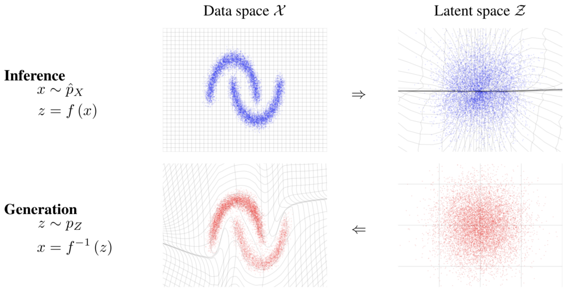
Deep Generative Models
Lecture 6

Roman Isachenko

Ozon Masters

Spring, 2021

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

Recap of previous lecture

Planar flow

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

Sylvester flow

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

NICE/RealNVP: Affine coupling law

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

Glow: invertible 1x1 conv

$$\mathbf{W} = \mathbf{P}\mathbf{L}(\mathbf{U} + \mathrm{diag}(\mathbf{s})).$$

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015
Berg R. et al. Sylvester normalizing flows for variational inference, 2018
Dinh L., Krueger D., Bengio Y. NICE: Non-linear Independent Components Estimation, 2014
Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016
Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Recap of previous lecture

ELBO

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

- Normal variational distribution $q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}^2_\phi(\mathbf{x}))$ is poor (e.g. has only one mode).
- Flows models convert a simple base distribution to a compex one using an invertible transformation with simple Jacobian.

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's 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.

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

Recap of previous lecture

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

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

- 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 an inverse flow function if we use flows in variational inference.

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

Gaussian autoregressive model

Consider an autoregressive model

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

with conditionals

$$p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}) = \mathcal{N}\left(\mu_i(\mathbf{x}_{1:i-1}), \sigma_i^2(\mathbf{x}_{1:i-1})\right).$$

Sampling: reparametrization trick

$$x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}), \quad z_i \sim \mathcal{N}(0, 1).$$

Sampling from the autoregressive model is **sequential**.
Note that we could interpret this sampling as a transformation $\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta})$, where $\mathbf{z}$ comes from base distribution $\mathcal{N}(0, 1)$.

Gaussian autoregressive model

Sampling: reparametrization trick

$$x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}), \quad z_i \sim \mathcal{N}(0, 1).$$

Inverse transform

$$z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}.$$

Jacobian

Autoregressive model has triangular Jacobian

$$\log \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right| = - \log \left| \det \left(\frac{\partial g(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}} \right) \right| = - \sum_{i=1}^{m} \log \sigma_i(\mathbf{x}_{1:i-1}).$$

We get an autoregressive model with tractable (triangular) Jacobian, which is easily invertible. It is a flow!

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

Gaussian autoregressive flow

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}).$$
$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}.$$

Generation function $g(\mathbf{z}, \boldsymbol{\theta})$ is **sequential**. Inference function $f(\mathbf{x}, \boldsymbol{\theta})$ is **not sequential**.

Forward KL for flow model

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

- We need to be able to compute $f(\mathbf{x}, \boldsymbol{\theta})$ and its Jacobian.
- We need to be able to compute the density $p(\mathbf{z})$.
- We don't need to think about computing the function $g(\mathbf{z}, \boldsymbol{\theta}) = f^{-1}(\mathbf{z}, \boldsymbol{\theta})$ until we want to sample from the flow.

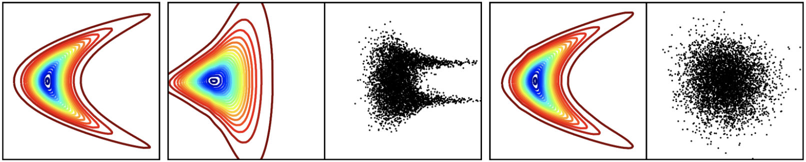
Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

Masked autoregressive flow (MAF)

Gaussian autoregressive model

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{m} p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}) = \prod_{i=1}^{m} \mathcal{N}\left(x_i|\mu_i(\mathbf{x}_{1:i-1}), \sigma_i^2(\mathbf{x}_{1:i-1})\right).$$

We could use MADE (masked autoencoder) as a conditional model. The sampling order might be crucial.



