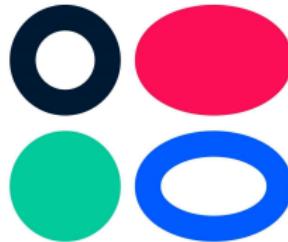


Deep Generative Models

Lecture 5

Roman Isachenko



Ozon Masters

Spring, 2021

Recap of previous lecture

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

Forward KL

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

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

Reverse KL

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

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 $p(\mathbf{x})$;
- ▶ $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ 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);
- ▶ $\left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \left| \det \left(\frac{\partial g^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \left| \det \left(\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} \right) \right|^{-1}$;
- ▶ $f(\mathbf{x}, \theta)$ 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

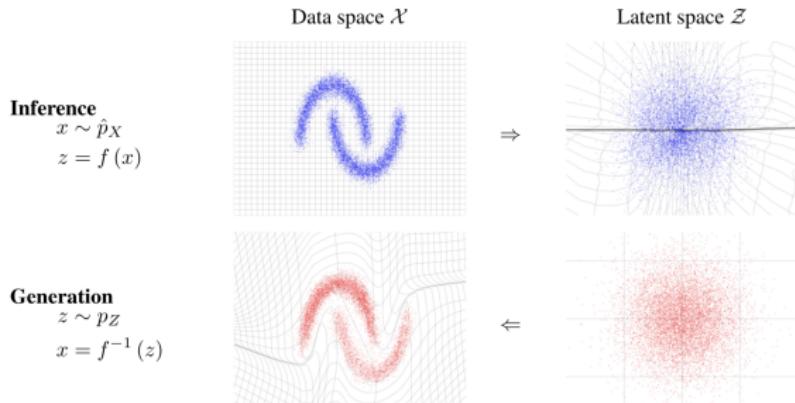
Normalizing flow is a *differentiable, invertible* mapping from data \mathbf{x} to the noise \mathbf{z} .

- ▶ "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

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

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

Recap of previous lecture



Flow likelihood

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

What we want

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

Planar Flows

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

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

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

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

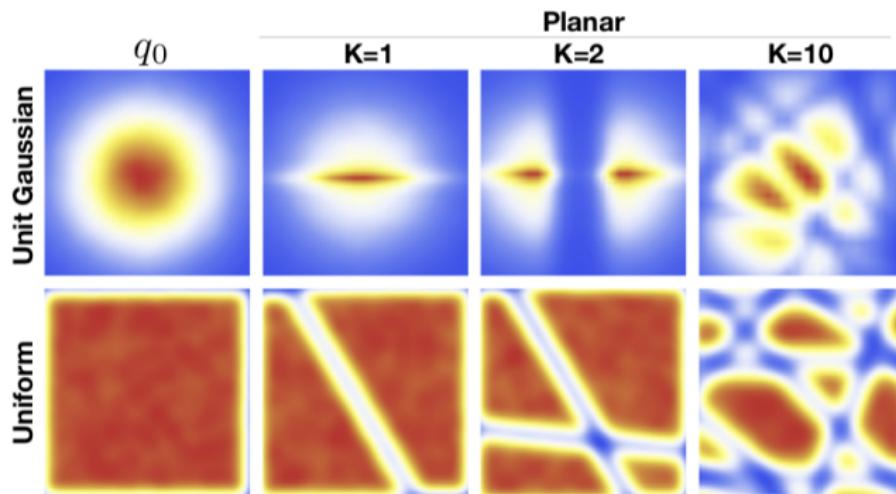
$$g(\mathbf{z}, \theta) = \mathbf{z} + \mathbf{A} h(\mathbf{B} \mathbf{z} + \mathbf{b}).$$

Planar Flows

Composition of planar layers

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

Expressiveness of planar flows



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?

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

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

- ▶ 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?

