Deep Generative Models

Rianne van den Berg

Principal Research Manager @AI4Science - Microsoft Research Amsterdam

AI4Science at Microsoft

[AI4Science @Microsoft Research]

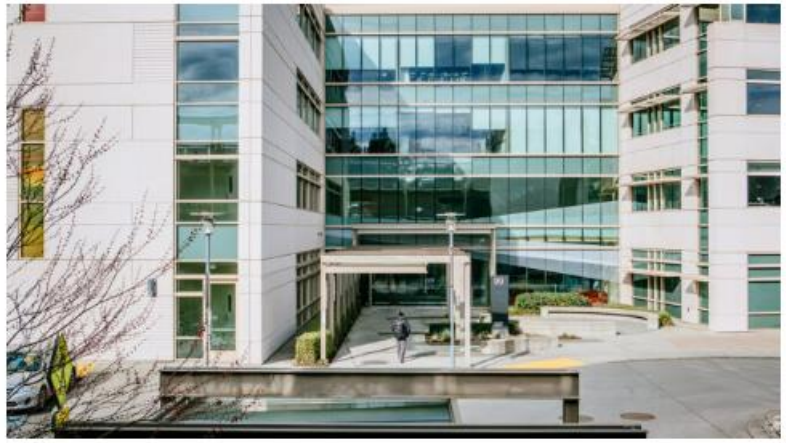
Our locations

Amsterdam, Netherlands

Beijing, China

Berlin, Germany

Cambridge, UK



Redmond, USA

Shanghai, China

Today’s lecture team

Chin-Wei Huang
Senior Researcher
AI4Science, Microsoft Research
Amsterdam

Victor Garcia Satorras
Senior Researcher
AI4Science, Microsoft Research
Amsterdam

Morning program: lecture

Probabilistic generative modeling:

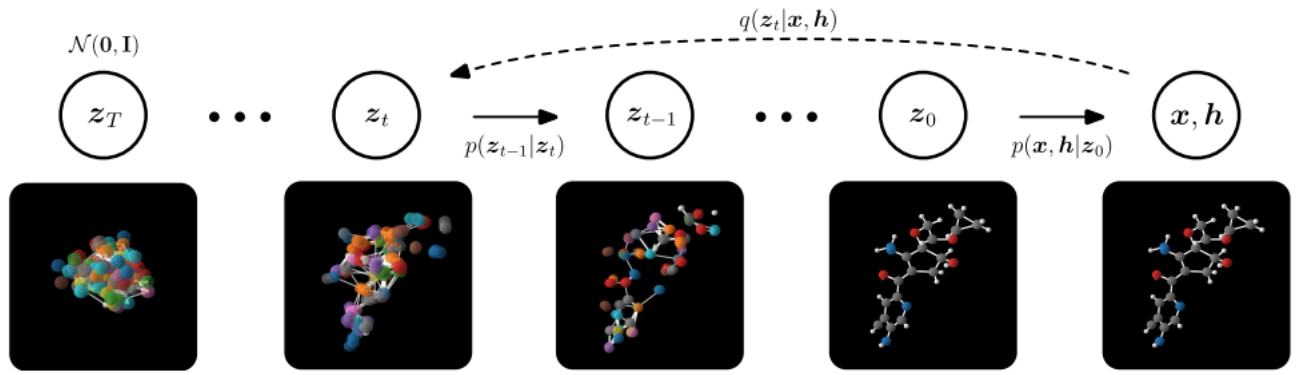
1. Variational auto-encoders
2. Normalizing flows
3. Denoising diffusion models

Afternoon program

Workshop on Diffusion Models for Molecule Generation

Equivariant Diffusion for Molecule Generation in 3D

Emiel Hoogeboom [*1] Victor Garcia Satorras [*1] Clément Vignac [*2] Max Welling [1]



