# Deep Generative Models

## Lecture 5

Roman Isachenko

Moscow Institute of Physics and Technology

2023, Autumn

# Recap of previous lecture

## Jacobian matrix

Let $f : \mathbb{R}^m \to \mathbb{R}^m$ be a differentiable function.

$$\mathbf{z} = f(\mathbf{x}), \quad \mathbf{J} = \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \dots & \frac{\partial z_1}{\partial x_m} \\ \dots & \dots & \dots \\ \frac{\partial z_m}{\partial x_1} & \dots & \frac{\partial z_m}{\partial x_m} \end{pmatrix} \in \mathbb{R}^{m \times m}$$

## Change of variable theorem (CoV)

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

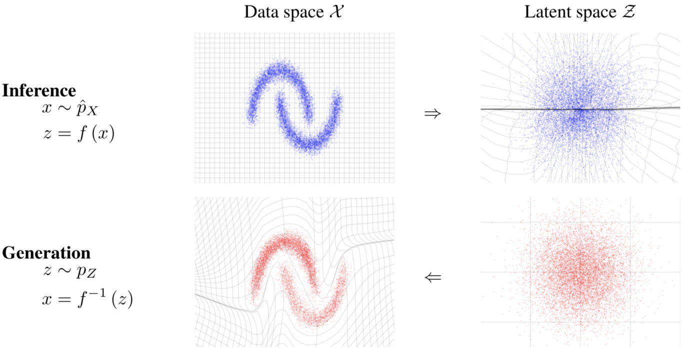
$$p(\mathbf{x}) = p(\mathbf{z})|\det(\mathbf{J}_f)| = p(\mathbf{z}) \left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)\right| = p(f(\mathbf{x})) \left|\det\left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right)\right|$$

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

# Recap of previous lecture

## Definition

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




## Log likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f_K \circ \cdots \circ f_1(\mathbf{x})) + \sum_{k=1}^{K} \log |\det(\mathbf{J}_{f_k})|$$

---

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

# Recap of previous lecture

## Forward KL for flow model

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)|$$

## Reverse KL for flow model

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

## Flow KL duality

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

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

---

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

# Recap of previous lecture

## Flow log-likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)|$$

The main challenge is a determinant of the Jacobian.

## Linear flows

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}, \quad \boldsymbol{\theta} = \mathbf{W}, \quad \mathbf{J}_f = \mathbf{W}^T$$

- LU-decomposition
$$\mathbf{W} = \mathbf{P}\mathbf{L}\mathbf{U}.$$
- QR-decomposition
$$\mathbf{W} = \mathbf{Q}\mathbf{R}.$$

Decomposition should be done only once in the beggining. Next, we fit decomposed matrices ($\mathbf{P}/\mathbf{L}/\mathbf{U}$ or $\mathbf{Q}/\mathbf{R}$).

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

*Hoogeboom E., et al. Emerging convolutions for generative normalizing flows, 2019*

# Recap of previous lecture

Consider an autoregressive model

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

## Gaussian autoregressive NF

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

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

- We have an **invertible** and **differentiable** transformation from $p(\mathbf{z})$ to $p(\mathbf{x}|\boldsymbol{\theta})$.
- Jacobian of such transformation is triangular!

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

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

# Gaussian autoregressive NF

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

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

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

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

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

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

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

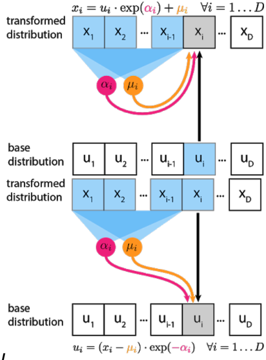



*image credit: https://blog.evjang.com/2018/01/nf2.html*

# Outline

1. RealNVP: coupling layer
2. Normalizing flows as VAE model
3. Discrete data vs continuous model
   - Discretization of continuous distribution
   - Dequantization of discrete data

# Outline

1. RealNVP: coupling layer
2. Normalizing flows as VAE model
3. Discrete data vs continuous model
   - Discretization of continuous distribution
   - Dequantization of discrete data