Coupling layer

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

- ▶ $c : \mathbb{R}^d \rightarrow \mathbb{R}^k$ – coupling function (do not need to be invertible);
- ▶ $\tau : \mathbb{R}^{m-d} \times c(\mathbb{R}^d) \rightarrow \mathbb{R}^{m-d}$ – 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.

Coupling layer

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

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

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

Jacobian

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

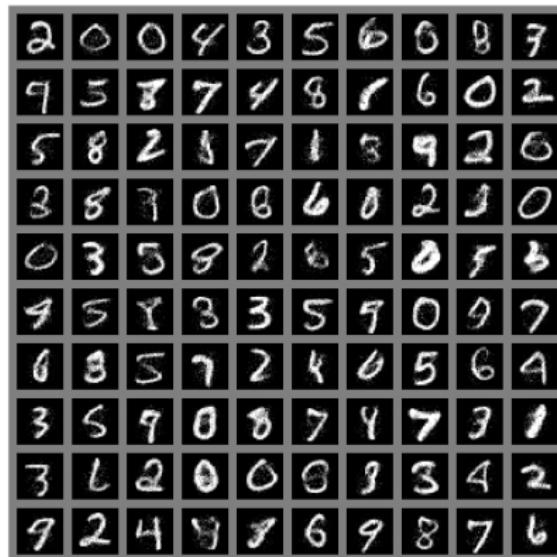
What is the Jacobian for the additive coupling law?

In this case, the transformation is *volume preserving*.

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

RealNVP

Affine coupling law

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

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

Jacobian

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

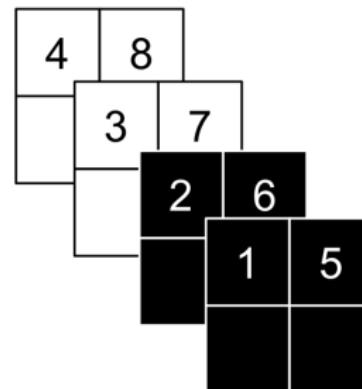
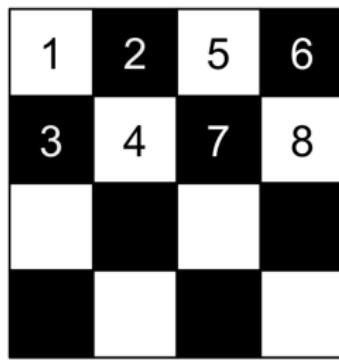
Non-Volume Preserving (the determinant of Jacobian $\neq 0$).

RealNVP

Affine coupling law

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

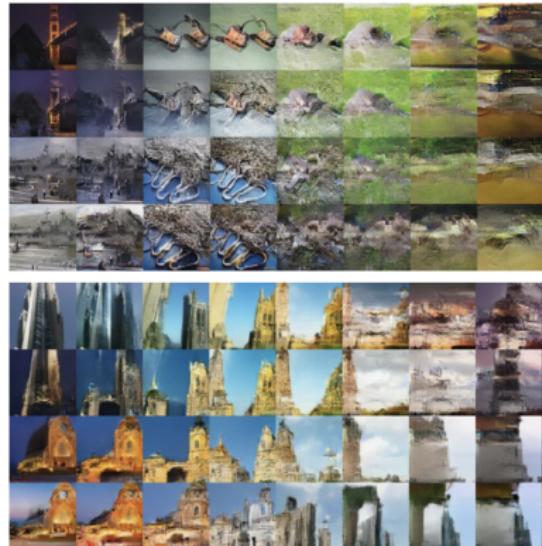
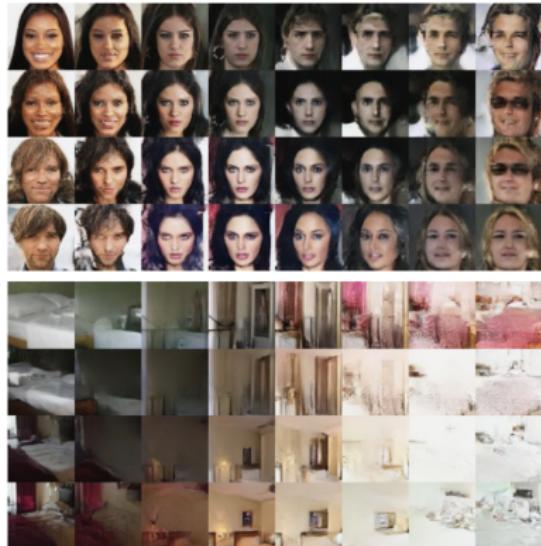
How to choose variable partitioning for images?



