Deep Generative Models

Lecture 5

Roman Isachenko

Moscow Institute of Physics and Technology

Autumn, 2021

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

$$\nabla_\phi \mathcal{L}(\phi, \theta) = \nabla_\phi \int q(\mathbf{z}|\mathbf{x}, \phi) \left[\log p(\mathbf{x}|\mathbf{z}, \theta) + \log \frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}, \phi)} \right] d\mathbf{z}$$

$$= \int r(\epsilon) \nabla_\phi \left[\log p(\mathbf{x}, g(\mathbf{x}, \epsilon, \phi)|\theta) - \log q\big(g(\mathbf{x}, \epsilon, \phi)|\mathbf{x}, \phi\big) \right] d\epsilon$$

$$\approx \nabla_\phi \left[\log p(\mathbf{x}, g(\mathbf{x}, \epsilon^*, \phi)|\theta) - \log q\big(g(\mathbf{x}, \epsilon^*, \phi)|\mathbf{x}, \phi\big) \right]$$

Variational assumption

$$r(\epsilon) = \mathcal{N}(0, \mathbf{I}); \quad q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}^2_\phi(\mathbf{x})).$$

$$\mathbf{z} = g(\mathbf{x}, \epsilon, \phi) = \boldsymbol{\sigma}_\phi(\mathbf{x}) \cdot \epsilon + \boldsymbol{\mu}_\phi(\mathbf{x}).$$

Here $\boldsymbol{\mu}_\phi(\cdot), \boldsymbol{\sigma}_\phi(\cdot)$ are parameterized functions (outputs of neural network).

If we are able to calculate $\log p(\mathbf{x}, \mathbf{z}|\theta)$ and $\log q(\mathbf{z}|\mathbf{x}, \phi)$, we are done.

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

$$\nabla_\phi \mathcal{L}(\phi, \theta) \approx \nabla_\phi \big[\log p(\mathbf{x}, g(\mathbf{x}, \epsilon^*, \phi)|\theta) - \log q\big(g(\mathbf{x}, \epsilon^*, \phi)|\mathbf{x}, \phi\big)\big]$$

First term

$$\log p(\mathbf{x}, \mathbf{z}|\theta) = \log p(\mathbf{x}|\mathbf{z}, \theta) + \log p(\mathbf{z}).$$

- $p(\mathbf{z})$ – prior distribution on latent variables $\mathbf{z}$. We could specify any distribution that we want. Let say $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.
- $p(\mathbf{x}|\mathbf{z}, \theta)$ - generative distibution. Since it is a parameterized function let it be neural network with parameters θ.

Second term

Function $\mathbf{z} = g(\mathbf{x}, \epsilon, \phi) = \sigma_\phi(\mathbf{x}) \cdot \epsilon + \mu_\phi(\mathbf{x})$ is invertible.

$$q(\mathbf{z}|\mathbf{x}, \phi) = r(\epsilon) \cdot \left|\frac{\partial \epsilon}{\partial \mathbf{z}}\right| \quad \Rightarrow \quad \log q(\mathbf{z}|\mathbf{x}, \phi) = \log r(\epsilon) - \sum_{i=1}^{d} \log \left[\sigma_\phi(\mathbf{x})\right]_i$$

Recap of previous lecture

Fix probabilistic model $p(\mathbf{x}|\boldsymbol{\theta})$ – a set of parametrised distributions. Instead of searching true $\pi(\mathbf{x})$ over all probability distributions, we learn function approximation $p(\mathbf{x}|\boldsymbol{\theta}) \approx \pi(\mathbf{x})$.

Forward KL

$$KL(\pi||p) = \int \pi(\mathbf{x}) \log \frac{\pi(\mathbf{x})}{p(\mathbf{x}|\boldsymbol{\theta})} d\mathbf{x} \to \min_{\boldsymbol{\theta}}$$

Maximum likelihood estimation is equivalent to minimization of the Monte-Carlo estimation of forward KL.

Reverse KL

$$KL(p||\pi) = \int p(\mathbf{x}|\boldsymbol{\theta}) \log \frac{p(\mathbf{x}|\boldsymbol{\theta})}{\pi(\mathbf{x})} d\mathbf{x} \to \min_{\boldsymbol{\theta}}$$

Used for variational inference.

Recap of previous lecture

Likelihood-based models so far...

Autoregressive models

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{m} p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta})$$

- tractable likelihood,
- no inferred latent factors.

Latent variable models

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

- latent feature representation,
- intractable likelihood.

How to build a model with latent variables and tractable likelihood?

Recap of previous lecture

Change of variable theorem

- $\mathbf{x}$ is a random variable with density function $p(\mathbf{x})$;
- $f : \mathbb{R}^m \to \mathbb{R}^m$ is a differentiable, invertible function (diffeomorphism);
- $\mathbf{z} = f(\mathbf{x})$, $\mathbf{x} = f^{-1}(\mathbf{z}) = g(\mathbf{z})$ (here $g = f^{-1}$).

Then

$$p(\mathbf{x}) = p(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x})) \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

$$p(\mathbf{z}) = p(\mathbf{x}) \left| \det \left(\frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) \right| = p(g(\mathbf{z})) \left| \det \left(\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} \right) \right|.$$

- $\mathbf{x}$ and $\mathbf{z}$ have the same dimensionality (lies in $\mathbb{R}^m$);
- $\left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \left| \det \left(\frac{\partial g^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \left| \det \left(\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} \right) \right|^{-1}$;
- $f(\mathbf{x}, \boldsymbol{\theta})$ can be a parametric function.

