

# Deep Generative Models

## Lecture 4

Roman Isachenko



AI Masters

2024, Spring

## Recap of previous lecture

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

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1; \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \sigma_{\theta}(\mathbf{z}_1) + \mu_{\theta}(\mathbf{z}_1). \end{cases} \quad \begin{cases} \mathbf{z}_1 = \mathbf{x}_1; \\ \mathbf{z}_2 = (\mathbf{x}_2 - \mu_{\theta}(\mathbf{x}_1)) \odot \frac{1}{\sigma_{\theta}(\mathbf{x}_1)}. \end{cases}$$

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

### Jacobian

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

Coupling layer is a special case of autoregressive NF.

# Recap of previous lecture

## 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 variables (latent variables/parameters);
- ▶  $p(\mathbf{x}|\mathbf{t})$  – likelihood;
- ▶  $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{t})p(\mathbf{t})d\mathbf{t}$  – evidence;
- ▶  $p(\mathbf{t})$  – prior distribution,  $p(\mathbf{t}|\mathbf{x})$  – posterior distribution.

## Posterior distribution

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{p(\mathbf{X}|\theta)p(\theta)}{\int p(\mathbf{X}|\theta)p(\theta)d\theta}$$

# Recap of previous lecture

## Latent variable models (LVM)

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

## MLE problem for LVM

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg \max_{\boldsymbol{\theta}} \log p(\mathbf{X}|\boldsymbol{\theta}) = \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^n \log p(\mathbf{x}_i|\boldsymbol{\theta}) = \\ &= \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^n \log \int p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\theta})p(\mathbf{z}_i)d\mathbf{z}_i.\end{aligned}$$

## Naive Monte-Carlo estimation

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})p(\mathbf{z})d\mathbf{z} = \mathbb{E}_{p(\mathbf{z})}p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) \approx \frac{1}{K} \sum_{k=1}^K p(\mathbf{x}|\mathbf{z}_k, \boldsymbol{\theta}),$$

where  $\mathbf{z}_k \sim p(\mathbf{z})$ .

# Recap of previous lecture

## ELBO derivation 1 (inequality)

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z} \geq \mathbb{E}_q \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})} = \mathcal{L}(q, \boldsymbol{\theta})$$

## ELBO derivation 2 (equality)

$$\begin{aligned} \mathcal{L}(q, \boldsymbol{\theta}) &= \int q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})} d\mathbf{z} = \int q(\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) p(\mathbf{x}|\boldsymbol{\theta})}{q(\mathbf{z})} d\mathbf{z} = \\ &= \log p(\mathbf{x}|\boldsymbol{\theta}) - KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})) \end{aligned}$$

## Variational decomposition

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

# Recap of previous lecture

## Variational lower Bound (ELBO)

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

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})} d\mathbf{z} = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - KL(q(\mathbf{z})||p(\mathbf{z}))$$

## Log-likelihood decomposition

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathbb{E}_q \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - KL(q(\mathbf{z})||p(\mathbf{z})) + KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})).$$

- ▶ Instead of maximizing incomplete likelihood, maximize ELBO

$$\max_{\boldsymbol{\theta}} p(\mathbf{x}|\boldsymbol{\theta}) \rightarrow \max_{q, \boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta})$$

- ▶ Maximization of ELBO by variational distribution  $q$  is equivalent to minimization of KL

$$\arg \max_q \mathcal{L}(q, \boldsymbol{\theta}) \equiv \arg \min_q KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})).$$

# Recap of previous lecture

## EM-algorithm

### ► E-step

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

### ► M-step

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

## Amortized variational inference

Restrict a family of all possible distributions  $q(\mathbf{z})$  to a parametric class  $q(\mathbf{z} | \mathbf{x}, \phi)$  conditioned on samples  $\mathbf{x}$  with parameters  $\phi$ .