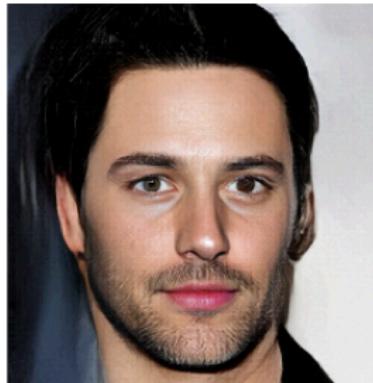
Masked convolutions are used to define ordering.

RealNVP

Flow samples

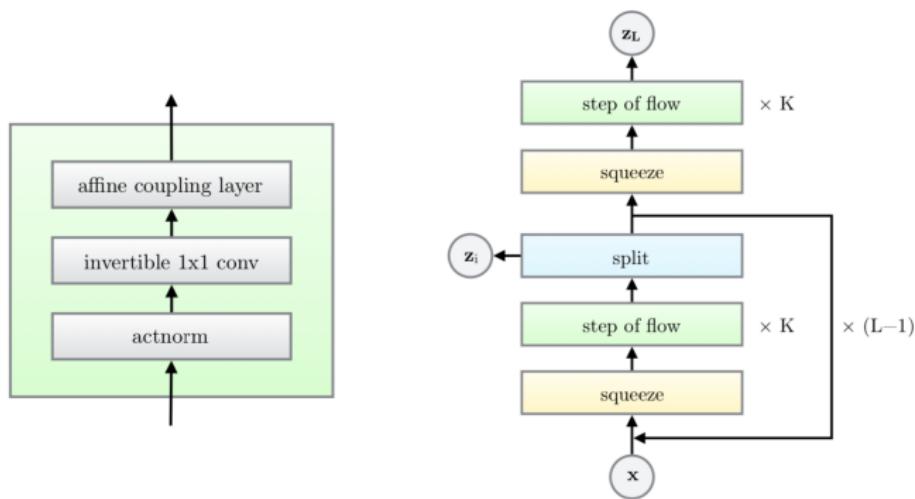


Glow, 2018



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

Model architecture



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

NICE

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

- ▶ First step is a **split** operator which decouples a variable into 2 subparts: \mathbf{x}_1 and \mathbf{x}_2 (usually channel-wise). The order of decoupling should be manually changed between layers.
- ▶ Could we use a more general operator?
- ▶ Let's 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 $O(c^3)$.

Basic flow operations

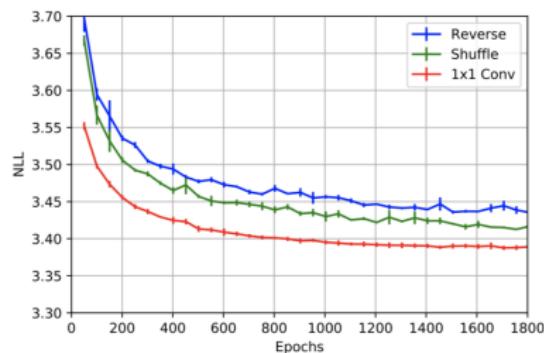
Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i, j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i, j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \text{sum}(\log \mathbf{s})$
Invertible 1×1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