Recap of previous lecture

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

Flow definition

Normalizing flow is a *differentiable*, *invertible* mapping from data $\mathbf{x}$ to the noise $\mathbf{z}$.

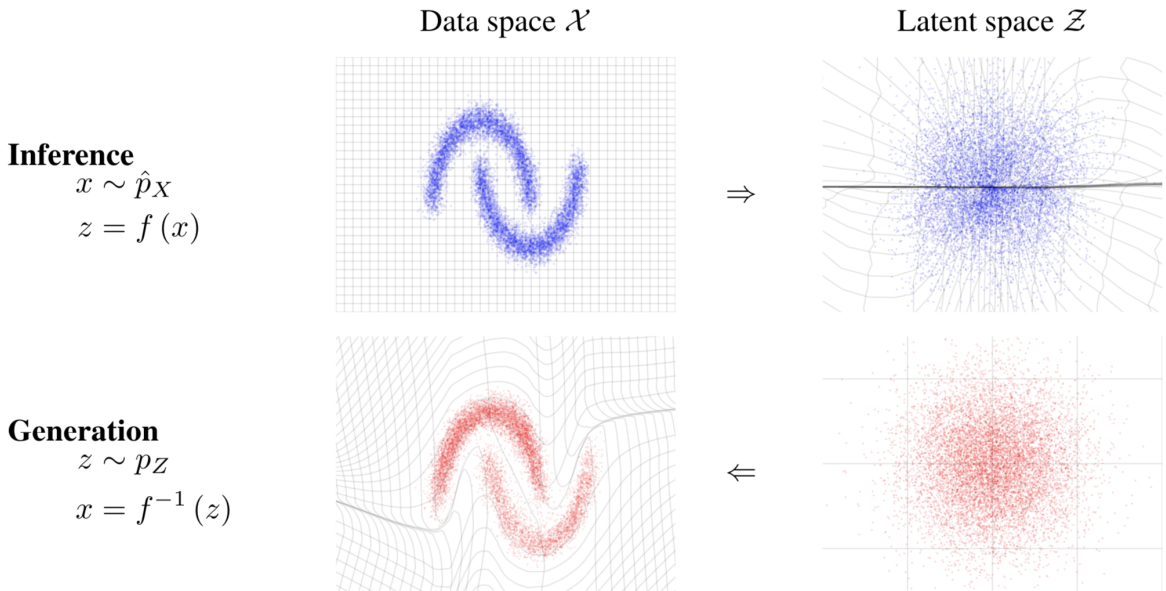
- "Normalizing" means that the inverse flow takes samples from $p(\mathbf{x})$ and normalizes them into samples from density $p(\mathbf{z})$.
- "Flow" refers to the trajectory followed by samples from $p(\mathbf{z})$ as they are transformed by the sequence of transformations

$$\mathbf{z} = f_K \circ \cdots \circ f_1(\mathbf{x}); \quad \mathbf{x} = f_1^{-1} \circ \cdots \circ f_K^{-1}(\mathbf{z}) = g_1 \circ \cdots \circ g_K(\mathbf{z})$$

$$p(\mathbf{x}) = p(f_K \circ \cdots \circ f_1(\mathbf{x})) \left| \det \left(\frac{\partial f_K \circ \cdots \circ f_1(\mathbf{x})}{\partial \mathbf{x}} \right) \right| =$$

$$= p(f_K \circ \cdots \circ f_1(\mathbf{x})) \prod_{k=1}^{K} \left| \det \left(\frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right|.$$

Recap of previous lecture



Flow likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

What we want

- Efficient computation of Jacobian $\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}}$;
- Efficient sampling from the base distribution $p(\mathbf{z})$;
- Efficient inversion of $f(\mathbf{x}, \boldsymbol{\theta})$.

Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

Planar Flows

$$g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{z} + \mathbf{u}\, h(\mathbf{w}^T \mathbf{z} + b).$$

- $\boldsymbol{\theta} = \{\mathbf{u}, \mathbf{w}, b\}$;
- h is a smooth element-wise non-linearity.

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

The transformation is invertible, for example, if

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

Sylvester flow: planar flow extension

$$g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{z} + \mathbf{A}\, h(\mathbf{B}\mathbf{z} + \mathbf{b}).$$