# RealNVP

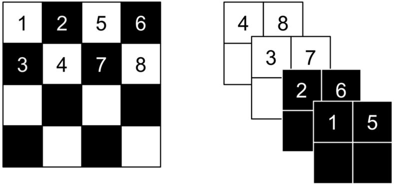
Let split $\mathbf{x}$ and $\mathbf{z}$ in two parts:

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2] = [\mathbf{x}_{1:d}, \mathbf{x}_{d+1:m}]; \quad \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2] = [\mathbf{z}_{1:d}, \mathbf{z}_{d+1:m}].$$

## Coupling layer

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1; \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \boldsymbol{\sigma}(\mathbf{z}_1, \boldsymbol{\theta}) + \boldsymbol{\mu}(\mathbf{z}_1, \boldsymbol{\theta}). \end{cases} \qquad \begin{cases} \mathbf{z}_1 = \mathbf{x}_1; \\ \mathbf{z}_2 = (\mathbf{x}_2 - \boldsymbol{\mu}(\mathbf{x}_1, \boldsymbol{\theta})) \odot \frac{1}{\boldsymbol{\sigma}(\mathbf{x}_1, \boldsymbol{\theta})}. \end{cases}$$

## Image partitioning




- Checkerboard ordering uses masking.
- Channelwise ordering uses splitting.

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

# RealNVP

## Coupling layer

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1; \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \boldsymbol{\sigma}(\mathbf{z}_1, \boldsymbol{\theta}) + \boldsymbol{\mu}(\mathbf{z}_1, \boldsymbol{\theta}). \end{cases} \qquad \begin{cases} \mathbf{z}_1 = \mathbf{x}_1; \\ \mathbf{z}_2 = (\mathbf{x}_2 - \boldsymbol{\mu}(\mathbf{x}_1, \boldsymbol{\theta})) \odot \frac{1}{\boldsymbol{\sigma}(\mathbf{x}_1, \boldsymbol{\theta})}. \end{cases}$$

Estimating the density takes 1 pass, sampling takes 1 pass!

## Jacobian

$$\det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) = \det \begin{pmatrix} \mathbf{I}_d & 0_{d \times m-d} \\ \frac{\partial \mathbf{z}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{z}_2}{\partial \mathbf{x}_2} \end{pmatrix} = \prod_{j=1}^{m-d} \frac{1}{\sigma_j(\mathbf{x}_1, \boldsymbol{\theta})}.$$

## Gaussian AR NF

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

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

How to get RealNVP coupling layer from gaussian AR NF?

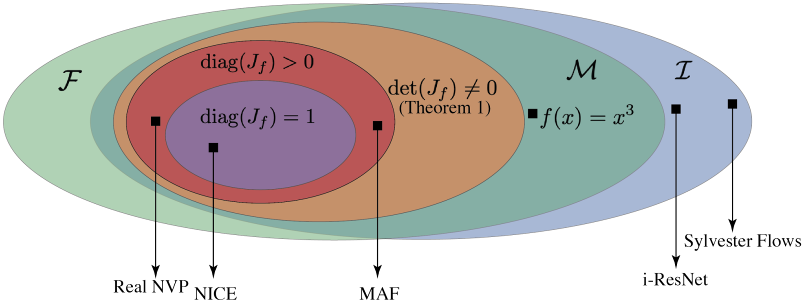
*Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016*

# Glow samples

Glow model: coupling layer + linear flows (1x1 convs)

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

# Venn diagram for Normalizing flows




- $\mathcal{I}$ – invertible functions.
- $\mathcal{F}$ – continuously differentiable functions whose Jacobian is lower triangular.
- $\mathcal{M}$ – invertible functions from $\mathcal{F}$.

