Variational Autoencoders

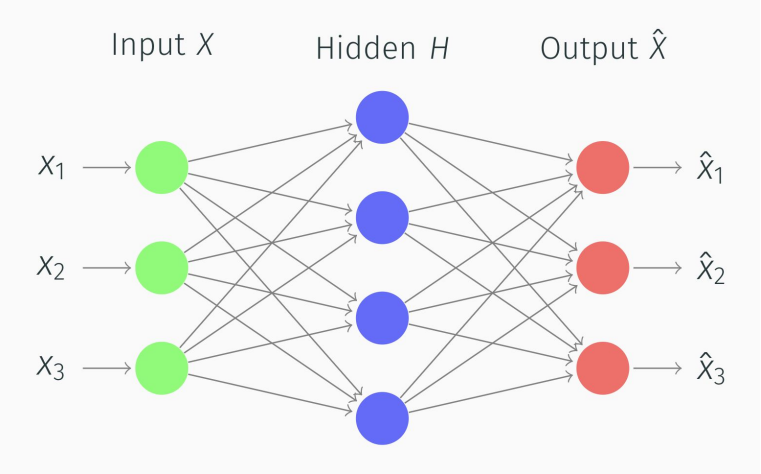
Chin-Wei Huang

(2020/03/11)

Autoencoders as Generative Models

Autoencoder $X \xrightarrow{g_\phi} H \xrightarrow{f_\theta} \hat{X}$

with loss function $l(\cdot, \cdot)$



Recover the distribution $p(H)$ of

the hidden units by regularizing H

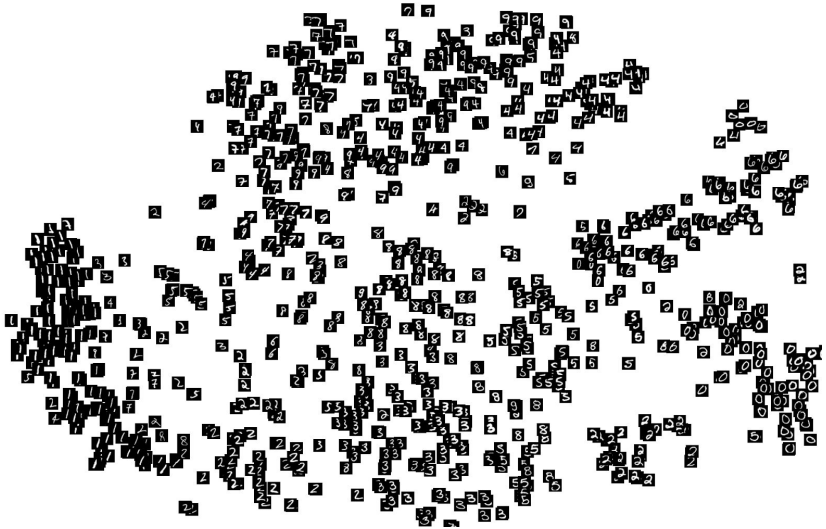


Figure from blog post: Latent Space Visualization — Deep Learning Bits by Julien Despois

A Probabilistic Regularizer

Regularized autoencoder's objective: $\sum_{i=1}^{N} \underbrace{l(f_\theta(g_\phi(x_i)), x_i)}_{reconstruction} + \lambda \underbrace{\Omega(\phi, \theta, g_\phi(x_i))}_{regularization}$

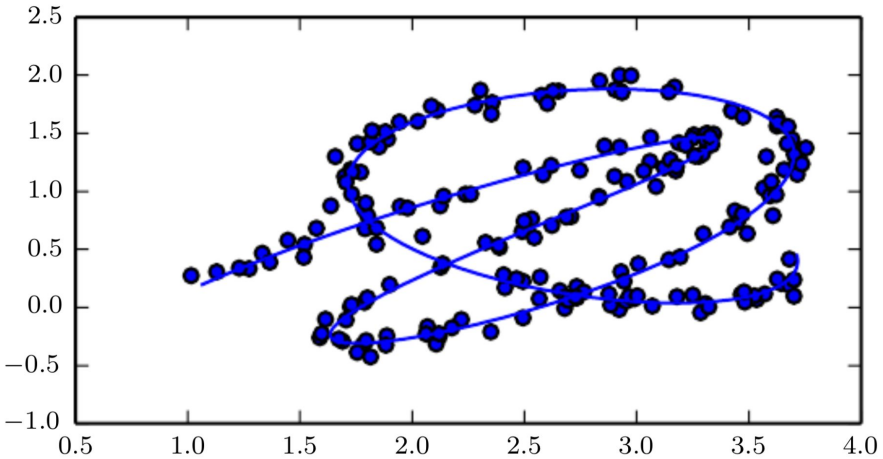
What do we want from the regularizer?

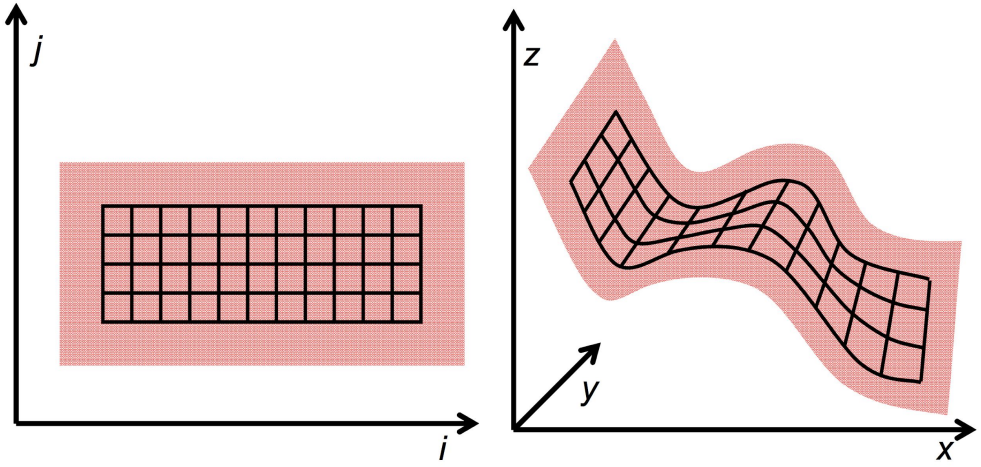
- we want the code $g_\phi(x)$ to be distributed by a prior $p(h)$
  (or we want to fit $p(h)$ on $g_\phi(x)$)

The *evidence lower bound* (ELBO, variational lower bound):

$$\hat{L}(x) = \underbrace{\log p_\theta(x|\mu_\phi(x) + \epsilon \odot \sigma_\phi(x))}_{reconstruction} - \overbrace{D_{KL}(q_\phi(z|x)||p(z))}^{regularization}$$

Manifold Hypothesis





Figures from the DL book (left) and \textit{3D Surface Parameterization Using Manifold Learning for Medial Shape Representation} by Ward and Hamarneh, 2007 (right)

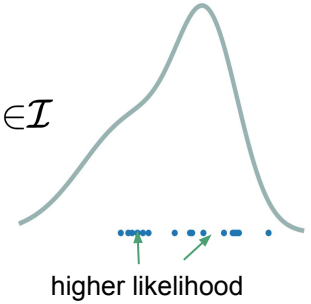
Goals of this lecture

- **Background**
- **Variational Autoencoder**
  - Variational Inference and ELBO
  - Reparameterization
  - Amortization
- **Hierarchical VAE** (hierarchical representation, lossy encoding)
- **Inference Suboptimality** (bias in VI)
- **Debiasing Variational Inference via Importance Sampling**
  - Importance weighted autoencoder
  - *Stochastically unbiased marginalization objective

Background

Maximum Likelihood Principle

dataset: $D = (x_i)_{i \in \mathcal{I}}$



Maximum Likelihood is an parameter estimation criterion that maximizes the likelihood of the dataset being generated by the model: $\log p_\theta(x_i)$

Maximum Conditional Likelihood maximizes the conditional likelihood of targets y_i following some mapping $\pi_\theta : \mathcal{X} \to \mathcal{Y}$: $\log p_\theta(y_i|x_i) = \log p(y_i; \pi_\theta(x_i))$

Maximum Marginal Likelihood maximizes the *observed* data likelihood under the marginal distribution (assuming some *latent/unobserved* variable)

$$\log p_\theta(x_i) = \log \sum_c p_\theta(x_i|c)p(c) \quad or \quad \log \int_{z \in \mathcal{Z}} p_\theta(x_i, z)dz$$

Expected Complete Data Likelihood

$$\log p_\theta(x_i) = \log \sum_c p_\theta(x_i|c)p(c) \quad or \quad \log \int_{z \in Z} p_\theta(x_i, z)dz$$

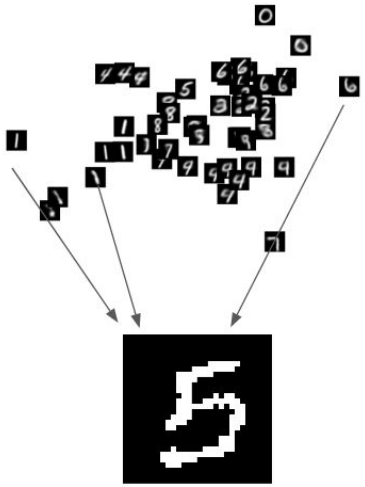
In general, the summation above can be computationally expensive to compute, and the integral is even not tractable. This is because z is not observed.

