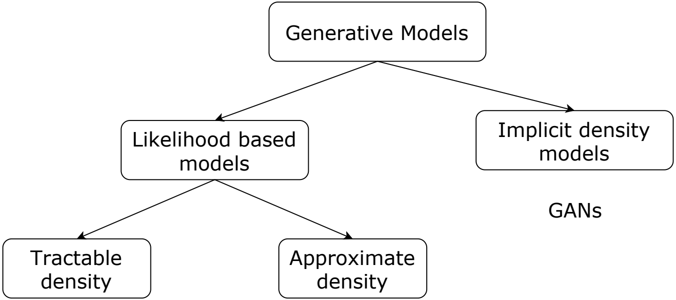
Deep Generative Models

Roman Isachenko

Moscow Institute of Physics and Technology

2019

Generative models zoo



- Autoregressive models
- Flow models

Latent variable models

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 (latent) variable;
- $p(\mathbf{x}|\mathbf{t})$ – likelihood;
- $p(\mathbf{x})$ – evidence;
- $p(\mathbf{t})$ – prior.

Variational Lower Bound

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

Empirical Lower BOund (ELBO)

$$\log p(\mathbf{X}) = \log \frac{p(\mathbf{X}, \mathbf{T})}{p(\mathbf{T}|\mathbf{X})} =$$

$$= \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{p(\mathbf{T}|\mathbf{X})} d\mathbf{T} = \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})q(\mathbf{T})}{p(\mathbf{T}|\mathbf{X})q(\mathbf{T})} d\mathbf{T} =$$

$$= \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{q(\mathbf{T})} d\mathbf{T} + \int q(\mathbf{T}) \log \frac{q(\mathbf{T})}{p(\mathbf{T}|\mathbf{X})} d\mathbf{T} =$$

$$= \mathcal{L}(q) + KL(q(\mathbf{T})||p(\mathbf{T}|\mathbf{X})) \geq \mathcal{L}(q).$$

Mean field approximation

Assumption

$$q(\mathbf{T}) = \prod_{i=1}^{k} q_i(\mathbf{T}_i).$$

Empirical Lower BOund (ELBO)

$$\mathcal{L}(q) = \int q(\mathbf{T}) \log \frac{p(\mathbf{X}, \mathbf{T})}{q(\mathbf{T})} d\mathbf{T} = \int \prod_{i=1}^{k} q_i(\mathbf{T}_i) \log \frac{p(\mathbf{X}, \mathbf{T})}{\prod_{i=1}^{k} q_i(\mathbf{T}_i)} \prod_{i=1}^{k} d\mathbf{T}_i =$$

$$= \int \prod_{i=1}^{k} q_i \log p(\mathbf{X}, \mathbf{T}) \prod_{i=1}^{k} d\mathbf{T}_i - \sum_{i=1}^{k} \int \prod_{j=1}^{k} q_j \log q_i \prod_{i=1}^{k} d\mathbf{T}_i =$$

$$= \int q_j \left[\int \log p(\mathbf{X}, \mathbf{T}) \prod_{i \neq j} q_i d\mathbf{T}_i \right] d\mathbf{T}_j -$$

$$- \int q_j \log q_j d\mathbf{T}_j + \text{const}(q_j) \to \max_{q_j}$$

Mean field approximation

$$\mathcal{L}(q) = \int q_j \left[\int \log p(\mathbf{X}, \mathbf{T}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right] d\mathbf{T}_j - \int q_j \log q_j d\mathbf{T}_j + \text{const}(q_j) =$$

$$= \int q_j \log \hat{p}(\mathbf{X}, \mathbf{T}_j) d\mathbf{T}_j - \int q_j \log q_j d\mathbf{T}_j + \text{const}(q_j) \to \max_{q_j}$$

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

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

$$\mathcal{L}(q) = \int q_j(\mathbf{T}_j) \log \hat{p}(\mathbf{X}, \mathbf{T}_j) d\mathbf{T}_j - \int q_j(\mathbf{T}_j) \log q_j(\mathbf{T}_j) d\mathbf{T}_j + \text{const}(q_j) =$$

$$= KL(q_j(\mathbf{T}_j) || \hat{p}(\mathbf{X}, \mathbf{T}_j)) + \text{const}(q_j) \to \max_{q_j}.$$

Mean field approximation

ELBO

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

Solution

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

Let use factorization on two parts: $\mathbf{T} = \{\mathbf{Z}, \boldsymbol{\theta}\}$.

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

- Initialize $\boldsymbol{\theta}^*$;
- E-step

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

- M-step

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta});$$

- Repeat E-step and M-step until convergence.

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 model with latent variables and tractable likelihood?