(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

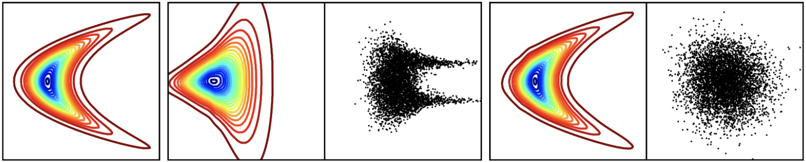
Samples from the base distribution could be an indicator of how good the flow was fitted.

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

Masked autoregressive flow (MAF)

Gaussian autoregressive model

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{m} p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}) = \prod_{i=1}^{m} \mathcal{N}\left(x_i|\mu_i(\mathbf{x}_{1:i-1}), \sigma_i^2(\mathbf{x}_{1:i-1})\right).$$



(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

MAF is just a stacked MADE model with different ordering.

- Parallel density estimation.
- Sequential sampling.

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

Inverse autoregressive flow (IAF)

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

Gaussian autoregressive flow

$$x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}) = (z_i - \tilde{\mu}_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\tilde{\sigma}_i(\mathbf{x}_{1:i-1})}$$

$$z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})} = \tilde{\sigma}_i(\mathbf{x}_{1:i-1}) \cdot x_i + \tilde{\mu}_i(\mathbf{x}_{1:i-1}).$$

Let just swap $\mathbf{z}$ and $\mathbf{x}$.

Inverse autoregressive flow

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_i = \tilde{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot z_i + \tilde{\mu}_i(\mathbf{z}_{1:i-1})$$

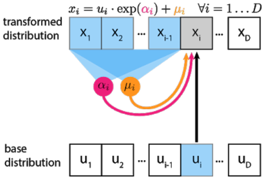
$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_i = (x_i - \tilde{\mu}_i(\mathbf{z}_{1:i-1})) \cdot \frac{1}{\tilde{\sigma}_i(\mathbf{z}_{1:i-1})}.$$

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

Inverse autoregressive flow (IAF)

Gaussian autoregressive flow: $f(\mathbf{x}, \boldsymbol{\theta})$

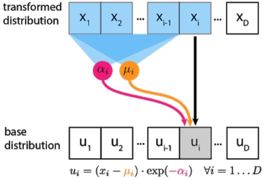
$$x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}).$$



Inverse transform: $g(\mathbf{z}, \boldsymbol{\theta})$

$$z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})};$$

$$z_i = \tilde{\sigma}_i(\mathbf{x}_{1:i-1}) \cdot x_i + \tilde{\mu}_i(\mathbf{x}_{1:i-1}).$$



Inverse autoregressive flow: $f(\mathbf{x}, \boldsymbol{\theta})$

$$x_i = \tilde{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot z_i + \tilde{\mu}_i(\mathbf{z}_{1:i-1}).$$

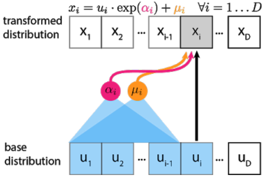


image credit: https://blog.evjang.com/2018/01/nf2.html

Autoregressive flows

Forward and inverse transform in MAF

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}.$$

- Sampling is sequential.
- Density estimation is parallel.