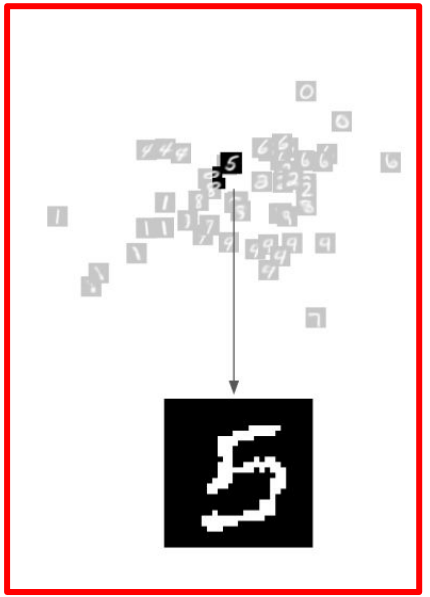
- If z is given, we have the complete-data likelihood $\log p(x, z)$
- Taking the expectation over some $q(z)$ (chosen to be the "*more likely*" z's that cause x) gives the **expected complete-data likelihood** (ECDL)

$$\mathbb{E}_{q(z)}\left[\log p(x, z)\right]$$

Approximate Inference

How to get $q(z)$?





$$q^*(z) = p(z|x)$$

Two major approaches:

- Markov Chain Monte Carlo (MCMC):
  - a family of sampling methods
  - (temporally) asymptotically unbiased but high variance
- Variational Inference (VI):
  - a family of optimization methods
  - low variance but biased

Variational Autoencoder

Variational Inference

Let **x** denote the data and **z** denote the corresponding latent variable, following the joint distribution $p(x, z) = p(x|z)p(z)$

- Sampling from the model: ancestral sampling
- Evaluating the “marginal likelihood” of some x

Variational Inference

Let **x** denote the data and **z** denote the corresponding latent variable, following the joint distribution $p(x, z) = p(x|z)p(z)$

- Sampling from the model: ancestral sampling
- Evaluating the “marginal likelihood” of some x

** Jensen’s inequality*

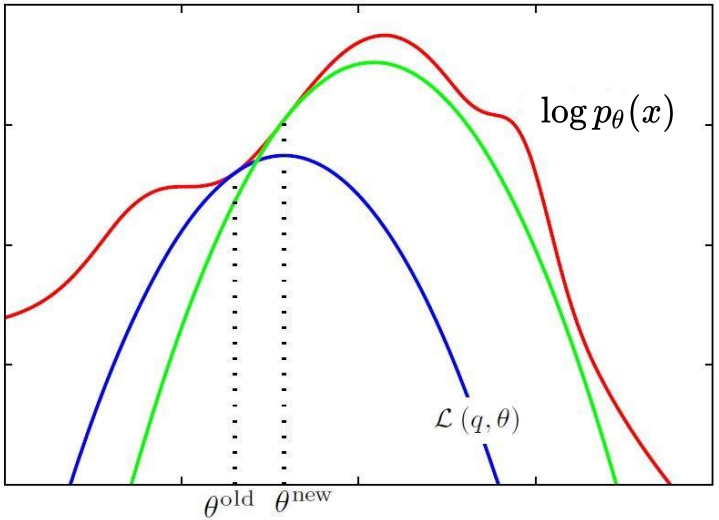
$$\log p(x) = \log \int_z p(x,z)dz = \log \int_z \frac{p(x,z)}{q(z)} q(z)dz = \log \mathbb{E}_{z\sim q(z)}\left[\frac{p(x,z)}{q(z)}\right]$$

$$\geq^* \mathbb{E}_{z\sim q(z)}\left[\log \frac{p(x,z)}{q(z)}\right] = \underbrace{\mathbb{E}_{q(z)}\left[\log p(x,z)\right]}_{\text{ECDL}} + \underbrace{H(q)}_{\text{entropy}} =: L[q]$$

Variational Expectation Maximization

Traditionally, we would solve this variational problem exactly if the true posterior is computable. If we can't, we let q be within a "variational family" Q, find the maximizer of the lower bound, update the model parameter of $p_\theta(x|z)$.

How EM maximizes the likelihood



For computational tractability (simplicity), we assume Q is the family of multivariate Gaussian distributions with diagonal covariance matrix. (When the variables are assumed to be independent, we also call it the *mean field* approximation.)

Figure from blog post: Inference using EM algorithm by Rishabh Misra, 2019

Optimization of the “Model”

Let θ, ϕ denote the parameters of $p_\theta(x|z)$, $q_\phi(z)$.

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] := L(\theta,\phi)$$

Main idea (1): if the lower bound is a good approximation to (log) marginal likelihood, then maximizing it is approximately equivalent to maximizing the marginal

Estimate the gradient using Monte Carlo

$$\nabla_\theta L(\theta,\phi) = \mathbb{E}_{z\sim q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right] \approx \frac{1}{m}\sum_{j=1}^{m} \nabla_\theta \log p_\theta(x,z_j)$$

where $z_j \sim q_\phi(z)$ i.i.d.

Optimization of the “Variational Distribution”

Main idea (2): maximizing the lower-bound by tuning the variational distribution q so that (1) makes more sense

$$\nabla_\phi L(\theta, \phi) = \nabla_\phi \int q_\phi(z) \left(\log p_\theta(x, z) - \log q_\phi(z)\right) dz$$

1. Score function estimator (REINFORCE)
2. Bonnet and Price estimator
3. Path derivative gradient estimator (reparameterization trick)

Why choose one estimator over another?

- Error: bias and variance
- Efficiency (variance)

Path Derivative Gradient Estimator (Reparameterization)

> Intuitively, the reparameterization trick provides more informative gradients by exposing the dependence of sampled latent variables z on variational parameters φ. In contrast, the REINFORCE gradient estimate only depends on the relationship between the density function log qφ(z) and its parameters. (Roeder et al, 2017)

Given some function g_ϕ such that for some $\epsilon \sim q_\epsilon$, $g_\phi(\epsilon) \overset{d}{=} z \sim q_\phi(z)$

g_ϕ depends on ϕ but q_ϵ does not.

Path Derivative Gradient Estimator (Reparameterization)

> Intuitively, the reparameterization trick provides more informative gradients by exposing the dependence of sampled latent variables z on variational parameters φ. In contrast, the REINFORCE gradient estimate only depends on the relationship between the density function log qφ(z) and its parameters. (Roeder et al, 2017)

Given some function g_ϕ such that for some $\epsilon \sim q_\epsilon$, $g_\phi(\epsilon) \overset{d}{=} z \sim q_\phi(z)$

g_ϕ depends on ϕ but q_ϵ does not.

$$\nabla_\phi L(\theta, \phi) = \nabla_\phi \mathbb{E}_{z \sim q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z)\right] = \nabla_\phi \mathbb{E}_{\epsilon \sim q_\epsilon(\epsilon)} \left[\log p_\theta(x, g_\phi(\epsilon)) - \log q_\phi(g_\phi(\epsilon))\right]$$

$$= \mathbb{E}_{\epsilon \sim q(\epsilon)} \left[\nabla_\phi (\log p_\theta(x, g_\phi(\epsilon)) - \log q_\phi(g_\phi(\epsilon)))\right] \approx \frac{1}{m} \sum_{j=1}^{m} \nabla_\phi (\log p_\theta(x, g_\phi(\epsilon_j)) - \log q_\phi(g_\phi(\epsilon_j)))$$