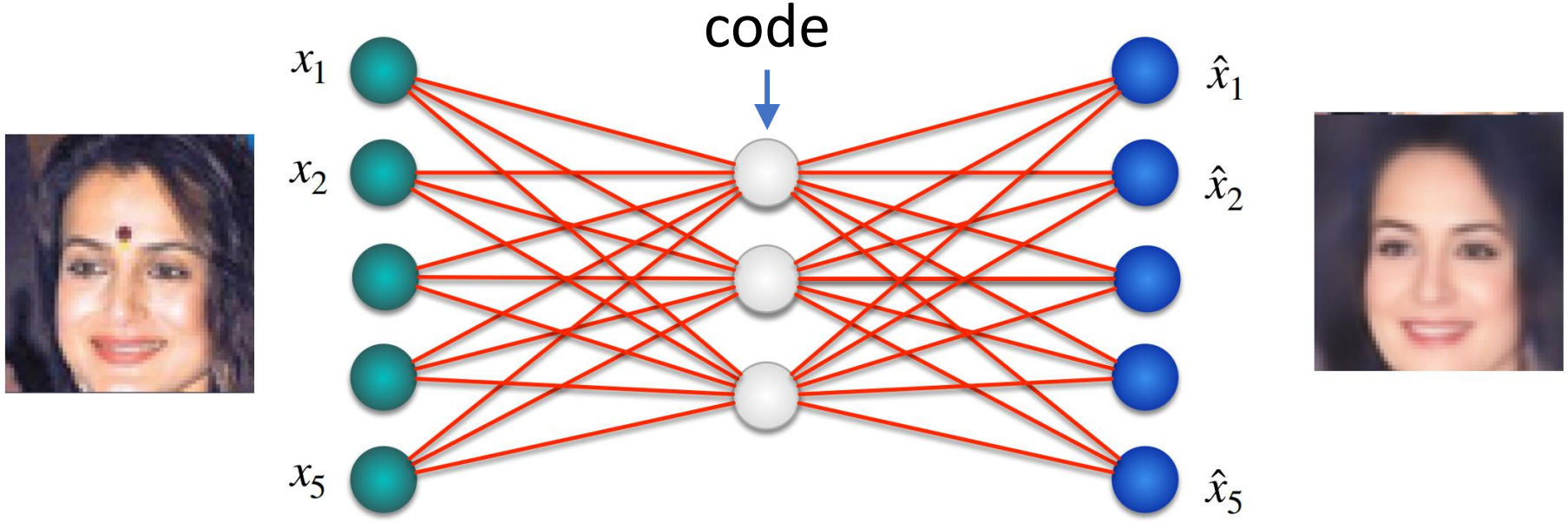
Morning program: lecture

Probabilistic generative modeling:

1. **Variational auto-encoders**
2. Normalizing flows
3. Denoising diffusion models