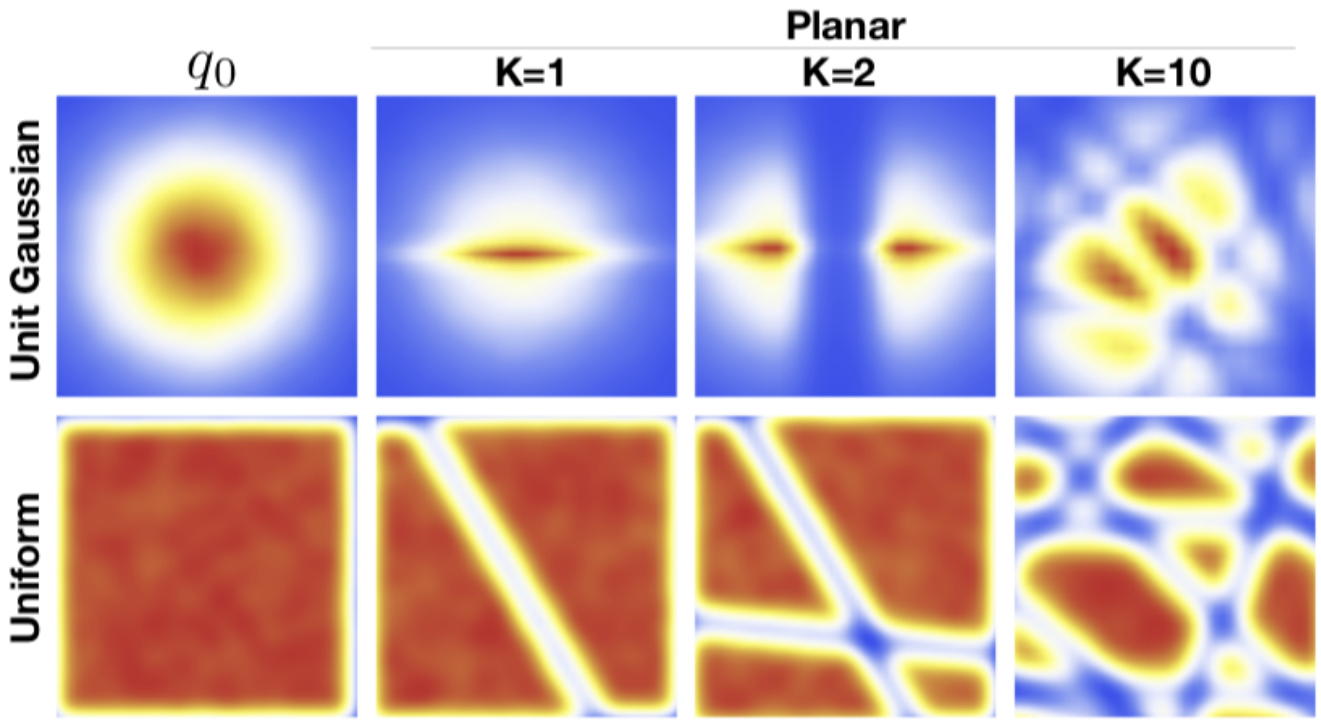
Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015
Berg R. et al. Sylvester normalizing flows for variational inference, 2018

Planar Flows

Composition of planar layers

$$\mathbf{z}_K = g_1 \circ \cdots \circ g_K(\mathbf{z}); \quad g_k = g(\mathbf{z}_k, \boldsymbol{\theta}_k).$$

Expressiveness of planar flows



Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015

Jacobian structure

Flow likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

- What is a determinant of a diagonal matrix?

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) = (f_1(x_1, \boldsymbol{\theta}), \ldots, f_m(x_m, \boldsymbol{\theta})).$$

$$\log \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right| = \log \left| \prod_{i=1}^{m} f_i'(x_i, \boldsymbol{\theta}) \right| = \sum_{i=1}^{m} \log \left| f_i'(x_i, \boldsymbol{\theta}) \right|.$$

- What is a determinant of a triangular matrix?
  Let z_i depend only on $\mathbf{x}_{1:i}$ (or without loss of generality x_i depends on $\mathbf{z}_{1:i}$).
  What is the Jacobian of such a transformation?

NICE

Coupling layer

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

- $c : \mathbb{R}^d \to \mathbb{R}^k$ – coupling function (do not need to be invertible);
- $\tau : \mathbb{R}^{m-d} \times c(\mathbb{R}^d) \to \mathbb{R}^{m-d}$ – coupling law.
- $$\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) = \det\begin{pmatrix} \mathbf{I}_d & 0_{d\times m-d} \\ \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{1:d}} & \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}} \end{pmatrix} = \det\left(\frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}}\right)$$

Coupling function $c(\cdot)$

Any complex function (without restrictions). For example, neural network.

Dinh L., Krueger D., Bengio Y. NICE: Non-linear Independent Components Estimation, 2014

NICE

Coupling layer

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

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

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

Jacobian

$$\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) = \det\begin{pmatrix} \mathbf{I}_d & 0_{d \times m-d} \\ \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{1:d}} & \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}} \end{pmatrix} = \det\left(\frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}}\right)$$