Gaussian Reparameterization

For example, if $q_\phi(z) = \mathcal{N}(z; \mu, \sigma^2)$, where the trainable parameters are $\phi = (\mu, \sigma)$

we can reparameterize z as $z := \mu + \epsilon \odot \sigma$ where $\epsilon \sim \mathcal{N}(\epsilon; \mathbf{0}, \mathbf{I})$

Verify $g_\phi(\epsilon) \overset{d}{=} z \sim q_\phi(z)$

- Characteristic function $\psi_X(t) = \mathbb{E}e^{itX}$
- Change of variable density formula

$$q(g(\epsilon)) = q_\epsilon(\epsilon) \left| \det \left(\frac{\partial g(\epsilon)}{\partial \epsilon} \right) \right|^{-1}$$

Comparing Gradient Estimators

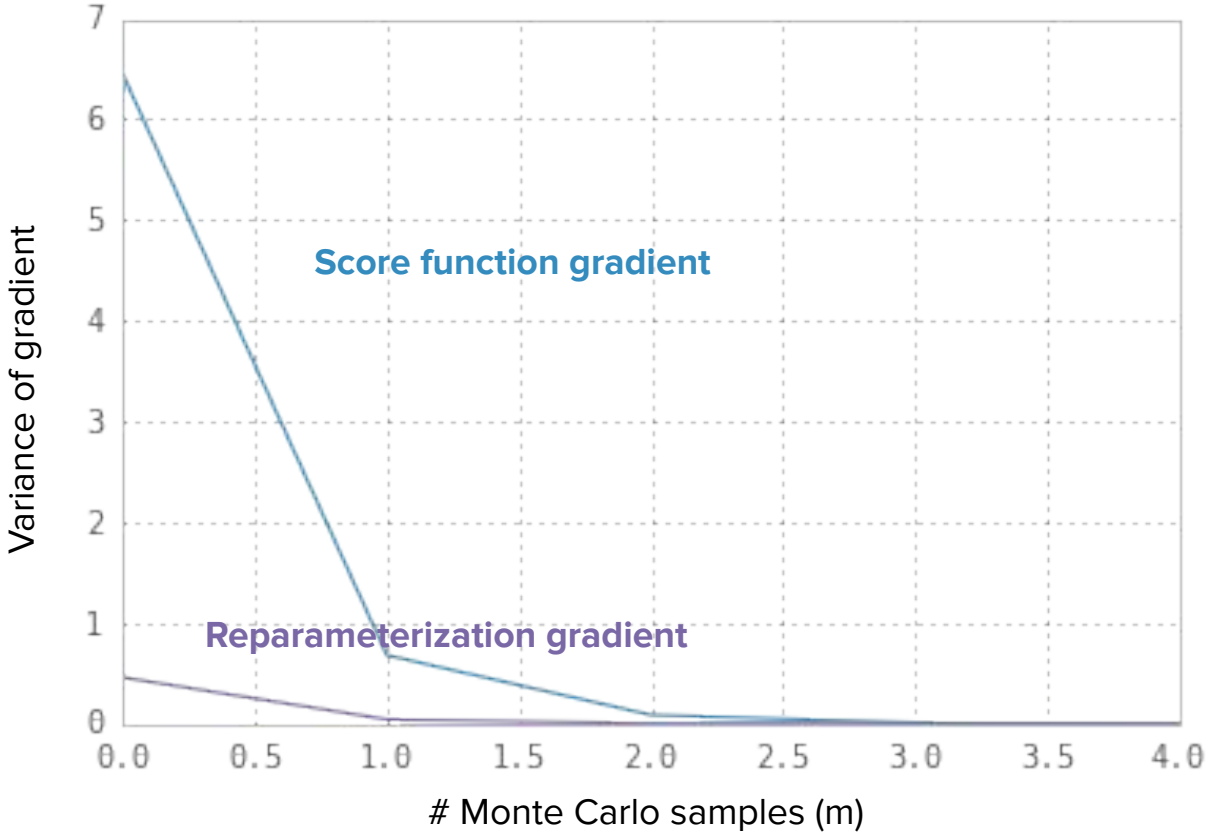


Figure from blog post: *Reparameterization Trick* by Goker Erdogan

Amortized Variational Inference

- Traditional variational inference

  For every data point $\boldsymbol{x_i}$, we need to parameterize its variational approximation $\boldsymbol{q_i}$.

  - $\mathcal{O}(n)$ storage.
  - Updating $\boldsymbol{q_i}$ does not affect $\boldsymbol{q_j}$

- **Amortized** variational inference

  Use an **inference network** (aka **encoder**) for all $\boldsymbol{x_i}$.

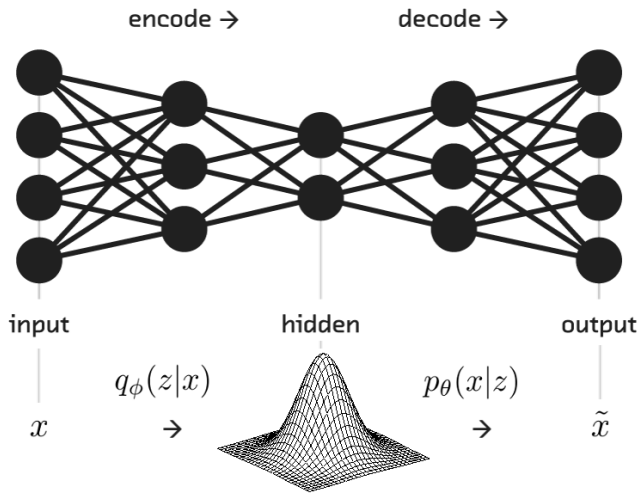


Figure from blog post: *Under the Hood of the Variational Autoencoder* by Miriam Shiffman

Vanilla VAE

Putting it all together $p(z) = \mathcal{N}(z; \mathbf{0}, \mathbf{I}), \quad q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma_\phi(x)^2)$

$$\nabla_{\theta,\phi} L(\theta, \phi) = \nabla_{\theta,\phi} \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))$$

$$\approx \nabla_{\theta,\phi} \underbrace{\log p_\theta(x|\mu_\phi(x) + \epsilon \odot \sigma_\phi(x))}_{reconstruction} - \overbrace{D_{KL}(q_\phi(z|x)||p(z))}^{regularization}$$

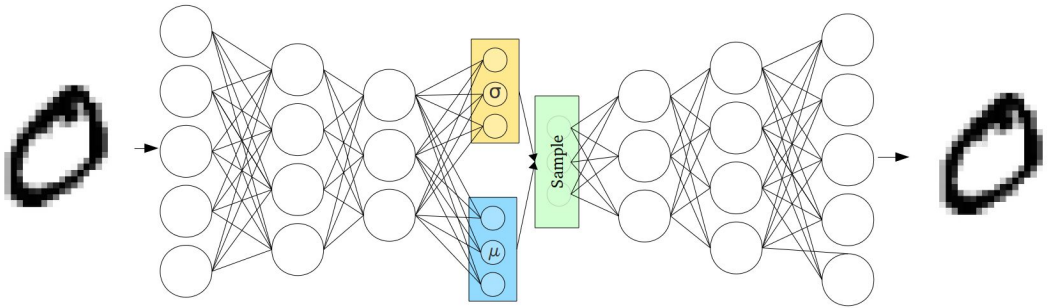


Figure from blog post: *Intuitively Understanding Variational Autoencoders* by Irhum Shafkat

Evaluating the KL divergence

Generally, the KL divergence can be estimated using Monte Carlo (with reparameterization so that the gradient can be estimated)

$$\approx \frac{1}{m}\sum_{j=1}^{m} \log q_\phi(z = g_\phi(\epsilon_j, x)|x) - \log p(z = g_\phi(\epsilon_j, x))$$

If both q and p are Gaussian, the KL has a convenient form

$$D_{KL}(\mathcal{N}(z;\mu,\sigma^2)||\mathcal{N}(z;\mathbf{0},\mathbf{I})) = \frac{1}{2}\sum_{d=1}^{D}\left(-1 - \log\sigma_d^2 + \mu_d^2 + \sigma_d^2\right)$$

(D is the dimensionality of the latent variable z)

```
enc = nn.Sequential(
    ...
    nn.Linear(dimh, dimz*2)
)
dec = nn.Sequential( ... )
```

Defining the model

```
q_params = enc(x) ## x is input
mu = q_params[:, :dimz]
log_sigma = q_params[:,dimz:]
sigma = torch.exp(log_sigma) + 1e-7
e = torch.randn(batch_size, dimz)
z = mu + sigma * e
x_ = dec(z) ## x_ is reconstruction
```