Forward and inverse transform in IAF

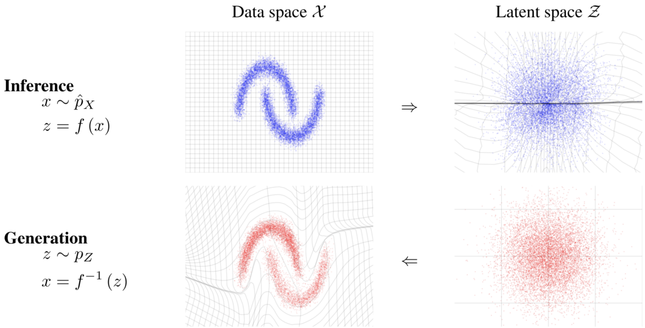
$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_i = \tilde{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot z_i + \tilde{\mu}_i(\mathbf{z}_{1:i-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_i = (x_i - \tilde{\mu}_i(\mathbf{z}_{1:i-1})) \cdot \frac{1}{\tilde{\sigma}_i(\mathbf{z}_{1:i-1})}.$$

- Sampling is parallel.
- Density estimation is sequential.

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

Flows



- MAF performs parallel inference that is useful for density estimation tasks (forward KL or MLE).
- IAF performs parallel generation that is useful for variational inference (reverse KL).

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

Inverse autoregressive flow (IAF)

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_i = \tilde{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot z_i + \tilde{\mu}_i(\mathbf{z}_{1:i-1}).$$
$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_i = (x_i - \tilde{\mu}_i(\mathbf{z}_{1:i-1})) \cdot \frac{1}{\tilde{\sigma}_i(\mathbf{z}_{1:i-1})}.$$

Reverse KL for flow model

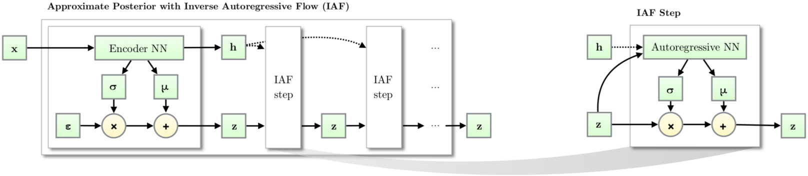
$$KL(p||\pi) = \mathbb{E}_{p(\mathbf{z})} \left[\log p(\mathbf{z}) - \log \left| \det \left(\frac{\partial g(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}} \right) \right| - \log \pi(g(\mathbf{z}, \boldsymbol{\theta})) \right]$$

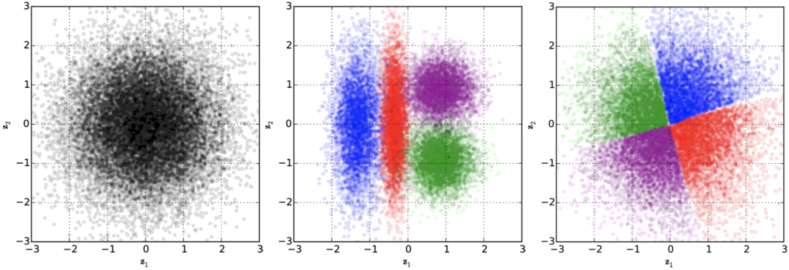
- We don't need to think about computing the function $f(\mathbf{x}, \boldsymbol{\theta})$.
- Inverse autoregressive flow is a natural choice for using flows in VAE:

$$\mathbf{z}_0 = \boldsymbol{\sigma}(\mathbf{x}) \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}(\mathbf{x}), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, 1); \quad \sim q(\mathbf{z}_0|\mathbf{x}, \phi).$$
$$\mathbf{z}_k = \boldsymbol{\sigma}_k(\mathbf{z}_{k-1}) \odot \mathbf{z}_{k-1} + \boldsymbol{\mu}_k(\mathbf{z}_{k-1}), \quad k \geq 1; \quad \sim q_k(\mathbf{z}_k|\mathbf{x}, \phi, \{\phi_j\}_{j=1}^k).$$

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

Inverse autoregressive flow (IAF)





(a) Prior distribution (b) Posteriors in standard VAE (c) Posteriors in VAE with IAF