## Variational Bayes

### ► E-step

$$\phi_k = \phi_{k-1} + \eta \nabla_{\phi} \mathcal{L}(\phi, \boldsymbol{\theta}_{k-1})|_{\phi=\phi_{k-1}}$$

### ► M-step

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\phi_k, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k-1}}$$

# Outline

1. ELBO gradients, reparametrization trick
2. Variational autoencoder (VAE)
3. Data dequantization



# Outline

1. ELBO gradients, reparametrization trick
2. Variational autoencoder (VAE)
3. Data dequantization

## ELBO gradients, (M-step, $\nabla_{\theta} \mathcal{L}(\phi, \theta)$ )

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

M-step:  $\nabla_{\theta} \mathcal{L}(\phi, \theta)$

$$\begin{aligned} \nabla_{\theta} \mathcal{L}(\phi, \theta) &= \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} \approx \\ &\approx \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}^*, \theta), \quad \mathbf{z}^* \sim q(\mathbf{z}|\mathbf{x}, \phi). \end{aligned}$$

Naive Monte-Carlo estimation

$$p(\mathbf{x}|\theta) = \int p(\mathbf{x}|\mathbf{z}, \theta) p(\mathbf{z}) d\mathbf{z} = \mathbb{E}_{p(\mathbf{z})} p(\mathbf{x}|\mathbf{z}, \theta) \approx \frac{1}{K} \sum_{k=1}^K p(\mathbf{x}|\mathbf{z}_k, \theta),$$

where  $\mathbf{z}_k \sim p(\mathbf{z})$ .

The variational posterior  $q(\mathbf{z}|\mathbf{x}, \phi)$  assigns typically more probability mass in a smaller region than the prior  $p(\mathbf{z})$ .

image credit: [https://jmtomczak.github.io/blog/4/4\\_VAE.html](https://jmtomczak.github.io/blog/4/4_VAE.html)

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

### E-step: $\nabla_{\phi} \mathcal{L}(\phi, \theta)$

Difference from M-step: density function  $q(\mathbf{z}|\mathbf{x}, \phi)$  depends on the parameters  $\phi$ , it is impossible to use the Monte-Carlo estimation:

$$\begin{aligned}\nabla_{\phi} \mathcal{L}(\phi, \theta) &= \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \left[ \log p(\mathbf{x}|\mathbf{z}, \theta) - \log \frac{q(\mathbf{z}|\mathbf{x}, \phi)}{p(\mathbf{z})} \right] d\mathbf{z} \\ &\neq \int q(\mathbf{z}|\mathbf{x}, \phi) \nabla_{\phi} \left[ \log p(\mathbf{x}|\mathbf{z}, \theta) - \log \frac{q(\mathbf{z}|\mathbf{x}, \phi)}{p(\mathbf{z})} \right] d\mathbf{z}\end{aligned}$$

### Reparametrization trick (LOTUS trick)

- ▶  $r(x) = \mathcal{N}(x|0, 1)$ ,  $y = \sigma \cdot x + \mu$ ,  $p_Y(y|\theta) = \mathcal{N}(y|\mu, \sigma^2)$ ,  $\theta = [\mu, \sigma]$ .
- ▶  $\epsilon^* \sim r(\epsilon)$ ,  $\mathbf{z} = g_{\phi}(\mathbf{x}, \epsilon)$ ,  $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \phi)$

$$\begin{aligned}\nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) f(\mathbf{z}) d\mathbf{z} &= \nabla_{\phi} \int r(\epsilon) f(\mathbf{z}) d\epsilon \\ &= \int r(\epsilon) \nabla_{\phi} f(g_{\phi}(\mathbf{x}, \epsilon)) d\epsilon \approx \nabla_{\phi} f(g_{\phi}(\mathbf{x}, \epsilon^*))\end{aligned}$$

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

$$\begin{aligned}\nabla_{\phi} \mathcal{L}(\phi, \theta) &= \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}, \phi) \log p(\mathbf{x}|\mathbf{z}, \theta) d\mathbf{z} - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z})) \\ &= \int r(\epsilon) \nabla_{\phi} \log p(\mathbf{x}|g_{\phi}(\mathbf{x}, \epsilon), \theta) d\epsilon - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z})) \\ &\approx \nabla_{\phi} \log p(\mathbf{x}|g_{\phi}(\mathbf{x}, \epsilon^*), \theta) - \nabla_{\phi} \text{KL}(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z}))\end{aligned}$$

### Variational assumption

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

$$\mathbf{z} = g_{\phi}(\mathbf{x}, \epsilon) = \sigma_{\phi}(\mathbf{x}) \cdot \epsilon + \mu_{\phi}(\mathbf{x}).$$