Building the stochastic computational graph

```
KL = analytic_KL(mu, log_sigma)
logpx_z = - recon_loss(x_, x)
loss = - (logpx_z - KL).mean()
```

Computing the loss

Vanilla VAE as Generative Model

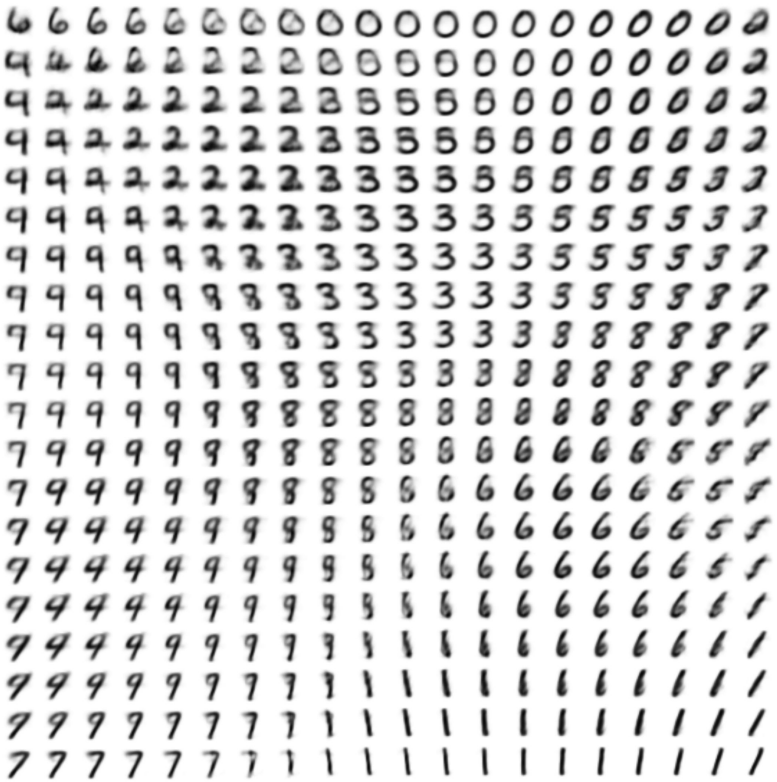


Figure from *Auto-Encoding Variational Bayes* by Kingma and Welling, 2013

Evaluation on Holdout set: Marginal Likelihood

p(x) can be estimated by importance sampling (which is *consistent*)

$$\frac{1}{K}\sum_{k=1}^{K}\frac{p(x,z_k)}{q(z_k|x)} \qquad z_k \overset{iid}{\sim} q(z|x)$$

Likewise, log marginal can also be estimated by

log makes it biased (Jensen's) but still consistent (asymptotically unbiased + vanishing variance)

$$\log p(x) \approx L\sum_{k=1}^{K}E\{\log p(x,z_k) - \log q(z_k|x)\} - \log K$$

Further reading: *Annealed Importance Sampling* by Radford Neal (1998) and *On the Quantitative Analysis of Decoder-Based Generative Models by Wu et al. (2016)*

Table 1. Comparison of negative log-probabilities on the test set for the binarised MNIST data.

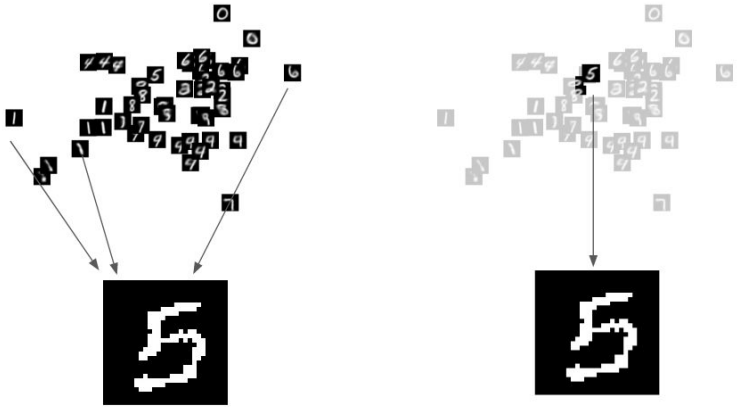
Model	$-\ln p(\mathbf{v})$
Factor Analysis	106.00
NLGBN (Frey & Hinton, 1999)	95.80
Wake-Sleep (Dayan, 2000)	91.3
DLGM diagonal covariance	87.30
DLGM rank-one covariance	86.60
Results below from Uria et al. (2014)	
MoBernoullis K=10	168.95
MoBernoullis K=500	137.64
RBM (500 h, 25 CD steps) approx.	86.34
DBN 2hl approx.	84.55
NADE 1hl (fixed order)	88.86
NADE 1hl (fixed order, RLU, minibatch)	88.33
EoNADE 1hl (2 orderings)	90.69
EoNADE 1hl (128 orderings)	87.71
EoNADE 2hl (2 orderings)	87.96
EoNADE 2hl (128 orderings)	85.10

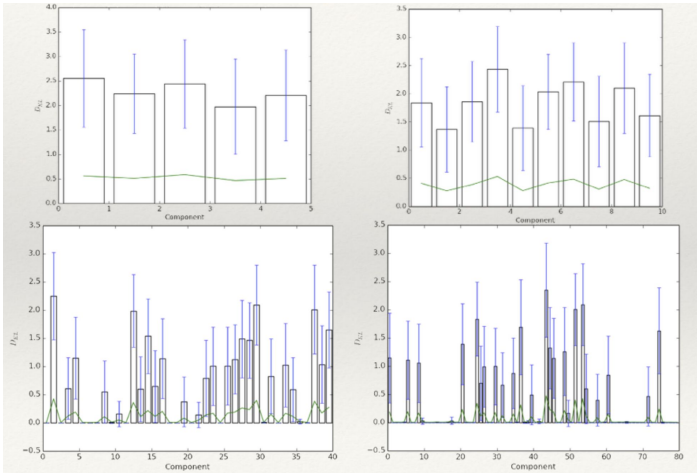
Table from *Stochastic Backpropagation and Approximate Inference in Deep Generative Models* by Rezende et al., 2014

Caveat on Regularization and Tricks

We don't really want the KL divergence, aks the “information gain”, to go to zero.

- Deterministic warm-up $\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \beta D_{KL}(q_\phi(z|x)||p(z))$
- Free-bits $\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \max\left\{\lambda, D_{KL}(q_\phi(z|x)||p(z))\right\}$





Second figure from slides *Training Neural Bayesian Nets* by Laurent Dinh and Vincent Dumoulin

The road so far

- **Background**
- **Variational Autoencoder**
  - Variational Inference and ELBO
  - Reparameterization
  - Amortization
- **Hierarchical VAE** (hierarchical representation, lossy encoding)
- **Inference Suboptimality** (bias in VI)
- **Debiasing Variational Inference via Importance Sampling**
  - Importance weighted autoencoder
  - *Stochastically unbiased marginalization objective

Hierarchical VAE

Hierarchical Models

Hierarchical generative model:

$$p(x, z_1, z_2, \ldots, z_L) = p(x|z_1)p(z_1|z_2)\ldots p(z_{L-1}|z_L)p(z_L)$$

Recognition network (independence):

$$q(z_1, z_2, \ldots, z_L|x) = q(z_1|x)\ldots q(z_L|x)$$

Exercise for the reader:
Derive the ELBO for this model using the recognition network as the variational distribution

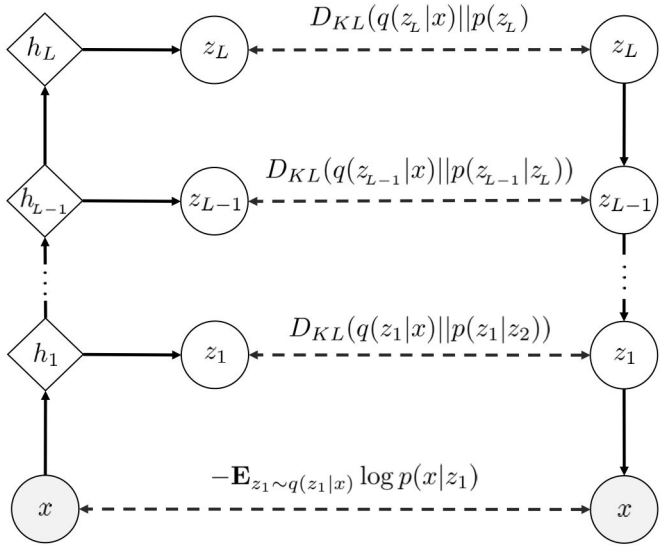


Figure from *PixelVAE: A Latent Variable Model for Natural Images* by Gulrajani el al., 2016

Best of Both Worlds: PixelCNN + VAE = PixelVAE

Autoregressive models are good at modeling local statistical dependencies, but lack *global coherence*.