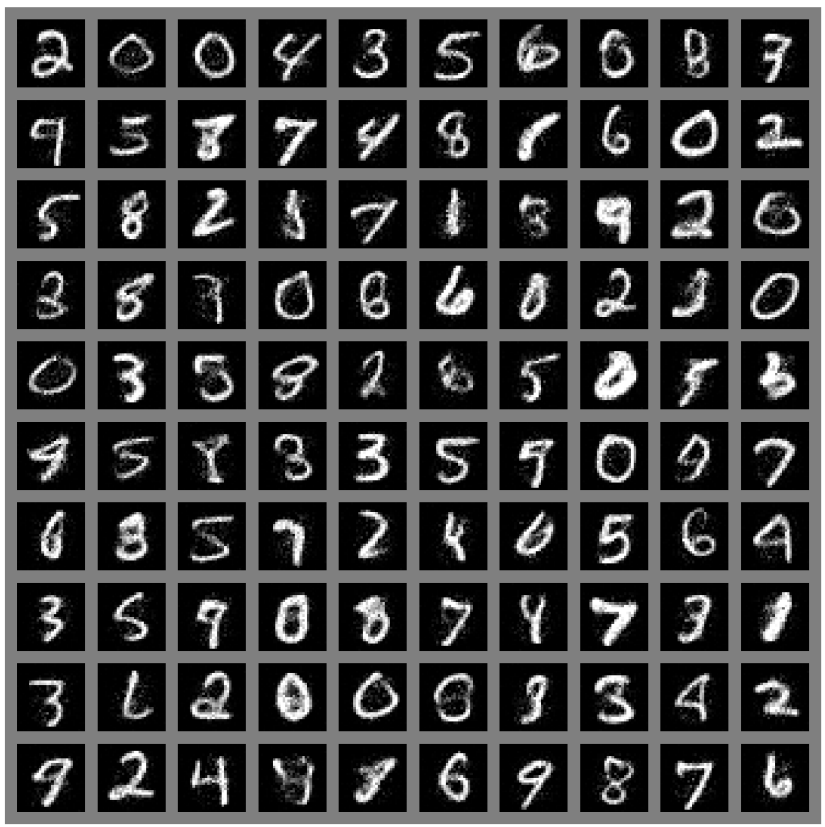
What is the Jacobian for the additive coupling law?
In this case, the transformation is *volume preserving*.

Dinh L., Krueger D., Bengio Y. NICE: Non-linear Independent Components Estimation, 2014

NICE

To obtain a more flexible class of distributions, stack more coupling layers (with different ordering of components!).

Flow samples



(a) Model trained on MNIST

(b) Model trained on TFD

Dinh L., Krueger D., Bengio Y. NICE: Non-linear Independent Components Estimation, 2014

RealNVP

Affine coupling law

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

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

Jacobian

$$\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) = \det\begin{pmatrix} \mathbf{I}_d & 0_{d \times m-d} \\ \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{1:d}} & \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}} \end{pmatrix} = \prod_{i=1}^{m-d} \exp(c_1(\mathbf{x}_{1:d}, \boldsymbol{\theta})_i).$$

Non-Volume Preserving (the determinant of Jacobian $\neq 0$).

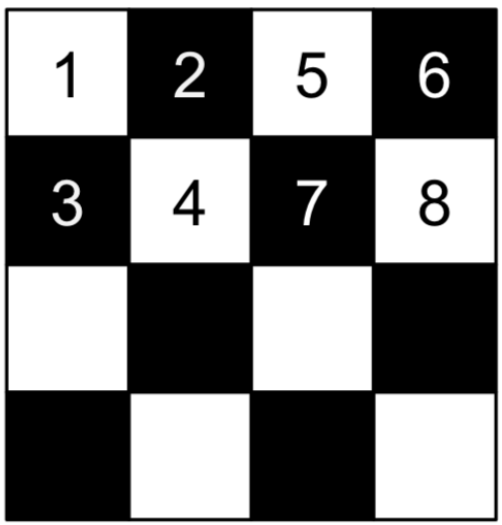
Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

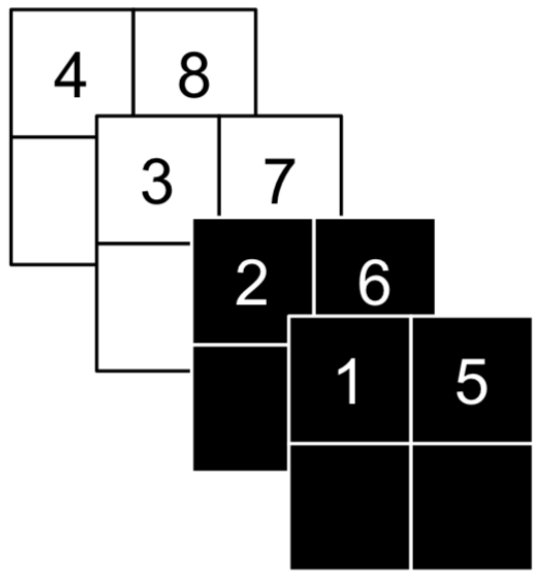
RealNVP

Affine coupling law

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

How to choose variable partitioning for images?