Invertible 1x1 conv

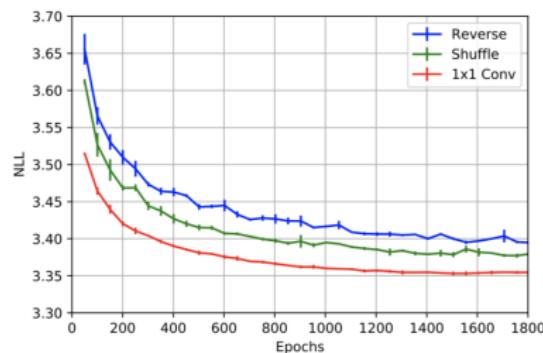
Cost to compute $\det(\mathbf{W})$ is $O(c^3)$. LU-decomposition reduces the cost to $O(c)$:

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

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.

Glow, 2018

Face interpolation



Face attributes manipulation



(a) Smiling

(b) Pale Skin



(c) Blond Hair

(d) Narrow Eyes

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

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

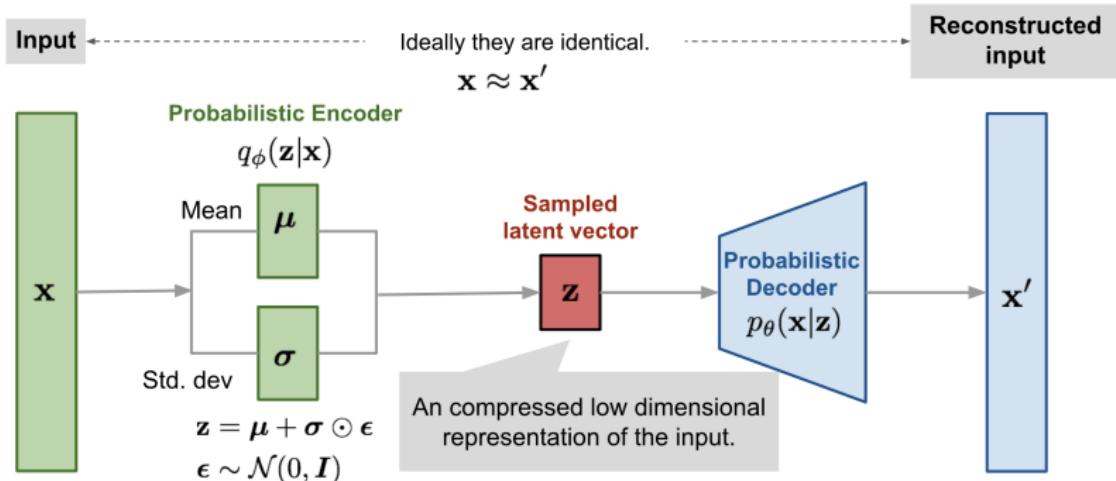


image credit:

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

VAE limitations

- ▶ Poor variational posterior distribution (encoder)

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

- ▶ Poor prior distribution

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

- ▶ Poor probabilistic model (decoder)

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

- ▶ Loose lower bound

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

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
 $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}_{\boldsymbol{\phi}}^2(\mathbf{x}))$ is poor (e.g. has only one mode).
- ▶ Flows models convert a simple base distribution to a complex 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

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

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 .

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

Flows in VAE

ELBO

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

Flow model in latent space

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

Let use $q_K(\mathbf{z}_K|\mathbf{x}, \phi_*)$, $\phi_* = \{\phi, \phi_1, \dots, \phi_K\}$ as a variational distribution. Here ϕ – encoder parameters, $\{\phi_k\}_{k=1}^K$ – flow parameters.

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

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

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

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

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

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

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

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

- ▶ 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.

Dequantization

- ▶ Images are discrete data, pixels lies in the $[0, 255]$ integer domain (the model is $P(\mathbf{x}|\theta) = \text{Categorical}(\boldsymbol{\pi}(\theta))$).
- ▶ Flow is a continuous model (it works with continuous data \mathbf{x}).