---

*Song Y., Meng C., Ermon S. Mintnet: Building invertible neural networks with masked convolutions, 2019*

# Outline

# VAE vs Normalizing flows

| | **VAE** | **NF** |
|---|---|---|
| **Objective** | ELBO $\mathcal{L}$ | Forward KL/MLE |
| **Encoder** | stochastic<br>$\mathbf{z} \sim q(\mathbf{z}\|\mathbf{x}, \boldsymbol{\phi})$ | deterministic<br>$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta})$<br>$q(\mathbf{z}\|\mathbf{x}, \boldsymbol{\theta}) = \delta(\mathbf{z} - f(\mathbf{x}, \boldsymbol{\theta}))$ |
| **Decoder** | stochastic<br>$\mathbf{x} \sim p(\mathbf{x}\|\mathbf{z}, \boldsymbol{\theta})$ | deterministic<br>$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta})$<br>$p(\mathbf{x}\|\mathbf{z}, \boldsymbol{\theta}) = \delta(\mathbf{x} - g(\mathbf{z}, \boldsymbol{\theta}))$ |
| **Parameters** | $\boldsymbol{\phi}, \boldsymbol{\theta}$ | $\boldsymbol{\theta} \equiv \boldsymbol{\phi}$ |

### Theorem

MLE for normalizing flow is equivalent to maximization of ELBO for VAE model with deterministic encoder and decoder:

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \delta(\mathbf{x} - f^{-1}(\mathbf{z}, \boldsymbol{\theta})) = \delta(\mathbf{x} - g(\mathbf{z}, \boldsymbol{\theta}));$$

$$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \delta(\mathbf{z} - f(\mathbf{x}, \boldsymbol{\theta})).$$

---

*Nielsen D., et al. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows, 2020*

# Normalizing flow as VAE

## Proof

1. Dirac delta function property

$$\mathbb{E}_{\delta(\mathbf{x}-\mathbf{y})} f(\mathbf{x}) = \int \delta(\mathbf{x}-\mathbf{y}) f(\mathbf{x}) d\mathbf{x} = f(\mathbf{y}).$$

2. CoV theorem and Bayes theorem:

$$p(\mathbf{x}|\boldsymbol{\theta}) = p(\mathbf{z})|\det(\mathbf{J}_f)|;$$

$$p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \frac{p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z})}{p(\mathbf{x}|\boldsymbol{\theta})}; \quad \Rightarrow \quad p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})|\det(\mathbf{J}_f)|.$$

3. Log-likelihood decomposition

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

*Nielsen D., et al. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows, 2020*

# Normalizing flow as VAE

## Proof

ELBO objective:

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

$$= \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} \left[ \log \frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} + \log p(\mathbf{z}) \right].$$

1. Dirac delta function property:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} \log p(\mathbf{z}) = \int \delta(\mathbf{z} - f(\mathbf{x},\boldsymbol{\theta})) \log p(\mathbf{z}) d\mathbf{z} = \log p(f(\mathbf{x},\boldsymbol{\theta})).$$

2. CoV theorem and Bayes theorem:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} \log \frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} \log \frac{p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})|\det(\mathbf{J}_f)|}{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} = \log|\det \mathbf{J}_f|.$$

3. Log-likelihood decomposition

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) = \log p(f(\mathbf{x},\boldsymbol{\theta})) + \log|\det \mathbf{J}_f|.$$

---

*Nielsen D., et al. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows, 2020*

# Outline

1. RealNVP: coupling layer
2. Normalizing flows as VAE model
3. Discrete data vs continuous model
   - Discretization of continuous distribution
   - Dequantization of discrete data

# Discrete data vs continuous model

Let our data $\mathbf{y}$ comes from discrete distribution $\Pi(\mathbf{y})$ and we have continuous model $p(\mathbf{x}|\boldsymbol{\theta}) = \text{NN}(\mathbf{x}, \boldsymbol{\theta})$.

- Images (and not only images) are discrete data, pixels lie in the integer domain ($\{0, 255\}$).
- By fitting a continuous density model $p(\mathbf{x}|\boldsymbol{\theta})$ to discrete data $\Pi(\mathbf{y})$, one can produce a degenerate solution with all probability mass on discrete values.

## Discrete model

- Use **discrete** model (e.x. $P(\mathbf{y}|\boldsymbol{\theta}) = \text{Cat}(\boldsymbol{\pi}(\boldsymbol{\theta}))$).
- Minimize any suitable divergence measure $D(\Pi, P)$.
- NF works only with continuous data $\mathbf{x}$ (there are discrete NF, see papers below).
- If pixel value is not presented in the train data, it won't be predicted.