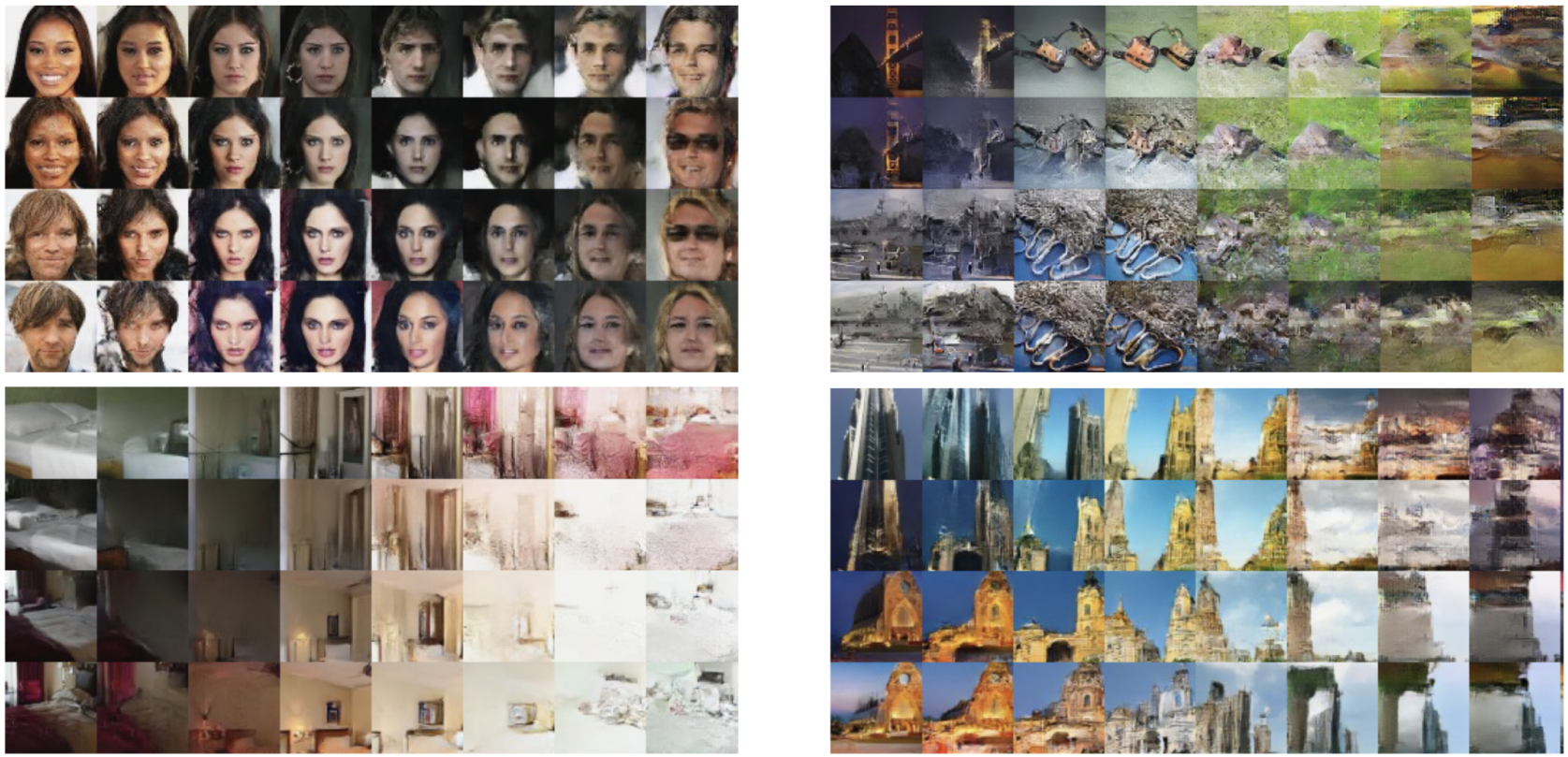


Masked convolutions are used to define ordering.

Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

RealNVP

Flow samples



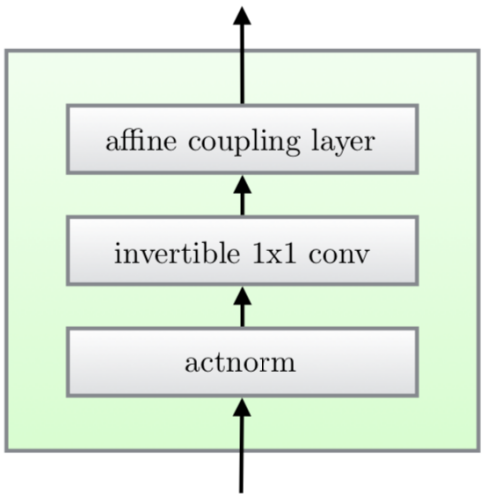
Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

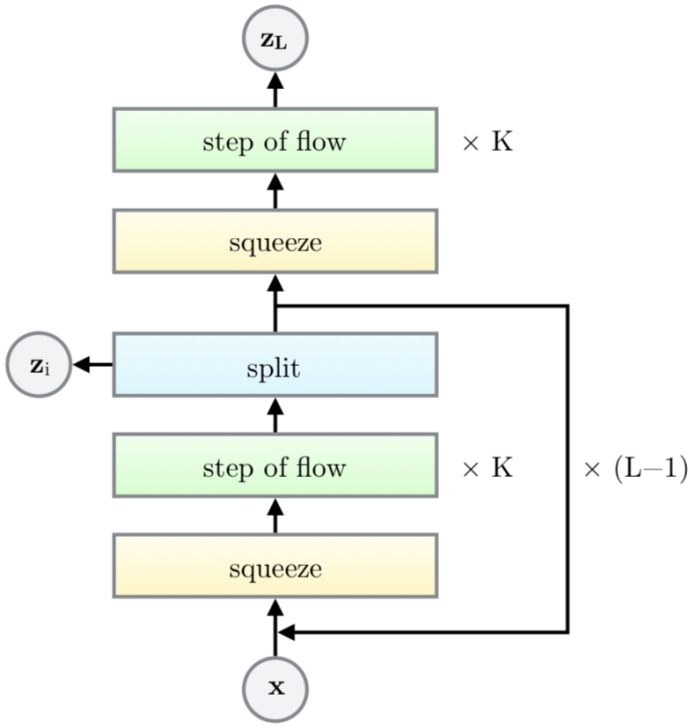
Glow, 2018

Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Glow, 2018

Model architecture





- Affine coupling layer (already known).
- Invertible 1x1 conv (contribution).
- Actnorm (architectural detail).

Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Glow, 2018

NICE

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

- First step is a **split** operator which decouples a variable into 2 subparts: $\mathbf{x}_1$ and $\mathbf{x}_2$ (usualy channel-wise). The order of decoupling should be manually changed between layers.
- Could we use a more general operator?
- Let use a rotation matrix via 1x1 invertible convolution.
  $\mathbf{W} \in \mathbb{R}^{c \times c}$ - kernel of 1x1 convolution with c input and c output channels.
  The computational complexity of computing or differentiating $\det(\mathbf{W})$ is $O(c^3)$.

Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Glow, 2018

Basic flow operations

Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

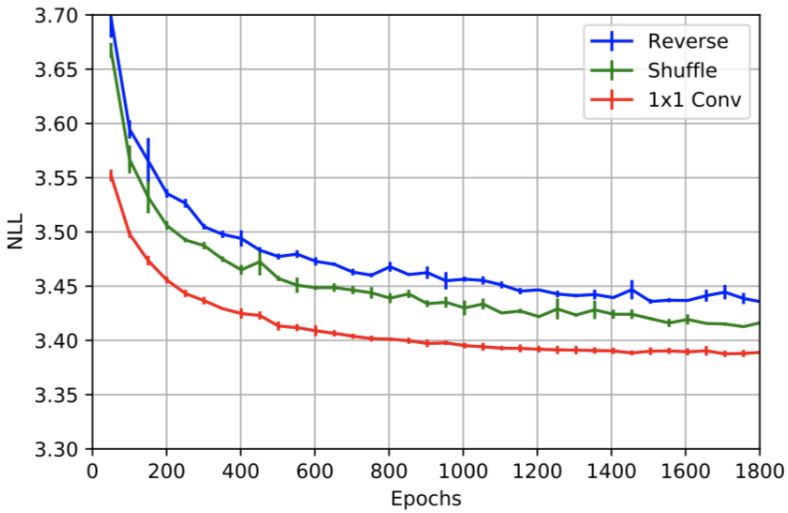
Glow, 2018

Invertible 1x1 conv

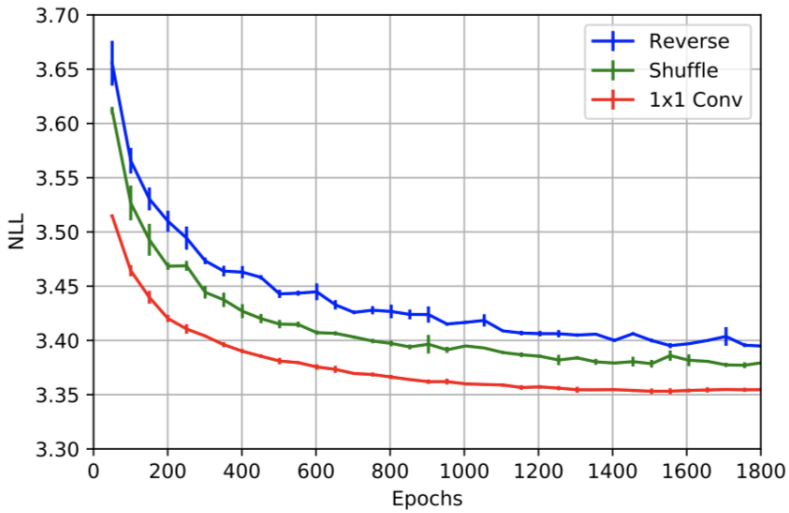
Cost to compute $\det(\mathbf{W})$ is $O(c^3)$. LU-decomposition reduces the cost to $O(c)$:

$$\mathbf{W} = \mathbf{PL}(\mathbf{U} + \text{diag}(\mathbf{s})),$$

where $\mathbf{P}$ is a permutation matrix, $\mathbf{L}$ is a lower triangular matrix with ones on the diagonal, $\mathbf{U}$ is an upper triangular matrix with zeros on the diagonal, and $\mathbf{s}$ is a vector.



(a) Additive coupling.



(b) Affine coupling.

Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Glow, 2018

Face interpolation

Face attributes manipulation

(a) Smiling

(b) Pale Skin

(c) Blond Hair

(d) Narrow Eyes

Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Likelihood-based models

Exact likelihood evaluation

- Autoregressive models (PixelCNN, WaveNet);
- Flow models (NICE, RealNVP, Glow).

Approximate likelihood evaluation

- Latent variable models (VAE).

What are the pros and cons of each of them?