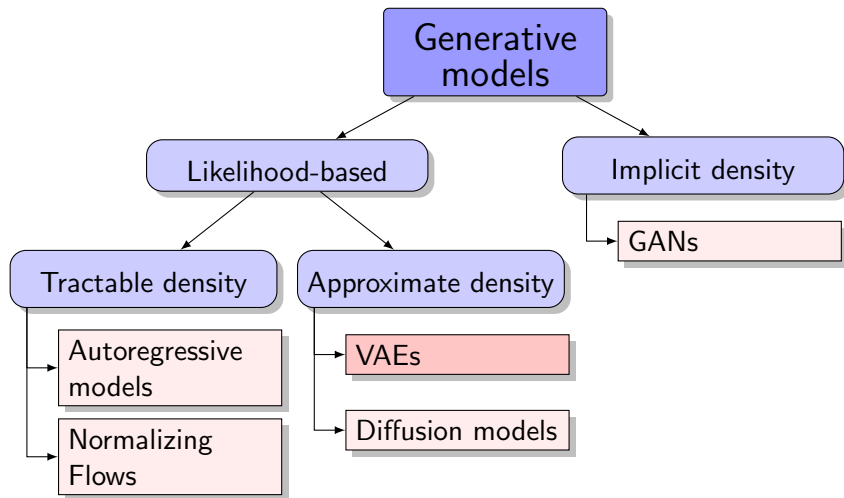
Here  $\mu_{\phi}(\cdot), \sigma_{\phi}(\cdot)$  are parameterized functions (outputs of neural network).

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

# Outline

1. ELBO gradients, reparametrization trick
2. Variational autoencoder (VAE)
3. Data dequantization

# Generative models zoo



# Variational autoencoder (VAE)

## Final EM-algorithm

- ▶ pick random sample  $\mathbf{x}_i, i \sim U[1, n]$ .
- ▶ compute the objective:

$$\epsilon^* \sim r(\epsilon); \quad \mathbf{z}^* = g(\mathbf{x}, \epsilon^*, \phi);$$

$$\mathcal{L}(\phi, \theta) \approx \log p(\mathbf{x}|\mathbf{z}^*, \theta) - KL(q(\mathbf{z}^*|\mathbf{x}, \phi)||p(\mathbf{z}^*)).$$

- ▶ compute a stochastic gradients w.r.t.  $\phi$  and  $\theta$

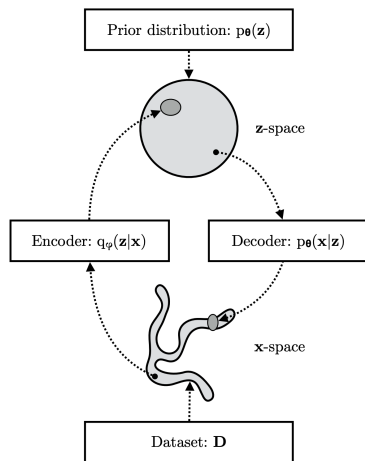
$$\begin{aligned}\nabla_{\phi} \mathcal{L}(\phi, \theta) &\approx \nabla_{\phi} \log p(\mathbf{x}|g_{\phi}(\mathbf{x}, \epsilon^*), \theta) - \nabla_{\phi} KL(q(\mathbf{z}|\mathbf{x}, \phi)||p(\mathbf{z})); \\ \nabla_{\theta} \mathcal{L}(\phi, \theta) &\approx \nabla_{\theta} \log p(\mathbf{x}|\mathbf{z}^*, \theta).\end{aligned}$$

- ▶ update  $\theta, \phi$  according to the selected optimization method (SGD, Adam, RMSProp):

$$\begin{aligned}\phi &:= \phi + \eta \nabla_{\phi} \mathcal{L}(\phi, \theta), \\ \theta &:= \theta + \eta \nabla_{\theta} \mathcal{L}(\phi, \theta).\end{aligned}$$