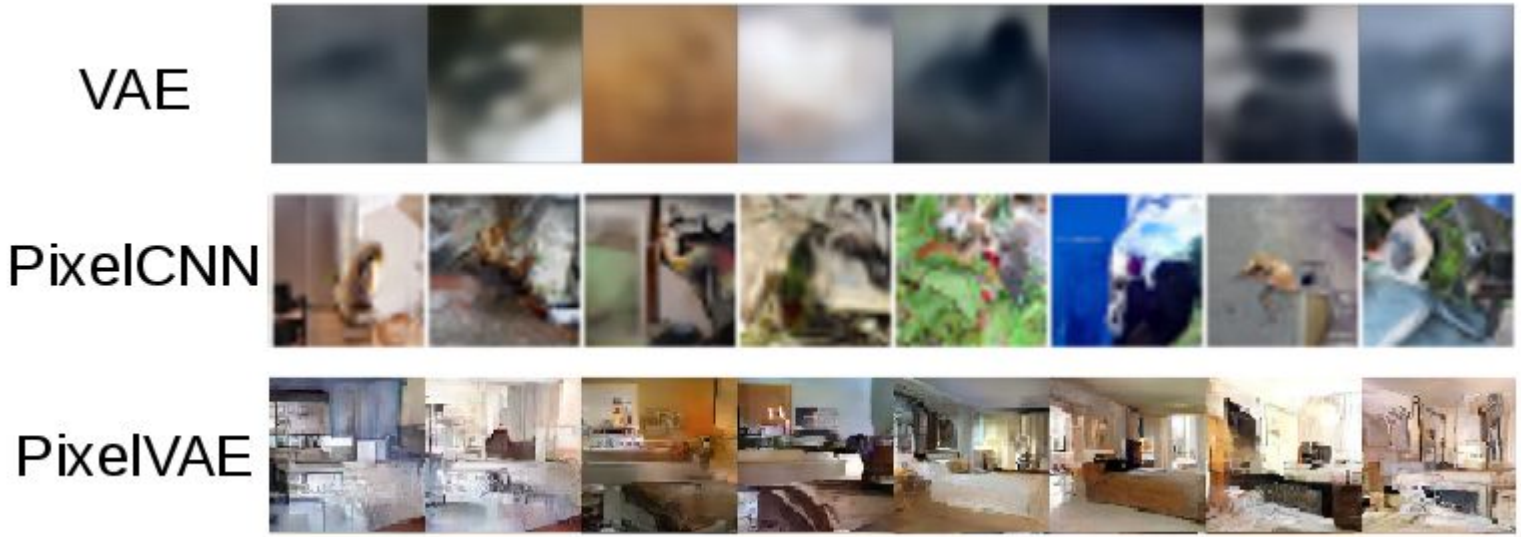


Figure from *PixelVAE: A Latent Variable Model for Natural Images* by Gulrajani el al., 2016

Latent Variable Model with Autoregressive Decoder

Autoregressive decoder: $p(x|z) = \prod_i p(x_i|x_1, \ldots, x_{i-1}, z)$

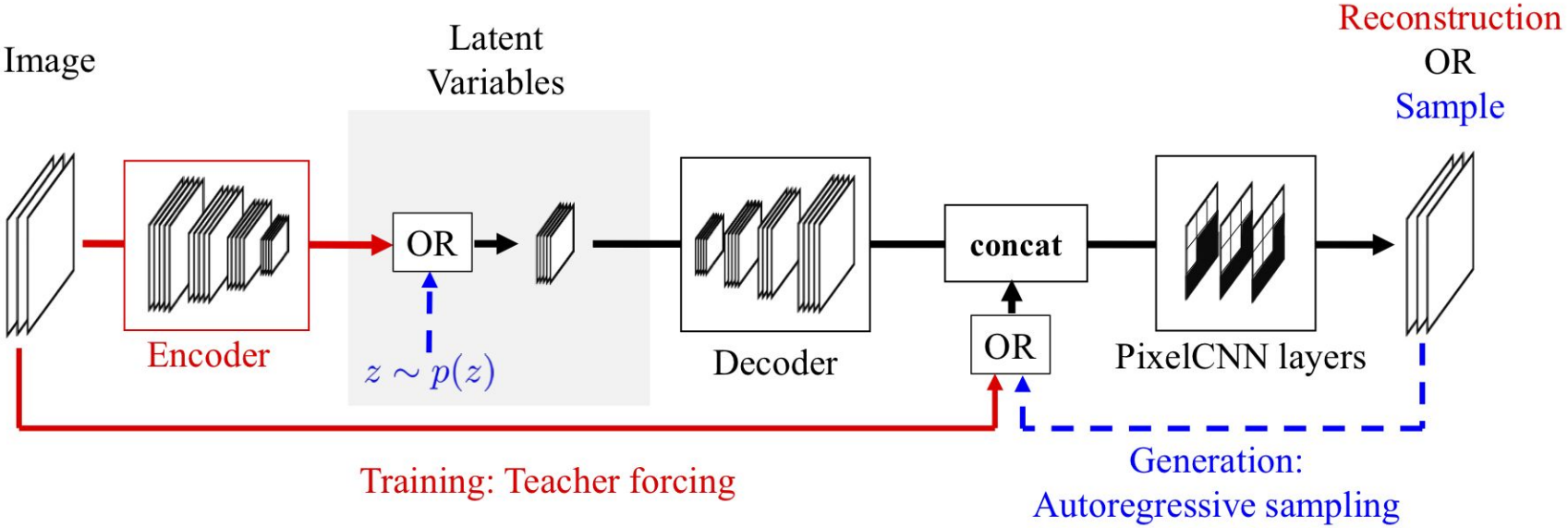


Figure from *PixelVAE: A Latent Variable Model for Natural Images* by Gulrajani el al., 2016

Hierarchical Representation

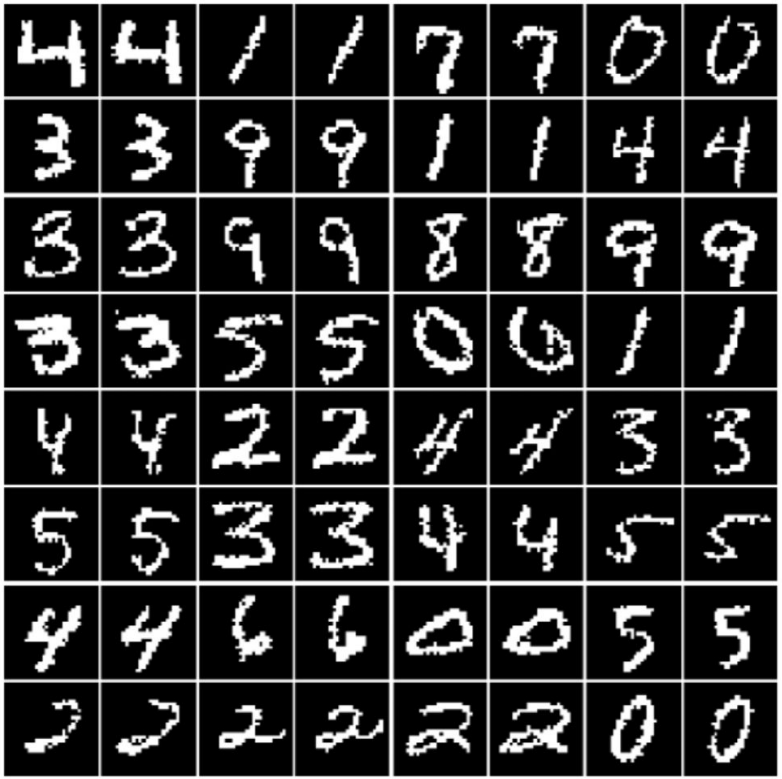
$$z_2 \sim p(z_2)$$

$$z_1 \sim p(z_1|z_2)$$

$$x \sim p(x|z_1, z_2)$$

Figure from *PixelVAE: A Latent Variable Model for Natural Images* by Gulrajani el al., 2016