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

MAF vs IAF vs RealNVP

MADE/MAF

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

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

IAF

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

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

NICE/RealNVP/Glow

$$\mathbf{x}_1 = \mathbf{z}_1;$$
$$\mathbf{x}_2 = \mathbf{z}_2 \odot \exp\left(c_1(\mathbf{z}_1, \boldsymbol{\theta})\right) + c_2(\mathbf{z}_1, \boldsymbol{\theta}).$$

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

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

MAF vs IAF vs RealNVP

RealNVP

$$
\begin{aligned}
\mathbf{x}_1 &= \mathbf{z}_1; \\
\mathbf{x}_2 &= \mathbf{z}_2 \odot \exp\left(c_1(\mathbf{z}_1, \boldsymbol{\theta})\right) + c_2(\mathbf{z}_1, \boldsymbol{\theta}).
\end{aligned}
$$

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

RealNVP is a special case of MAF and IAF:

MAF

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

IAF

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

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

MAF/IAF pros and cons

MAF

- Sampling is slow.
- Likelihood evaluation is fast.

IAF

- Sampling is fast.
- Likelihood evaluation is slow.

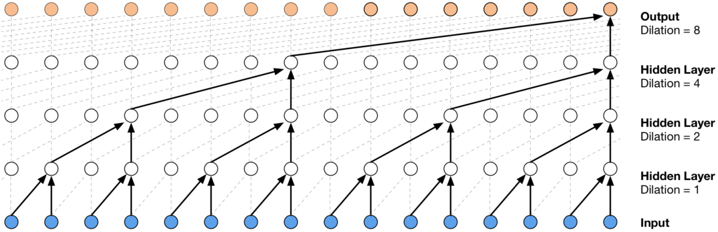
How to take the best of both worlds?

WaveNet (2016)

Autoregressive model for raw audio waveforms generation

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

The model uses causal dilated convolutions.



Oord A. et al. Wavenet: A generative model for raw audio, 2016

Parallel WaveNet, 2017

Previous WaveNet model

- raw audio is high-dimensional (e.g. 16000 samples per second for 16kHz audio);
- WaveNet encodes 8-bit signal with 256-way categorical distribution.

Goal

- improved fidelity (24kHz instead of 16kHz) $\rightarrow$ increase dilated convolution filter size from 2 to 3;
- 16-bit signals $\rightarrow$ mixture of logistics instead of categorical distribution.

Oord A. et al. Parallel WaveNet: Fast High-Fidelity Speech Synthesis, 2017

Parallel WaveNet, 2017

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.

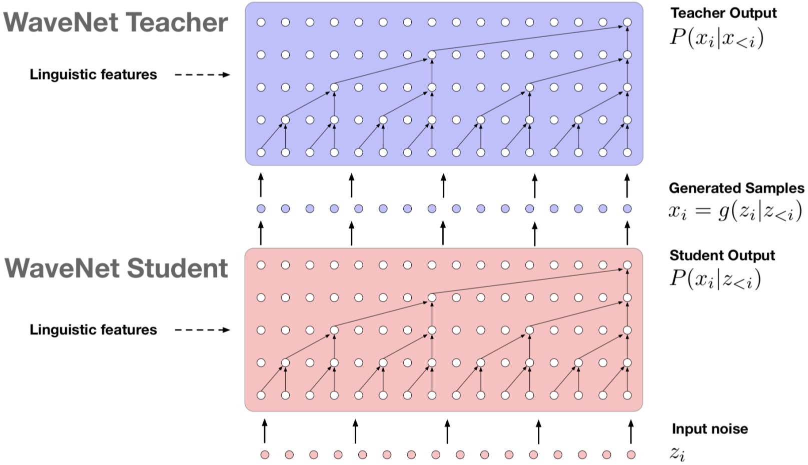
Student objective

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

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

Oord A. et al. Parallel WaveNet: Fast High-Fidelity Speech Synthesis, 2017

Parallel WaveNet, 2017



Oord A. et al. Parallel WaveNet: Fast High-Fidelity Speech Synthesis, 2017

Flow KL duality

Theorem

Fitting flow model $p(\mathbf{x}, \boldsymbol{\theta})$ to the target distribution $\pi(\mathbf{x})$ using forward KL (MLE) is equivalent to fitting the induced distribution $p(\mathbf{z}, \boldsymbol{\theta})$ to the base $p(\mathbf{z})$ using reverse KL:

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

- $p(\mathbf{z})$ is a base distribution; $\pi(\mathbf{x})$ is a data distribution;
- $\mathbf{z} \sim p(\mathbf{z})$, $\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta})$, $\mathbf{x} \sim p(\mathbf{x}, \boldsymbol{\theta})$;
- $\mathbf{x} \sim \pi(\mathbf{x})$, $\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta})$, $\mathbf{z} \sim p(\mathbf{z}|\boldsymbol{\theta})$;

$$\log p(\mathbf{z}|\boldsymbol{\theta}) = \log \pi(g(\mathbf{z}, \boldsymbol{\theta})) + \log \left| \det \left(\frac{\partial g(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}} \right) \right|;$$

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

Papamakarios G. et al. Normalizing flows for probabilistic modeling and inference, 2019

MAF vs IAF

Theorem

Fitting flow model $p(\mathbf{x}, \boldsymbol{\theta})$ to the target distribution $\pi(\mathbf{x})$ using forward KL (MLE) is equivalent to fitting the induced distribution $p(\mathbf{z}, \boldsymbol{\theta})$ to the base $p(\mathbf{z})$ using reverse KL:

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

Proof

$$KL\left(p(\mathbf{z}|\boldsymbol{\theta})||\pi(\mathbf{z})\right) = \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}\bigl[\log p(\mathbf{z}|\boldsymbol{\theta}) - \log p(\mathbf{z})\bigr] =$$
$$= \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\theta})}\left[\log \pi(g(\mathbf{z}, \boldsymbol{\theta})) + \log\left|\det\left(\frac{\partial g(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}}\right)\right| - \log p(\mathbf{z})\right] =$$
$$= \mathbb{E}_{\pi(\mathbf{x})}\left[\log \pi(\mathbf{x}) - \log\left|\det\left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}}\right)\right| - \log p(f(\mathbf{x}, \boldsymbol{\theta}))\right].$$

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

MAF vs IAF

Theorem

Fitting flow model $p(\mathbf{x}, \boldsymbol{\theta})$ to the target distribution $\pi(\mathbf{x})$ using forward KL (MLE) is equivalent to fitting the induced distribution $p(\mathbf{z}, \boldsymbol{\theta})$ to the base $p(\mathbf{z})$ using reverse KL:

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

Proof (continued)

$$KL\left(p(\mathbf{z}|\boldsymbol{\theta}) || p(\mathbf{z})\right) =$$

$$= \mathbb{E}_{\pi(\mathbf{x})}\left[\log \pi(\mathbf{x}) - \log \left| \det \left(\frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right| - \log p(f(\mathbf{x}, \boldsymbol{\theta}))\right] =$$

$$= \mathbb{E}_{\pi(\mathbf{x})}\left[\log \pi(\mathbf{x}) - \log p(\mathbf{x}|\boldsymbol{\theta})\right] = KL(\pi(\mathbf{x}) || p(\mathbf{x}|\boldsymbol{\theta})).$$

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

Dequantization

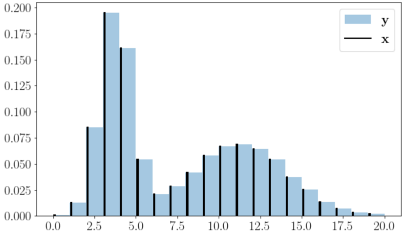
- Images are discrete data, pixels lie in the [0, 255] integer domain (the model is $P(\mathbf{x}|\boldsymbol{\theta}) = \text{Categorical}(\boldsymbol{\pi}(\boldsymbol{\theta}))$).
- Flow is a continuous model (it works with continuous data $\mathbf{x}$).