VAE recap

ELBO

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

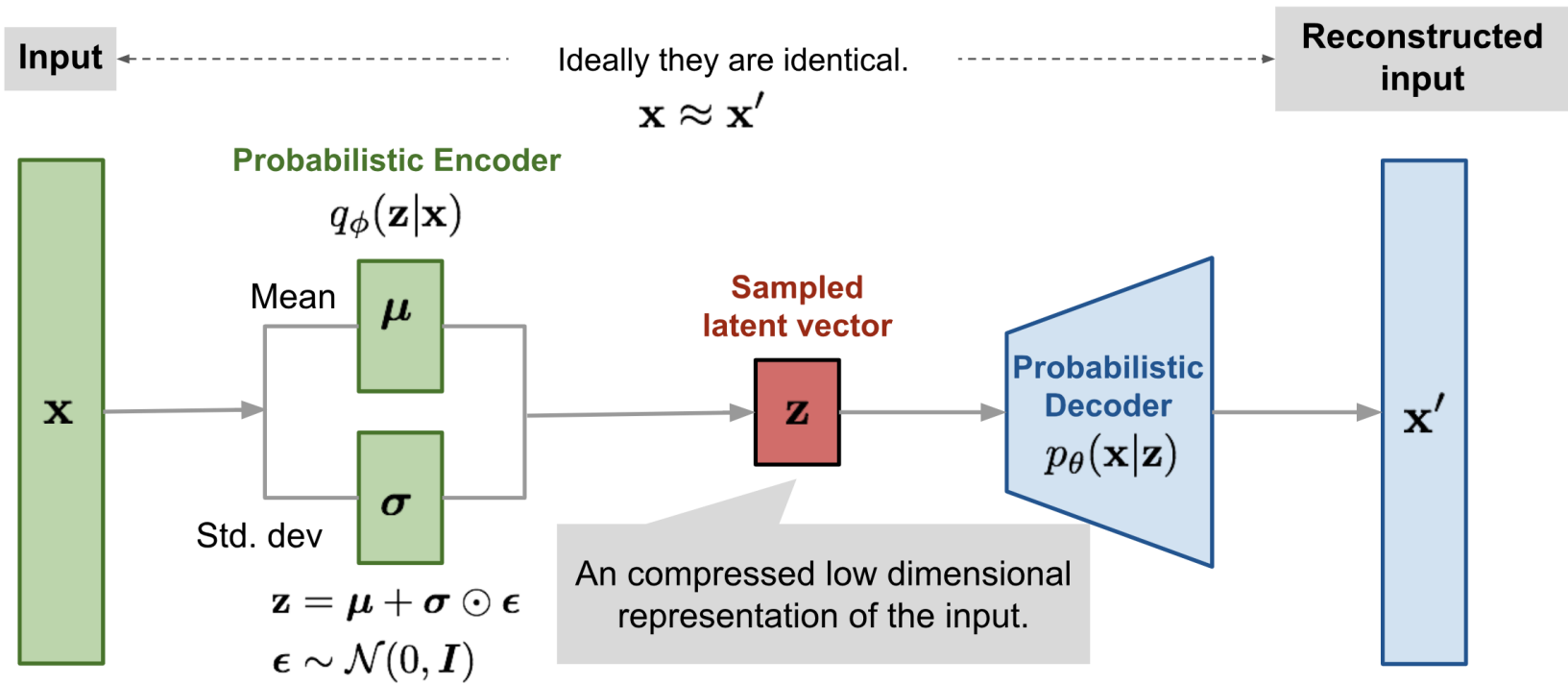


image credit:
https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

VAE limitations

- Poor variational posterior distribution (encoder)

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

- Poor prior distribution

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

- Poor probabilistic model (decoder)

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\mathbf{z})) \quad (\text{or Softmax}(\pi(\mathbf{z}))).$$

- Loose lower bound

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

Variational posterior

ELBO

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

- In E-step of EM-algorithm we wish $KL(q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi})||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) = 0$.
  (In this case the lower bound is tight $\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta})$).
- Normal variational distribution $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{x}))$ is poor (e.g. has only one mode).
- Flows models convert a simple base distribution to a compex one using invertible transformation with simple Jacobian. How to use flows in VAE?

Flows in VAE

Apply a sequence of transformations to the random variable

$$\mathbf{z}_0 \sim q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}^2_\phi(\mathbf{x})).$$

Here, $q(\mathbf{z}|\mathbf{x}, \phi)$ (which is a VAE encoder) plays a role of a base distribution.

$$\mathbf{z}_0 \xrightarrow{g_1} \mathbf{z}_1 \xrightarrow{g_2} \dots \xrightarrow{g_K} \mathbf{z}_K, \quad \mathbf{z}_K = g(\mathbf{z}_0), \quad g = g_K \circ \dots \circ g_1.$$

Each g_k is a flow transformation (e.g. planar, coupling layer) parameterized by ϕ_k.

$$\log q_K(\mathbf{z}_K|\mathbf{x}, \phi, \{\phi_k\}_{k=1}^K) = \log q(\mathbf{z}_0|\mathbf{x}, \phi) - \sum_{k=1}^K \log \left| \det \left(\frac{\partial g_k(\mathbf{z}_{k-1}, \phi_k)}{\partial \mathbf{z}_{k-1}} \right) \right|.$$

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015

Flows in VAE

ELBO

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

Flow model in latent space

$$\log q_K(\mathbf{z}_K|\mathbf{x}, \boldsymbol{\phi}_*) = \log q(\mathbf{z}_0|\mathbf{x}, \boldsymbol{\phi}) - \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial g_k(\mathbf{z}_{k-1}, \boldsymbol{\phi}_k)}{\partial \mathbf{z}_{k-1}} \right) \right|.$$

Let use $q_K(\mathbf{z}_K|\mathbf{x}, \boldsymbol{\phi}_*)$, $\boldsymbol{\phi}_* = \{\boldsymbol{\phi}, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K\}$ as a variational distribution. Here $\boldsymbol{\phi}$ – encoder parameters, $\{\boldsymbol{\phi}_k\}_{k=1}^K$ – flow parameters.