Lossy Encoding



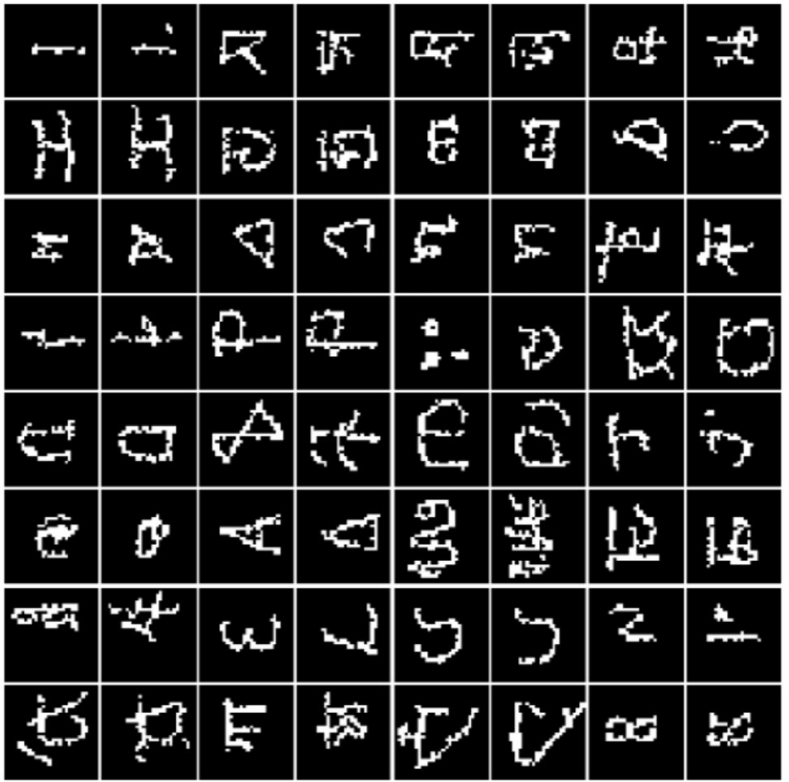
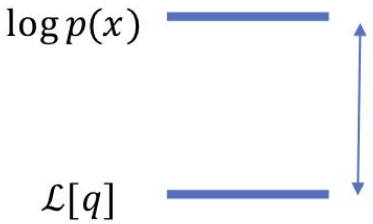


Figure from *Variational Lossy Autoencoder* by Chen el al., 2016

Inference Suboptimality

Variational Gap

An alternative derivation of the ELBO

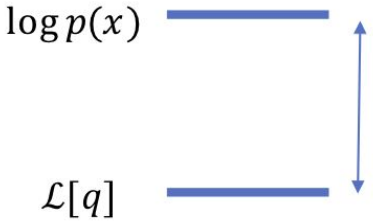


The KL between the “posteriors” is called the **variational gap**

Variational Gap

An alternative derivation of the ELBO

$$\begin{aligned}\log p(x) - D_{KL}(q(z|x)||p(z|x)) &= \log p(x) - \mathbb{E}_{q(z|x)}[\log q(z|x) - \log p(z|x)] \\ &= \log p(x) + \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{p(x)} - \log q(z|x)\right] \\ &= \mathbb{E}_{q(z|x)}[\log p(x,z) - \log q(z|x)] := L[q(z|x)]\end{aligned}$$



The KL between the “posteriors” is called the **variational gap**

Decomposing Variational Gap

Let $q^* = \arg\max_{q \in \mathcal{Q}} L[q]$ denote the the best variational approximation within the family of variational distributions Q (e.g. all diagonal Gaussian distributions), where L is the ELBO on the marginal likelihood of a specific datapoint x

$$\log p(x) - L[q(z|x)]$$
$$= \underbrace{\log p(x) - L[q^*]}_{\text{Approximation gap}} + \underbrace{L[q^*] - L[q(z|x)]}_{\text{Amortization gap}}$$

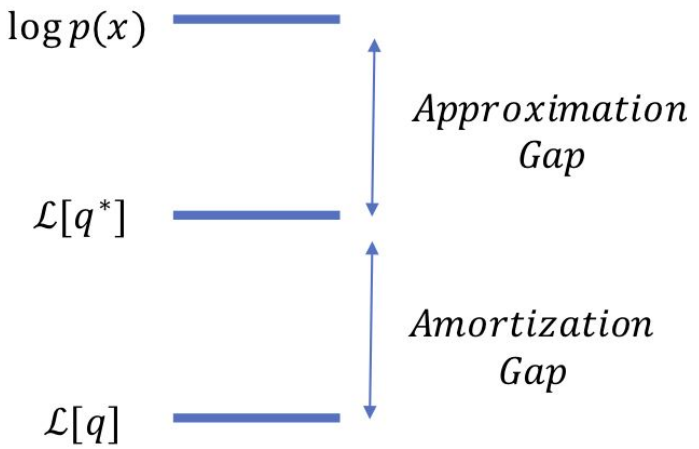


Figure from *Inference Suboptimality in Variational Autoencoders* by Cremer et al., 2018

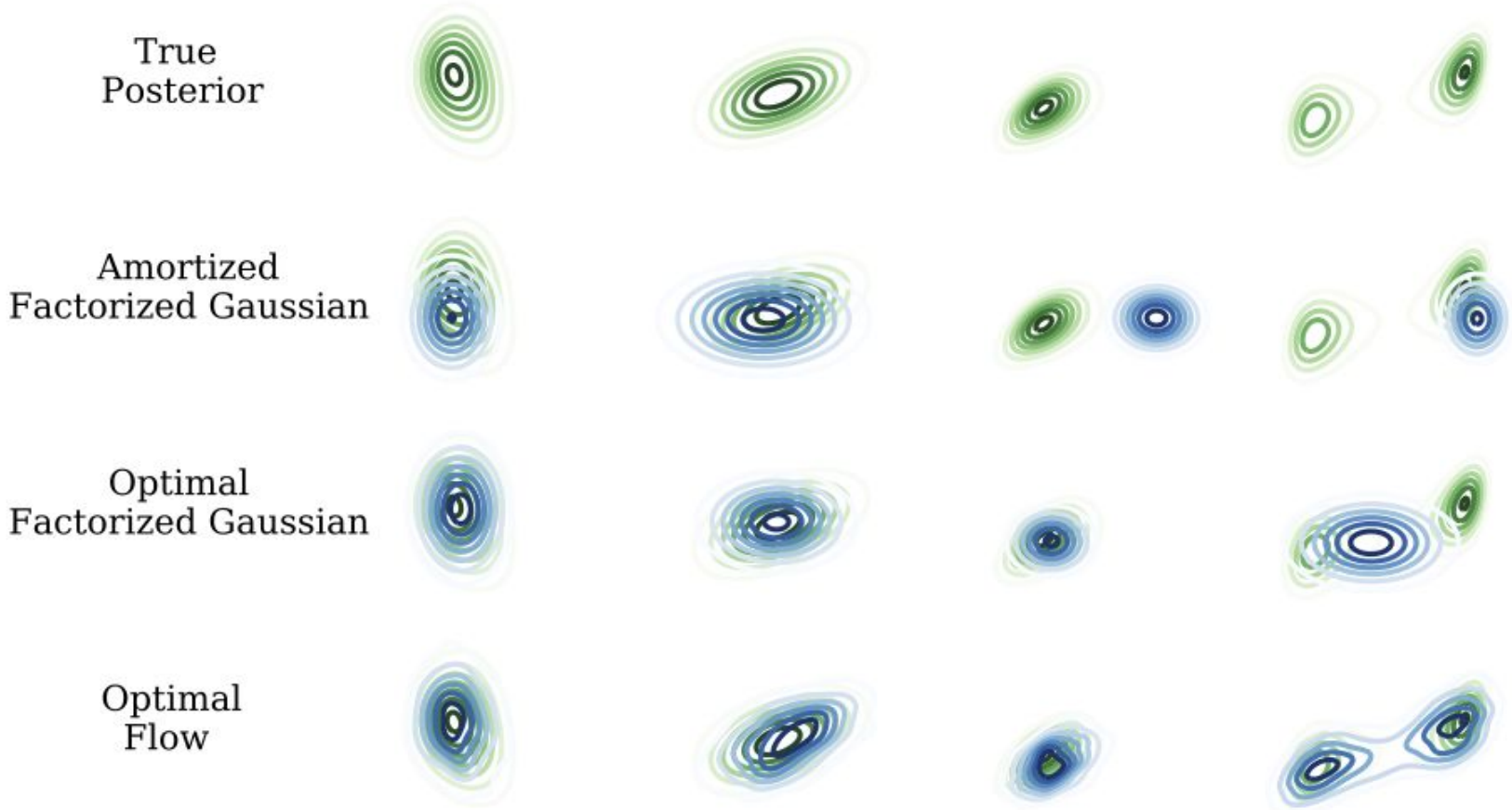
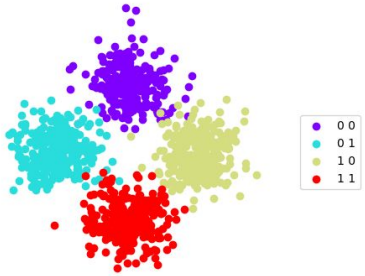