# Variational autoencoder (VAE)

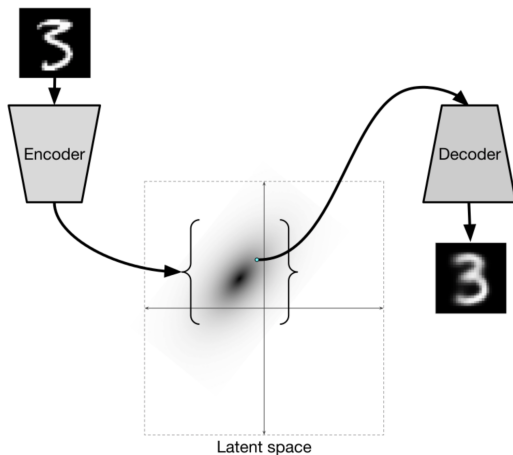
- ▶ VAE learns stochastic mapping between  $\mathbf{x}$ -space, from complicated distribution  $\pi(\mathbf{x})$ , and a latent  $\mathbf{z}$ -space, with simple distribution.
- ▶ The generative model learns a joint distribution  $p(\mathbf{x}, \mathbf{z}|\theta) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}, \theta)$ , with a prior distribution  $p(\mathbf{z})$ , and a stochastic decoder  $p(\mathbf{x}|\mathbf{z}, \theta)$ .
- ▶ The stochastic encoder  $q(\mathbf{z}|\mathbf{x}, \phi)$  (inference model), approximates the true but intractable posterior  $p(\mathbf{z}|\mathbf{x}, \theta)$  of the generative model.





# Variational Autoencoder

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



# Variational autoencoder (VAE)

- ▶ Encoder  $q(\mathbf{z}|\mathbf{x}, \phi) = \text{NN}_e(\mathbf{x}, \phi)$  outputs  $\mu_\phi(\mathbf{x})$  and  $\sigma_\phi(\mathbf{x})$ .
- ▶ Decoder  $p(\mathbf{x}|\mathbf{z}, \theta) = \text{NN}_d(\mathbf{z}, \theta)$  outputs parameters of the sample distribution.

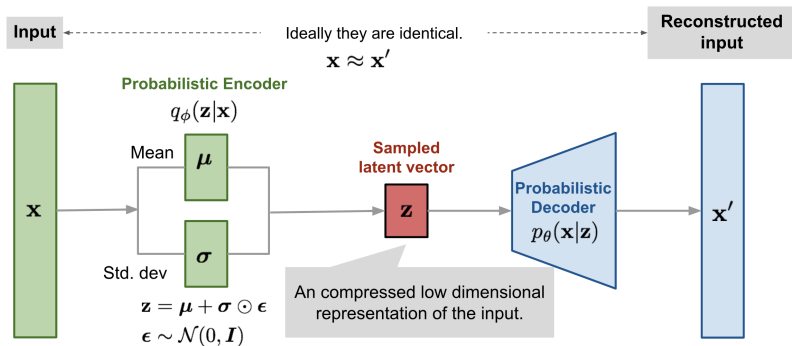


image credit:

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

# Outline

1. ELBO gradients, reparametrization trick
2. Variational autoencoder (VAE)
3. Data dequantization

## Discrete data vs continuous model

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

- ▶ Images (and not only images) are discrete data, pixels lie in the integer domain  $\{0, 255\}$ .
- ▶ By fitting a continuous density model  $p(\mathbf{x}|\theta)$  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.  $P(\mathbf{y}|\theta) = \text{Cat}(\pi(\theta))$ ).
- ▶ 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*

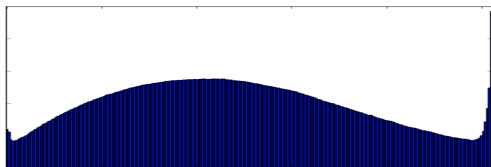
*Tran D. et al. Discrete flows: Invertible generative models of discrete data, 2019*

# Discrete data vs continuous model

## Continuous model

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

## CIFAR-10 pixel values distribution



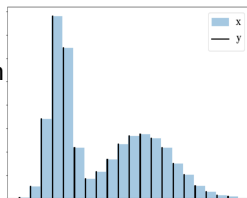
# Uniform dequantization

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

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



## Proof

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

# Summary

- ▶ Amortized variational inference allows to efficiently compute the stochastic gradients for ELBO using Monte-Carlo estimation.
- ▶ The reparametrization trick gets unbiased gradients w.r.t to the variational posterior distribution  $q(\mathbf{z}|\mathbf{x}, \phi)$ .
- ▶ The VAE model is an LVM with two neural network: stochastic encoder  $q(\mathbf{z}|\mathbf{x}, \phi)$  and stochastic decoder  $p(\mathbf{x}|\mathbf{z}, \theta)$ .
- ▶ Lots of data are discrete. We able to discretize the model or to dequantize our data to use continuous model.
- ▶ Uniform dequantization helps to make discrete data continuous. It gives us lower bound on the log-likelihood.