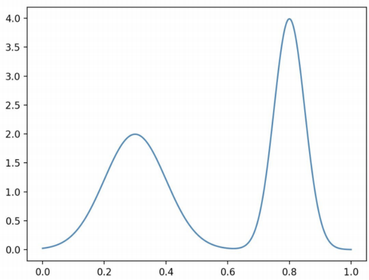
Flows intuition

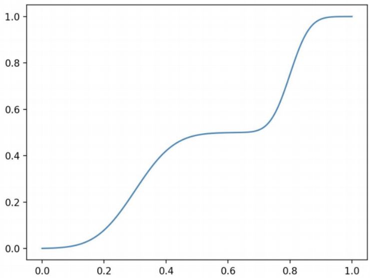
Let X be a random variable with density $p_X(x)$. Then

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

Hence

$$Z \sim U[0,1]; \quad X = F^{-1}(Z) \quad X \sim p(x).$$





---

https://sites.google.com/view/berkeley-cs294-158-sp19/home

Change of variables

Theorem

Let

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

Then

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

Note

- $\mathbf{x}$ and $\mathbf{z}$ have the same dimensionality;
- $\left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \left| \det \left(\frac{\partial g^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \left| \det \left(\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} \right) \right|^{-1}$.

Fitting flows

MLE problem

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{X}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\theta}).$$

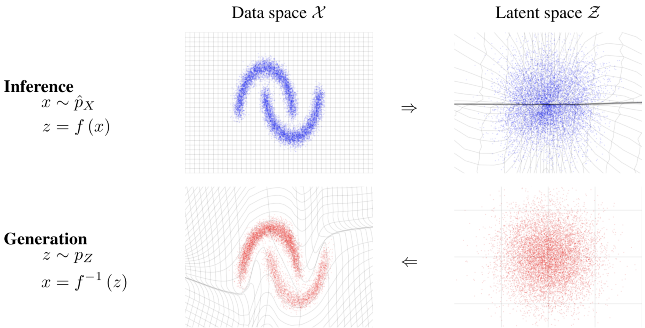
Challenge

$p(\mathbf{x}|\boldsymbol{\theta})$ could be intractable.

Fitting flow to solve MLE

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

Flows



- Likelihood is given by $\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta})$ and change of variables.
- Sampling of $\mathbf{x}$ is performed by sampling from a base distribution $p(\mathbf{z})$ and applying $\mathbf{x} = f^{-1}(\mathbf{z}, \boldsymbol{\theta}) = g(\mathbf{z}, \boldsymbol{\theta})$.
- Latent representation is given by $\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta})$.

https://arxiv.org/pdf/1605.08803.pdf

Flows

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

Definition

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

- Normalizing - convert data distribution to *noise*.
- Flow - sequence of such mapping is also a flow

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

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

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

Flows

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

What we want

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

Planar Flows, 2015

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

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

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

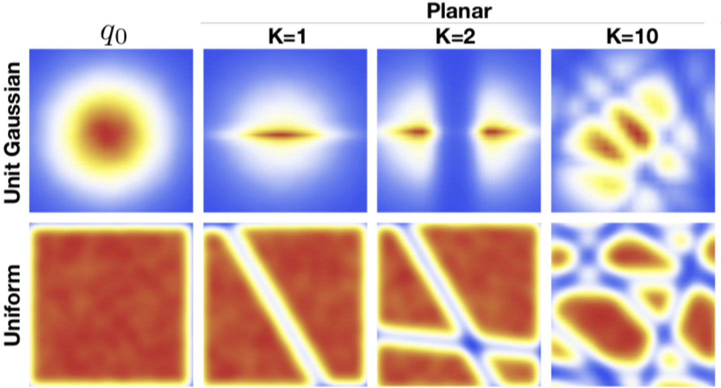
The transformation is invertible if (just one of example)

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

https://arxiv.org/pdf/1505.05770.pdf

Planar Flows, 2015

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



https://arxiv.org/pdf/1505.05770.pdf

Jacobian structure

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

NICE, 2014

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;
- $\tau : \mathbb{R}^{m-d} \times c(\mathbb{R}^d) \to \mathbb{R}^{m-d}$ – coupling law.
- $$\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) = \det\begin{pmatrix} \mathbf{I}_d & 0_{d \times m-d} \\ \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{1:d}} & \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}} \end{pmatrix} = \det\left(\frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}}\right)$$

---

https://arxiv.org/pdf/1410.8516.pdf

NICE, 2014

Coupling layer

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

Coupling function $c(\cdot)$

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

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

- $\tau(x, c) = x + c$ – *additive*;
- $\tau(x, c) = x \odot c, \; c \neq 0$ – multiplicative;
- $\tau(x, c) = x \odot c_1 + c_2, \; c_1 \neq 0$ – affine.