Figure from *Inference Suboptimality in Variational Autoencoders* by Cremer et al., 2018

Combating Inference Suboptimality

- Reduce the amortization gap
  - larger encoder
  - alternating updates of ϕ and θ
  - meta learning, semi-amortized
- Using larger family of $\mathcal{Q}$, non-Gaussian variational distribution
  - mixture distribution
  - normalizing flows (change of variable density)
  - hierarchical VI
  - implicit VI
- Combining variational inference with MCMC
- Using the information of the decoder and more samples (importance sampling)



Debiasing VI via Importance Sampling

Importance Sampling

Intuition: average likelihood ratio concentrates around the marginal likelihood

$$\int p(x,z)dz = \mathbb{E}_q\left[\frac{p(x,z)}{q(z)}\right] \approx \frac{1}{K}\sum_{k=1}^{K}\frac{p(x,z_k)}{q(z_k)}$$

$$z_k \overset{iid}{\sim} q(z)$$

Deterministic =>
Jensen's Inequality becomes equality

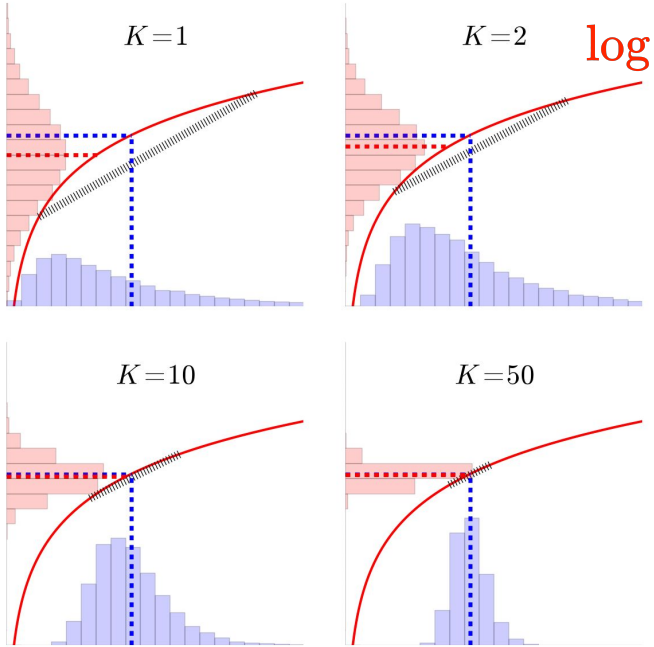


Figure from *Note on the bias and variance of variational inference* by Huang and Courville, 2019

Importance Weighted Autoencoder (IWAE)

A family of objectives (depending on K):

$$L_K(\theta, \phi) := \mathbb{E}_{z_{1:K} \sim \prod_{k=1}^K q_\phi(z_k|x)} \left[\log \frac{1}{K} \sum_{k=1}^K \frac{p_\theta(x, z_k)}{q_\phi(z_k)} \right]$$

Theorem (Monotonicity)

For all positive integer K, $\log p(x) \geq L_{K+1} \geq L_K$

Furthermore, under some mild assumption, $\lim_{K \to \infty} L_K = \log p(x)$

Training IWAE with Reparameterization

Let $\tilde{w}_k = \frac{w_k}{\sum_{k'} w_{k'}}, \; w_k = \frac{p_\theta(x, z_k)}{q_\phi(z_k|x)}, \; z_k = \mu_\phi(x) + \epsilon_k \odot \sigma_\phi(x)$

VAE update

$$\nabla L(\theta, \phi) \approx \sum_{k=1}^{K} \frac{1}{K} \nabla \log w_k$$

IWAE update

$$\nabla L_K(\theta, \phi) \approx \sum_{k=1}^{K} \tilde{w}_k \nabla \log w_k$$

		MNIST				OMNIGLOT			
		VAE		IWAE		VAE		IWAE	
# stoch. layers	k	NLL	active units	NLL	active units	NLL	active units	NLL	active units
1	1	86.76	19	86.76	19	108.11	28	108.11	28
	5	86.47	20	85.54	22	107.62	28	106.12	34
	50	86.35	20	84.78	25	107.80	28	104.67	41
2	1	85.33	16+5	85.33	16+5	107.58	28+4	107.56	30+5
	5	85.01	17+5	83.89	21+5	106.31	30+5	104.79	38+6
	50	84.78	17+5	82.90	26+7	106.30	30+5	103.38	44+7

Table from *Importance Weighted Autoencoders* by Burda et al., 2015

IWAE as VAE

The IWAE lower bound can be thought of as using a "*corrected*" variational distribution for the ELBO of VAE

$$\tilde{q}_{IW}(z|x, z_{2:K}) := \frac{p(x,z)}{\frac{1}{K}\left(\frac{p(x,z)}{q(z|x)} + \sum_{k=2}^{K} \frac{p(x,z_k)}{q(z_k|x)}\right)}$$

$$\rightarrow \frac{p(x,z)}{p(x)} \quad \text{almost surely as} \quad K \rightarrow \infty$$

$$L_K[q] = \mathbb{E}_{z_{2:K}}\left[L[\tilde{q}_{IW}(z|x, z_{2:K})]\right]$$

$$\leq L[q_{EW}]$$

$$q_{EW} := \mathbb{E}_{z_{2:K}}\left[\tilde{q}_{IW}(z|x, z_{2:K})\right]$$

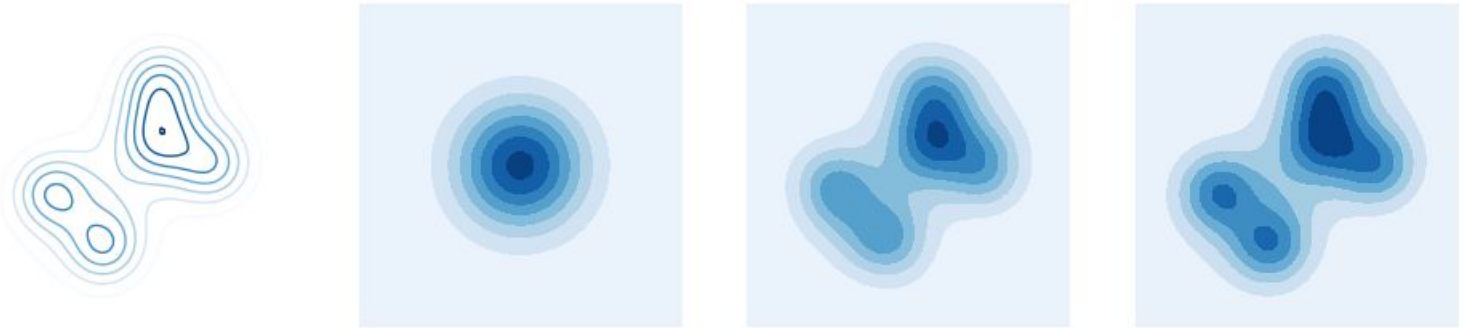


Figure from *Reinterpreting Importance-Weighted Autoencoders* by Cremer et al., 2017

IWAE as VAE

The IWAE lower bound can be thought of as using a "*corrected*" variational distribution for the ELBO of VAE

$$\tilde{q}_{IW}(z|x, z_{2:K}) := \frac{p(x,z)}{\frac{1}{K}\left(\frac{p(x,z)}{q(z|x)} + \sum_{k=2}^{K} \frac{p(x,z_k)}{q(z_k|x)}\right)}$$

$$\rightarrow \frac{p(x,z)}{p(x)} \quad \text{almost surely as} \quad K \rightarrow \infty$$

$$L_K[q] \overset{1.}{=} \mathbb{E}_{z_{2:K}}\left[L[\tilde{q}_{IW}(z|x, z_{2:K})]\right] \overset{2.}{\leq} L[q_{EW}]$$

$$q_{EW} := \mathbb{E}_{z_{2:K}}\left[\tilde{q}_{IW}(z|x, z_{2:K})\right]$$

Exercises for the reader:

- Show q_{EW} is a probability density function (integrating to 1).
- Show equality 1. and inequality 2.

