN

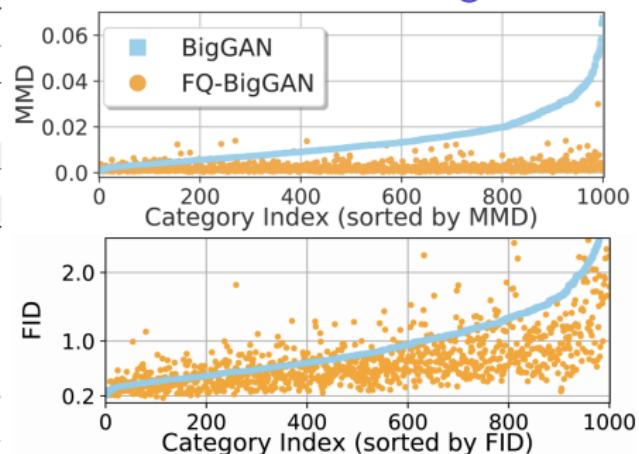
ImageNet-1000

Models	64 × 64		128 × 128	
	FID* ↓ / IS* ↑		FID* ↓ / IS* ↑	
Half	TAC-GAN	-	23.75 / 28.86±0.29 [‡]	
	BigGAN	12.75 / 21.84±0.34	22.77 / 38.05±0.79 [‡]	
256K	FQ-BigGAN	12.62 / 21.99±0.32	19.11 / 41.92±1.15	
	BigGAN	10.55 / 25.43±0.15	14.88 / 63.03±1.42 [†]	
	FQ-BigGAN	9.67 / 25.96±0.24	13.77 / 54.36±1.07	

FFHQ

Resolution	32 ²	64 ²	128 ²	1024 ²
StyleGAN	3.28	4.82	6.33	5.24
FQ-StyleGAN	3.01	4.36	5.98	4.89

Per class metrics for ImageNet



Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching
Implicit score matching

Outline

1. Autoregressive models
2. Variational inference
3. VAE-related topics
4. Normalizing Flows
5. ELBO surgery
6. Disentanglement
7. GANs
8. FFJORD
9. Quantized latents
10. Score matching
Implicit score matching

Implicit score matching

Theorem

Under some regularity conditions, it holds

$$\frac{1}{2} \mathbb{E}_\pi \| \mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log \pi(\mathbf{x}) \|_2^2 = \mathbb{E}_\pi \left[\frac{1}{2} \| \mathbf{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right] + \text{const}$$

Proof (only for 1D)

$$\mathbb{E}_\pi \| s(x) - \nabla_x \log \pi(x) \|_2^2 = \mathbb{E}_\pi [s(x)^2 + (\nabla_x \log \pi(x))^2 - 2[s(x) \nabla_x \log \pi(x)]]$$

$$\begin{aligned} \mathbb{E}_\pi [s(x) \nabla_x \log \pi(x)] &= \int \underbrace{\pi(x) s(x)}_g \underbrace{\nabla_x \log \pi(x)}_{\nabla f} dx = \int \underbrace{\nabla_x \log p(x)}_g \underbrace{\nabla_x \pi(x)}_{\nabla f} dx \\ &= \underbrace{\nabla_x \log p(x)}_g \underbrace{\pi(x)}_f \Big|_{-\infty}^{+\infty} - \int \underbrace{\nabla_x (\nabla_x \log p(x))}_{\nabla g} \underbrace{\pi(x)}_f dx \\ &= -\mathbb{E}_\pi \nabla_x s(x) \end{aligned}$$

$$\frac{1}{2} \mathbb{E}_\pi \| s(x) - \nabla_x \log \pi(x) \|_2^2 = \mathbb{E}_\pi \left[\frac{1}{2} s(x)^2 + \nabla_x s(x) \right] + \text{const.}$$

Implicit score matching

Theorem

$$\frac{1}{2} \mathbb{E}_\pi \| \mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log \pi(\mathbf{x}) \|_2^2 = \mathbb{E}_\pi \left[\frac{1}{2} \| \mathbf{s}_\theta(\mathbf{x}) \|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right] + \text{const}$$

Here $\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}) = \nabla_{\mathbf{x}}^2 \log p(\mathbf{x}|\theta)$ is a Hessian matrix.

1. The right hand side is complex due to Hessian matrix – **sliced score matching**.
2. The left hand side is intractable due to unknown $\pi(\mathbf{x})$ – **denoising score matching**.

Sliced score matching (Hutchinson's trace estimation)

$$\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) = \mathbb{E}_{p(\epsilon)} \left[\boldsymbol{\epsilon}^T \nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}) \boldsymbol{\epsilon} \right]$$

Song Y. Sliced Score Matching: A Scalable Approach to Density and Score Estimation, 2019

Song Y. Generative Modeling by Estimating Gradients of the Data Distribution, blog post, 2021