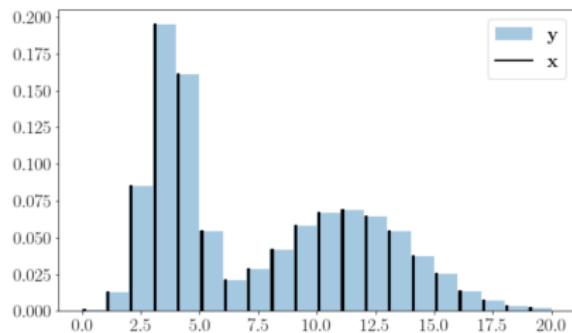
Fitting a continuous density model to discrete data, could produce a degenerate solution with all probability mass on discrete values.
How to convert discrete data distribution to the continuous one?

Uniform dequantization

$$\mathbf{x} \sim \text{Categorical}(\boldsymbol{\pi})$$

$$\mathbf{u} \sim U[0, 1]$$

$$\mathbf{y} = \mathbf{x} + \mathbf{u} \sim \text{Continuous}$$



Uniform dequantization

Statement

Fitting continuous model $p(\mathbf{y}|\theta)$ on uniformly dequantized data $\mathbf{y} = \mathbf{x} + \mathbf{u}$, $\mathbf{u} \sim U[0, 1]$ is equivalent to maximization of a lower bound on the log-likelihood for a discrete model:

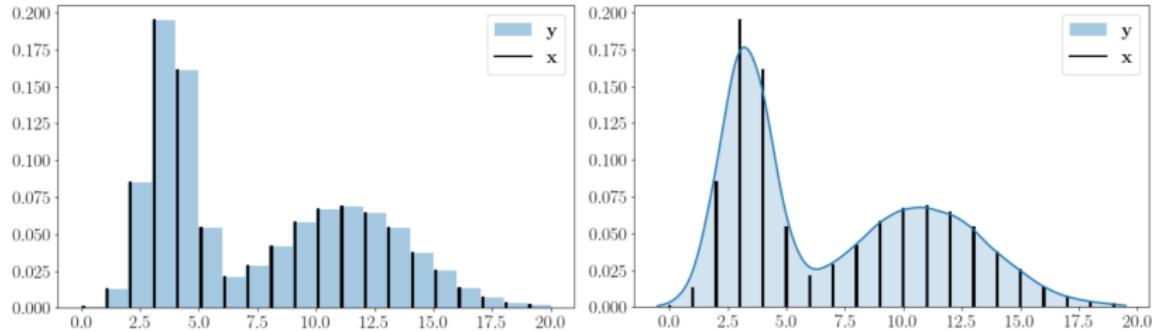
$$P(\mathbf{x}|\theta) = \int_{U[0,1]} p(\mathbf{x} + \mathbf{u}|\theta) d\mathbf{u}$$

Thus, maximizing the log-likelihood of the continuous model on \mathbf{y} cannot lead to the collapsing onto the discrete data (objective is bounded above by the log-likelihood of a discrete model).

Proof

$$\begin{aligned} \log P(\mathbf{x}|\theta) &= \log \int_{U[0,1]} p(\mathbf{x} + \mathbf{u}|\theta) d\mathbf{u} \geq \\ &\geq \int_{U[0,1]} \log p(\mathbf{x} + \mathbf{u}|\theta) d\mathbf{u} = \log p(\mathbf{y}|\theta). \end{aligned}$$

Variational dequantization



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

How to make the smooth dequantization?

Flow++

Variational dequantization

Introduce variational dequantization noise distribution $q(\mathbf{u}|\mathbf{x})$ and treat it as an approximate posterior.