By fitting a continuous density model to discrete data, one can produce a degenerate solution with all probability mass on discrete values.
How to convert a discrete data distribution to a continuous one?

Uniform dequantization

$$\mathbf{x} \sim \text{Categorical}(\boldsymbol{\pi})$$
$$\mathbf{u} \sim U[0, 1]$$

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



Theis L., Oord A., Bethge M. A note on the evaluation of generative models. 2015

Uniform dequantization

Statement

Fitting continious model $p(\mathbf{y}|\boldsymbol{\theta})$ 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 log-likelihood for a discrete model:

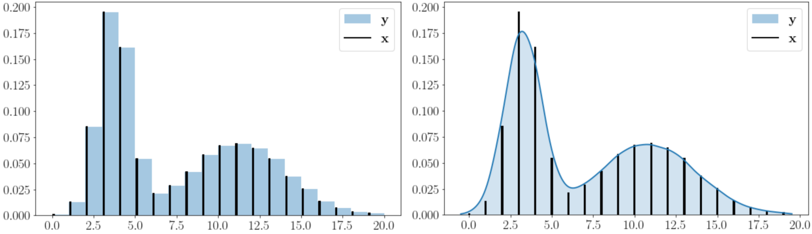
$$P(\mathbf{x}|\boldsymbol{\theta}) = \int_{U[0,1]} p(\mathbf{x} + \mathbf{u}|\boldsymbol{\theta}) d\mathbf{u}$$

Thus, the maximisation of continuous model log-likelihood on $\mathbf{y}$ can't lead to the a collapse onto the discrete data (the objective is bounded above by the discrete model log-likelihood).

Proof

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

Variational dequantization



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

How to perform 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

$$\log P(\mathbf{x}|\boldsymbol{\theta}) = \left[\log \int q(\mathbf{u}|\mathbf{x}) \frac{p(\mathbf{x}+\mathbf{u}|\boldsymbol{\theta})}{q(\mathbf{u}|\mathbf{x})} d\mathbf{u}\right] \geq$$

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

Uniform dequantization bound

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

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

Ho J. et al. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design, 2019

Flow++

Variational lower bound

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

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

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

Then

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

Ho J. et al. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design, 2019

Flow++

Variational lower

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

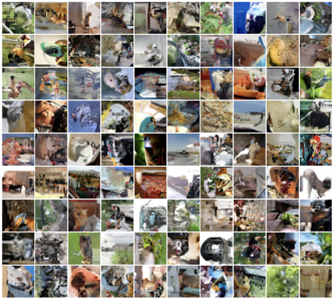
- If $p(\mathbf{x} + \mathbf{u}|\boldsymbol{\theta})$ is also a flow model, it is straightforward to calculate stochastic gradient of this ELBO.
- Uniform dequantization is a special case of variational dequantization ($q(\mathbf{u}|\mathbf{x}) = U[0,1]$). The gap between $\log P(\mathbf{x}|\boldsymbol{\theta})$ 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.

Ho J. et al. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design, 2019

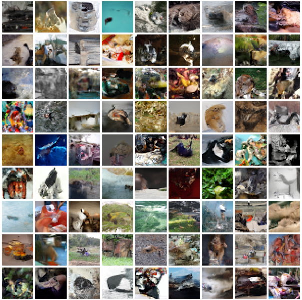
Flow++

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++

Ho J. et al. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design, 2019

Summary

- Gaussian autoregressive model is a special type of flow.
- MAF is an example of such a model which is suitable for density estimation tasks. IAF uses an inverse autoregressive transformation for variational inference task.
- RealNVP is a special case of IAF and MAF.
- There is a duality between forward and reverse KL for flow models.
- To apply a continuous model to a discrete distribution it is standard practice to dequantize data at first.
- Uniform dequantization is the simplest form of dequantization. Variational dequantization is a more natural type that was proposed in Flow++ model.