---

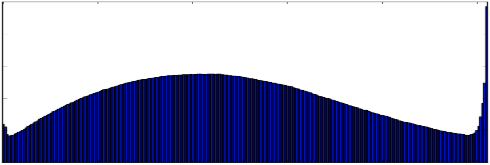
*Hoogeboom E. et al. Integer discrete flows and lossless compression, 2019*
*Tran D. et al. Discrete flows: Invertible generative models of discrete data, 2019*

# Discrete data vs continuous model

## Continuous model

- Use **continuous** model (e.x. $p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\mathbf{x}))$), but
  - **discretize** model (make the model outputs discrete): transform $p(\mathbf{x}|\boldsymbol{\theta})$ to $P(\mathbf{y}|\boldsymbol{\theta})$;
  - **dequantize** data (make the data continuous): transform $\Pi(\mathbf{y})$ to $\pi(\mathbf{x})$.
- Continuous distribution knows numerical relationships.

## CIFAR-10 pixel values distribution



*Salimans T. et al. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications, 2017*

# Outline

# Discretization of continuous distribution

## Model discretization through CDF

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

## Mixture of logistic distributions

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

## PixelCNN++

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

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

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

---

*Salimans T. et al. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications, 2017*

# Outline

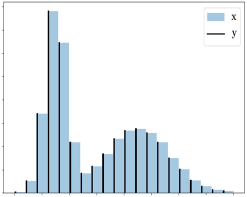
# Uniform dequantization

Let dequantize discrete distribution $\Pi(\mathbf{y})$ to continuous distribution $\pi(\mathbf{x})$ in the following way: $\mathbf{x} = \mathbf{y} + \mathbf{u}$, where $\mathbf{u} \sim U[0, 1]$.

## Theorem

Fitting continuous model $p(\mathbf{x}|\boldsymbol{\theta})$ on uniformly dequantized data is equivalent to maximization of a lower bound on log-likelihood for a discrete model:

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




## Proof

$$\mathbb{E}_\pi \log p(\mathbf{x}|\boldsymbol{\theta}) = \int \pi(\mathbf{x}) \log p(\mathbf{x}|\boldsymbol{\theta}) d\mathbf{x} = \sum \Pi(\mathbf{y}) \int_{U[0,1]} \log p(\mathbf{y}+\mathbf{u}|\boldsymbol{\theta}) d\mathbf{u} \leq$$

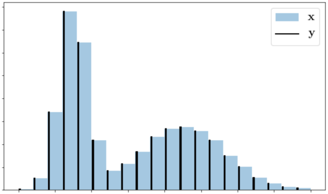
$$\leq \sum \Pi(\mathbf{y}) \log \int_{U[0,1]} p(\mathbf{y} + \mathbf{u}|\boldsymbol{\theta}) d\mathbf{u} =$$

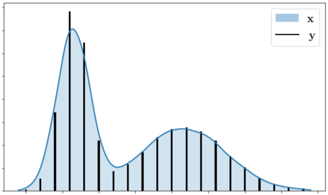
$$= \sum \Pi(\mathbf{y}) \log P(\mathbf{y}|\boldsymbol{\theta}) = \mathbb{E}_\Pi \log P(\mathbf{y}|\boldsymbol{\theta}).$$

---

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

# Variational dequantization




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

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

# Variational dequantization

## Variational lower bound

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

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

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

## Flow++: flow-based variational dequantization

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

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

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

---

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

# Summary

- Gaussian autoregressive flow is an autoregressive model with triangular Jacobian. It has fast inference function and slow generation function. Forward KL is a natural loss function.
- The RealNVP coupling layer is an effective type of flow (special case of AR flows) that has fast inference and generation modes.
- NF models could be treated as VAE model with deterministic encoder and decoder.
- Lots of data are discrete. We able to discretize the model or to dequantize our data to use continuous model.
- Uniform dequantization is the simplest form of dequantization. Variational dequantization is a more natural type that uses variational inference.