Variational lower bound

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

Uniform dequantization bound

$$\begin{aligned}\log P(\mathbf{x}|\theta) &= \log \int_{U[0,1]} p(\mathbf{x} + \mathbf{u}|\theta) d\mathbf{u} \geq \\ &\geq \int_{U[0,1]} \log p(\mathbf{x} + \mathbf{u}|\theta) d\mathbf{u} = \log p(\mathbf{y}|\theta).\end{aligned}$$

Flow++

Variational lower bound

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

Let $\mathbf{u} = h(\epsilon, \phi)$ is a flow model with base distribution $\epsilon \sim p(\epsilon) = \mathcal{N}(0, \mathbf{I})$:

$$q(\mathbf{u}|\mathbf{x}) = p(h^{-1}(\mathbf{u}, \phi)) \cdot \left| \det \frac{\partial h^{-1}(\mathbf{u}, \phi)}{\partial \mathbf{u}} \right|.$$

Then

$$\log P(\mathbf{x}|\theta) \geq \mathcal{L}(\phi, \theta) = \int p(\epsilon) \log \left(\frac{p(\mathbf{x} + h(\epsilon, \phi)|\theta)}{p(\epsilon) \cdot \left| \det \frac{\partial h(\epsilon, \phi)}{\partial \epsilon} \right|^{-1}} \right) d\epsilon.$$

Variational lower

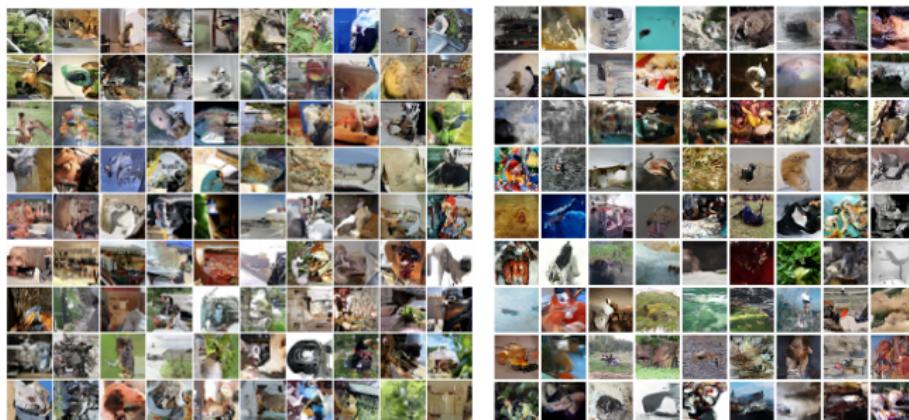
$$\log P(\mathbf{x}|\theta) \geq \int p(\epsilon) \log \left(\frac{p(\mathbf{x} + h(\epsilon, \phi))}{p(\epsilon) \cdot \left| \det \frac{\partial h(\epsilon, \phi)}{\partial \epsilon} \right|^{-1}} \right) d\epsilon.$$

- ▶ If $p(\mathbf{x} + \mathbf{u}|\theta)$ is also a flow model, it is straightforward to calculate stochastic gradient of this ELBO.
- ▶ Uniform dequantization is a special case of variational dequantization ($q(\mathbf{u}|\mathbf{x}) = U[0, 1]$). The gap between $\log P(\mathbf{x}|\theta)$ and the derived ELBO is $KL(q(\mathbf{u}|\mathbf{x})||p(\mathbf{u}|\mathbf{x}))$.
- ▶ In the case of uniform dequantization the model unnaturally places uniform density over each hypercube $\mathbf{x} + U[0, 1]$ due to inexpressive distribution q .

Flow++

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



(a) PixelCNN

(b) Flow++

Summary

- ▶ Flow models require tractable Jacobian.
- ▶ Planar flows 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.
- ▶ Glow model is first flow model with superior results.
- ▶ Flows could be used in variational inference to create powerful variational distribution.
- ▶ To apply continuous model to discrete distribution the standard practice is to dequantize data at first.
- ▶ Uniform dequantization is the simplest form of dequantization. Variational dequantization is more natural type that was proposed in Flow++ model.