- Encoder outputs base distribution $q(\mathbf{z}_0|\mathbf{x}, \boldsymbol{\phi})$.
- Flow model $\mathbf{z}_K = g(\mathbf{z}_0, \{\boldsymbol{\phi}_k\}_{k=1}^K)$ transforms the base distribution $q(\mathbf{z}_0|\mathbf{x}, \boldsymbol{\phi})$ to the distribution $q_K(\mathbf{z}_K|\mathbf{x}, \boldsymbol{\phi}_*)$.
- Distribution $q_K(\mathbf{z}_K|\mathbf{x}, \boldsymbol{\phi}_*)$ is used as a variational distribution for ELBO maximization.

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015

Flows in VAE

Flow model in latent space

$$\log q_K(\mathbf{z}_K|\mathbf{x},\phi_*) = \log q(\mathbf{z}_0|\mathbf{x},\phi) - \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial g_k(\mathbf{z}_{k-1},\phi_k)}{\partial \mathbf{z}_{k-1}} \right) \right|.$$

ELBO objective

$$\begin{aligned}
\mathcal{L}(\phi,\theta) &= \mathbb{E}_{q_K(\mathbf{z}_K|\mathbf{x},\phi_*)} \log \frac{p(\mathbf{x},\mathbf{z}_K|\theta)}{q_K(\mathbf{z}_K|\mathbf{x},\phi_*)} \\
&= \mathbb{E}_{q_K(\mathbf{z}_K|\mathbf{x},\phi_*)} \left[\log p(\mathbf{x},\mathbf{z}_K|\theta) - \log q_K(\mathbf{z}_K|\mathbf{x},\phi_*) \right] \\
&= \mathbb{E}_{q_K(\mathbf{z}_K|\mathbf{x},\phi_*)} \log p(\mathbf{x}|\mathbf{z}_K,\theta) - KL(q_K(\mathbf{z}_K|\mathbf{x},\phi_*)||p(\mathbf{z}_K)).
\end{aligned}$$

The second term in ELBO is reverse KL divergence. Planar flows was originally proposed for variational inference in VAE.

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015

Flows in VAE

Variational distribution

$$\log q_K(\mathbf{z}_K|\mathbf{x},\phi_*) = \log q(\mathbf{z}_0|\mathbf{x},\phi) - \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial g_k(\mathbf{z}_{k-1},\phi_k)}{\partial \mathbf{z}_{k-1}} \right) \right|.$$

ELBO objective

$$\begin{aligned}
\mathcal{L}(\phi,\theta) &= \mathbb{E}_{q_K(\mathbf{z}_K|\mathbf{x},\phi_*)} \left[\log p(\mathbf{x},\mathbf{z}_K|\theta) - \log q_K(\mathbf{z}_K|\mathbf{x},\phi_*) \right] \\
&= \mathbb{E}_{q(\mathbf{z}_0|\mathbf{x},\phi)} \left[\log p(\mathbf{x},\mathbf{z}_K|\theta) - \log q_K(\mathbf{z}_K|\mathbf{x},\phi_*) \right] \Big|_{\mathbf{z}_K = g(\mathbf{z}_0,\{\phi_k\}_{k=1}^K)} \\
&= \mathbb{E}_{q(\mathbf{z}_0|\mathbf{x},\phi)} \Bigg[\log p(\mathbf{x},\mathbf{z}_K|\theta) - \log q(\mathbf{z}_0|\mathbf{x},\phi) + \\
&\quad + \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial g_k(\mathbf{z}_{k-1},\phi_k)}{\partial \mathbf{z}_{k-1}} \right) \right| \Bigg].
\end{aligned}$$

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015

Flows in VAE

Variational distribution

$$\log q_K(\mathbf{z}_K|\mathbf{x},\phi_*) = \log q(\mathbf{z}_0|\mathbf{x},\phi) - \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial g_k(\mathbf{z}_{k-1},\phi_k)}{\partial \mathbf{z}_{k-1}} \right) \right|.$$

ELBO objective

$$\mathcal{L}(\phi,\theta) = \mathbb{E}_{q(\mathbf{z}_0|\mathbf{x},\phi)} \Bigg[\log p(\mathbf{x},\mathbf{z}_K|\theta) - \log q(\mathbf{z}_0|\mathbf{x},\phi) + \\ + \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial g_k(\mathbf{z}_{k-1},\phi_k)}{\partial \mathbf{z}_{k-1}} \right) \right| \Bigg].$$

- Obtain samples $\mathbf{z}_0$ from the encoder.
- Apply flow model $\mathbf{z}_K = g(\mathbf{z}_0, \{\phi_k\}_{k=1}^K)$.
- Compute likelihood for $\mathbf{z}_K$ using the decoder, base distribution for $\mathbf{z}_0$ and the Jacobian.
- We do not need inverse flow function, if we use flows in variational inference.

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015

Summary

- Flow models require tractable Jacobian.
- Planar flow is a simple form of an invertible flow model (Sylvester flows are their extension).
- The NICE/RealNVP model is a more powerful type of flow that use coupling layers.
- Glow model is a first flow model with superior results.
- Flows could be used in variational inference to create powerful variational distribution.