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

https://arxiv.org/pdf/1410.8516.pdf

NICE, 2014

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

What is the Jacobian for the additive coupling law
$\tau(x+c) = x + c$?
In this case the transformation is *volume preserving*.
The last layer is rescaling:

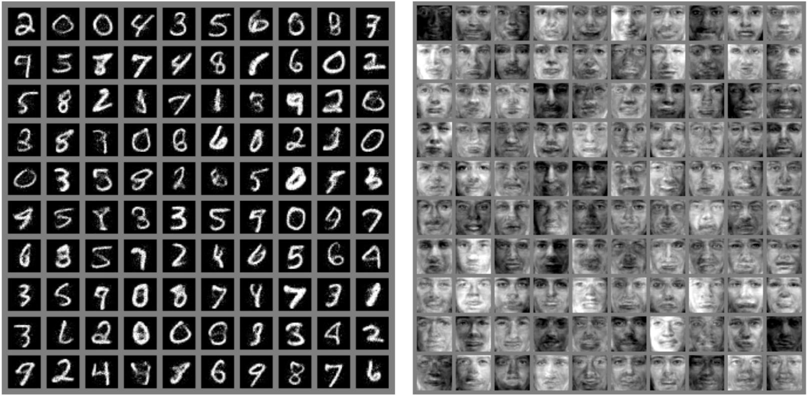
$$z_i = s_i x_i; \quad x_i = z_i / s_i.$$

What is the Jacobian of the last layer?

---

https://arxiv.org/pdf/1410.8516.pdf

NICE, 2014



(a) Model trained on MNIST (b) Model trained on TFD

---

https://arxiv.org/pdf/1410.8516.pdf

RealNVP, 2016

Affine coupling law

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

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

Jacobian

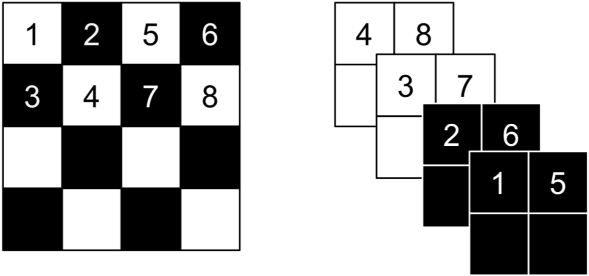
$$\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) = \det\begin{pmatrix} \mathbf{I}_d & 0_{d\times m-d} \\ \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{1:d}} & \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}} \end{pmatrix} = \prod_{i=1}^{m-d} \exp(c_1(\mathbf{x}_{1:d},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

Non-Volume Preserving.

---

https://arxiv.org/pdf/1605.08803.pdf

RealNVP, 2016

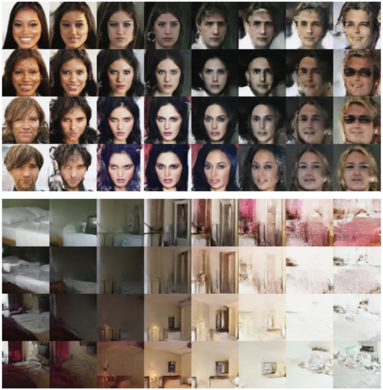


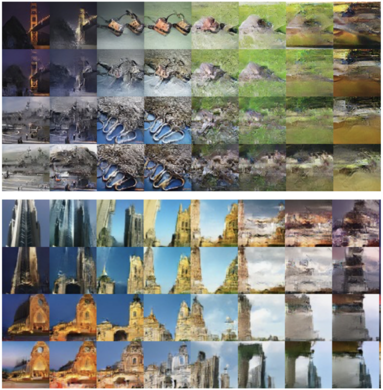
Masked convolutions are used to define ordering.

---

https://arxiv.org/pdf/1605.08803.pdf

RealNVP, 2016





https://arxiv.org/pdf/1605.08803.pdf

References

- Bishop, C. Pattern recognition and machine learning. 2006.
  Chapter 10.
- **NICE**: Non-linear Independent Components Estimation
  https://arxiv.org/abs/1410.8516
  Summary: Uses flows to model complex high-dimensional densities. Introduce the ways to compute determinant of Jacobian in a simple way. Triangular Jacobian, coupling layers, factorized distribution.
- Variational Inference with Normalizing Flows
  https://arxiv.org/abs/1505.05770
  Summary: Propose to use normalizing flows in variational inference. Discuss finite and infinitesimal flows. Uses invertible flows: planar, radial. Comparison with NICE.
- **RealNVP**: Density estimation using Real NVP
  https://arxiv.org/pdf/1605.08803.pdf
  Summary: Authors of NICE. The same idea and architecture, more practical. Lots of experiments and images. Coupling layers with checkerboard and channel-wise permutations.