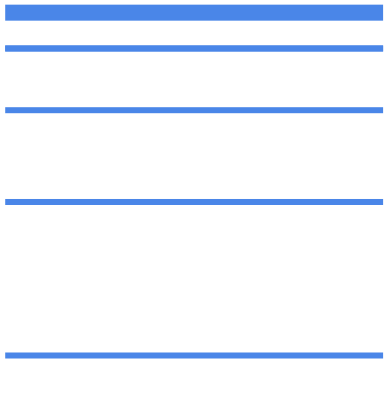
Figure from *Reinterpreting Importance-Weighted Autoencoders* by Cremer et al., 2017

Unbiased Estimation of Marginal Likelihood

How to achieve the asymptotic unbiasedness?

Idea:

- Telescoping summation trick
- Estimating infinite series



$$L_\infty = L_1 + (L_2 - L_1) + (L_3 - L_2) + \cdots$$
$$= L_1 + \sum_{K=1}^{\infty} L_{K+1} - L_K$$

$$IW_K := \log \frac{1}{K} \sum_{k=1}^{K} \frac{p(x,z_j)}{q(z_j|x)} \qquad L_K = \mathbb{E}[IW_K]$$

$$\Delta_K := IW_{K+1} - IW_K$$

$$L_\infty = \mathbb{E}[IW_1 + \textstyle\sum_{K=1}^{\infty} \Delta_K]$$

Russian Roulette Estimator

Enter the “Russian roulette” estimator (Kahn, 1955). Suppose we want to estimate

$$\sum_{k=1}^{\infty} \Delta_k$$

(Require $\sum_{k=1}^{\infty} |\Delta_k| < \infty$)

Slide taken from *Residual Flows for Invertible Generative Modeling* by Ricky Chen, 2019

Russian Roulette Estimator

Enter the “Russian roulette” estimator (Kahn, 1955). Suppose we want to estimate

$$\sum_{k=1}^{\infty} \Delta_k$$

(Require $\sum_{k=1}^{\infty} |\Delta_k| < \infty$)

Flip a coin b with probability q.

$$\begin{aligned}
&\mathbb{E}\left[\Delta_1 + \left[\frac{1}{1-q}\sum_{k=2}^{\infty}\Delta_k\right]\mathbb{1}_{b=0} + [0]\mathbb{1}_{b=1}\right] \\
&= \Delta_1 + \left[\frac{1}{1-q}\sum_{k=2}^{\infty}\Delta_k\right](1-q) \\
&= \sum_{k=1}^{\infty}\Delta_k
\end{aligned}$$

Slide taken from *Residual Flows for Invertible Generative Modeling* by Ricky Chen, 2019

Russian Roulette Estimator

Enter the “Russian roulette” estimator (Kahn, 1955). Suppose we want to estimate

$$\sum_{k=1}^{\infty} \Delta_k$$

(Require $\sum_{k=1}^{\infty} |\Delta_k| < \infty$)

Flip a coin b with probability q.

$$\begin{aligned}
&\mathbb{E}\left[\Delta_1 + \left[\frac{1}{1-q}\sum_{k=2}^{\infty}\Delta_k\right]\mathbb{1}_{b=0} + [0]\mathbb{1}_{b=1}\right] \\
&= \Delta_1 + \left[\frac{1}{1-q}\sum_{k=2}^{\infty}\Delta_k\right](1-q) \\
&= \sum_{k=1}^{\infty}\Delta_k
\end{aligned}$$

Has probability q of being evaluated in **finite** time.

Slide taken from *Residual Flows for Invertible Generative Modeling* by Ricky Chen, 2019

Russian Roulette Estimator

If we repeatedly apply the same procedure *infinitely many times*, we obtain an unbiased estimator of the infinite series.

$$\sum_{k=1}^{\infty} \Delta_k = \mathbb{E}_{n \sim p(N)} \left[\sum_{k=1}^{n} \frac{\Delta_k}{\mathbb{P}(N \geq k)} \right]$$

Computed in **finite** time with **prob. 1**!!

Slide modified from *Residual Flows for Invertible Generative Modeling* by Ricky Chen, 2019

Russian Roulette Estimator

If we repeatedly apply the same procedure *infinitely many times*, we obtain an unbiased estimator of the infinite series.

$$\sum_{k=1}^{\infty} \Delta_k = \mathbb{E}_{n \sim p(N)} \left[\sum_{k=1}^{n} \frac{\Delta_k}{\mathbb{P}(N \geq k)} \right]$$

Directly sample the first successful coin toss.

Computed in **finite** time with **prob. 1**!!

$p(N)$ is a geometric distribution in the case of coin toss.
It can be any distribution as long as $p(N \geq k) > 0$

Russian Roulette Estimator

If we repeatedly apply the same procedure *infinitely many times*, we obtain an unbiased estimator of the infinite series.

$$\sum_{k=1}^{\infty} \Delta_k = \mathbb{E}_{n \sim p(N)} \left[\sum_{k=1}^{n} \frac{\Delta_k}{\mathbb{P}(N \geq k)} \right]$$

Directly sample the first successful coin toss.

k-th term is weighted by prob. of seeing >= k tosses.

Computed in **finite** time with **prob. 1**!!

$p(N)$ is a geometric distribution in the case of coin toss.
It can be any distribution as long as $p(N \geq k) > 0$

Stochastically Unbiased Marginalization Objective

$$L_\infty = L_1 + (L_2 - L - 1) + (L_3 - L_2) + \cdots$$
$$= L_1 + \sum_{K=1}^{\infty} L_{K+1} - L_K$$

$$IW_K := \log \frac{1}{K} \sum_{k=1}^{K} \frac{p(x, z_j)}{q(z_j|x)} \qquad L_K = \mathbb{E}[IW_K]$$

$$\Delta_K := IW_{K+1} - IW_K$$

$$L_\infty = \mathbb{E}[IW_1 + \sum_{K=1}^{\infty} \Delta_K]$$

$$SUMO := IW_1 + \sum_{K=1}^{n} \frac{\Delta_K}{\mathbb{P}(N \geq K)} \quad \text{where } n \sim p(N)$$

Stochastically Unbiased Marginalization Objective

$$L_\infty = L_1 + (L_2 - L - 1) + (L_3 - L_2) + \cdots$$

$$= L_1 + \sum_{K=1}^{\infty} L_{K+1} - L_K$$

$$IW_K := \log \frac{1}{K} \sum_{k=1}^{K} \frac{p(x, z_j)}{q(z_j|x)} \qquad L_K = \mathbb{E}[IW_K]$$

$$\Delta_K := IW_{K+1} - IW_K$$

$$L_\infty = \mathbb{E}[IW_1 + \sum_{K=1}^{\infty} \Delta_K]$$

$$SUMO := IW_1 + \sum_{K=1}^{n} \frac{\Delta_K}{\mathbb{P}(N \geq K)} \quad \text{where } n \sim p(N)$$

$$\mathbb{E}_{n, z_1, z_2, \ldots}[SUMO] = L_\infty = \log p(x)$$

Table 1: Test negative log-likelihood of the trained model, estimated using IWAE(k=5000). For SUMO, k refers to the expected number of computed terms.

	MNIST			OMNIGLOT		
Training Objective	k=5	k=15	k=50	k=5	k=15	k=50
ELBO (Burda et al., 2016)	86.47	—	86.35	107.62	—	107.80
IWAE (Burda et al., 2016)	85.54	—	84.78	106.12	—	104.67
ELBO (Our impl.)	85.97±0.01	85.99±0.05	85.88±0.07	106.79±0.08	106.98±0.19	106.84±0.13
IWAE (Our impl.)	85.28±0.01	84.89±0.03	84.50±0.02	104.96±0.04	104.53±0.05	**103.99±0.12**
JVI (Our impl.)	—	—	84.75±0.03	—	—	104.08±0.11
SUMO	**85.09±0.01**	**84.71±0.02**	**84.40±0.03**	**104.85±0.04**	**104.29±0.12**	**103.79±0.14**