Auto-encoders

Copy input to output while going through a bottleneck



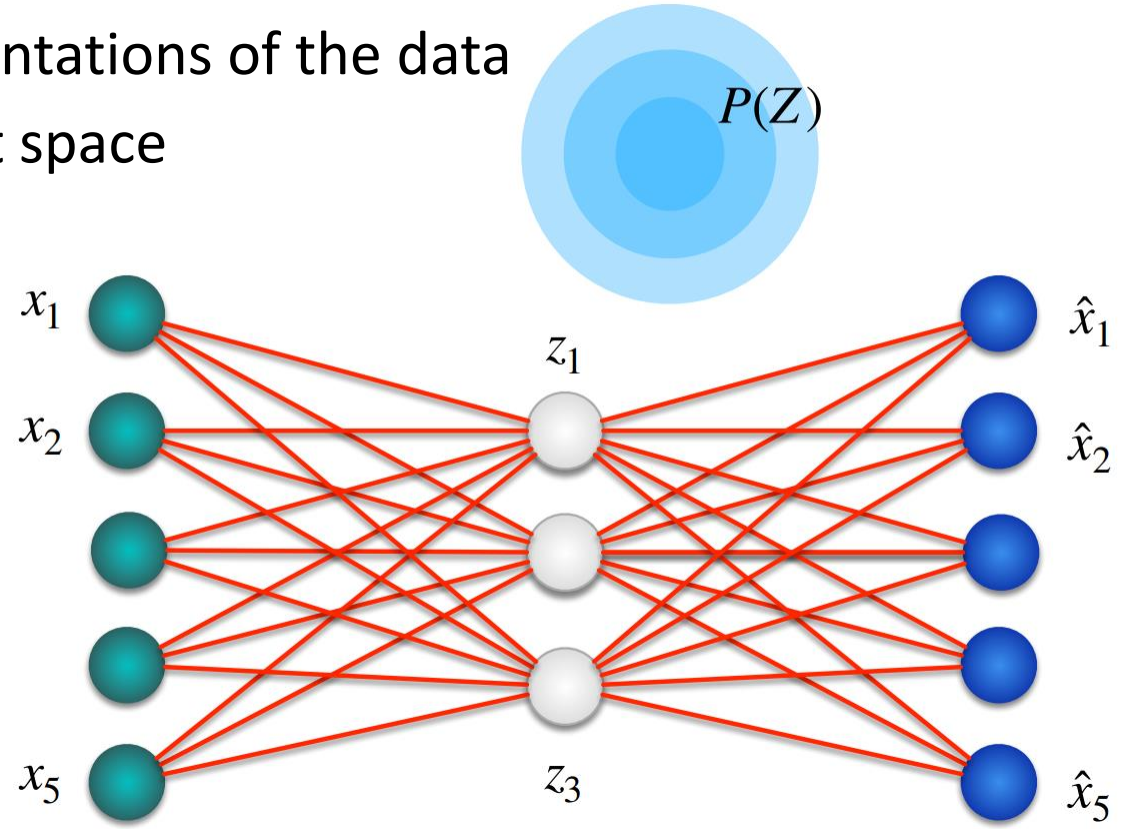
The code in the bottleneck should be a compressed representation of data x

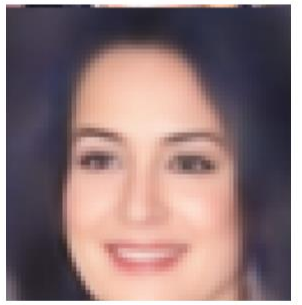
Face image source: Tolstikhin et al., ICLR 2018

Auto-encoders as generative models

Generative auto-encoders:

- Learn compressed representations of the data
- prior distribution on latent space



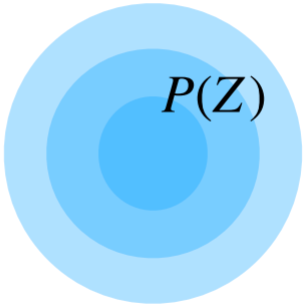


Face image source: Tolstikhin et al., ICLR 2018

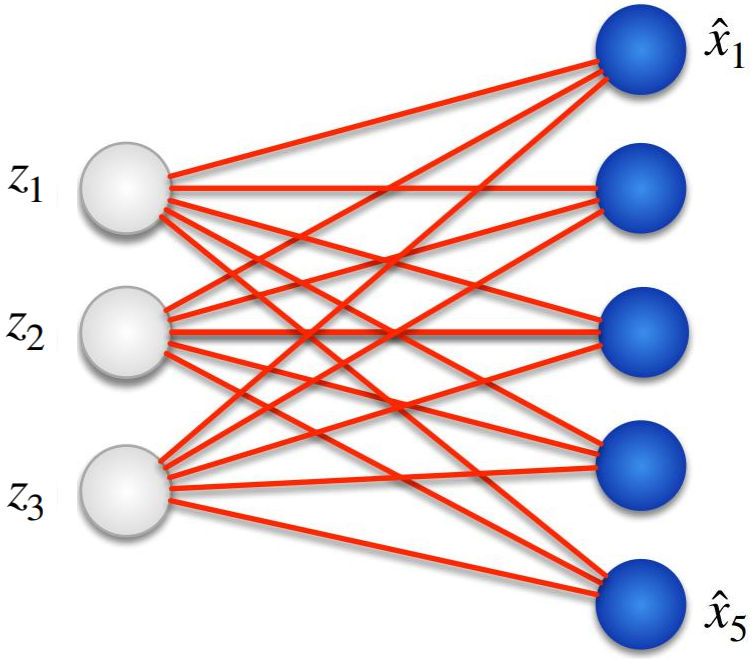
Auto-encoders as generative models

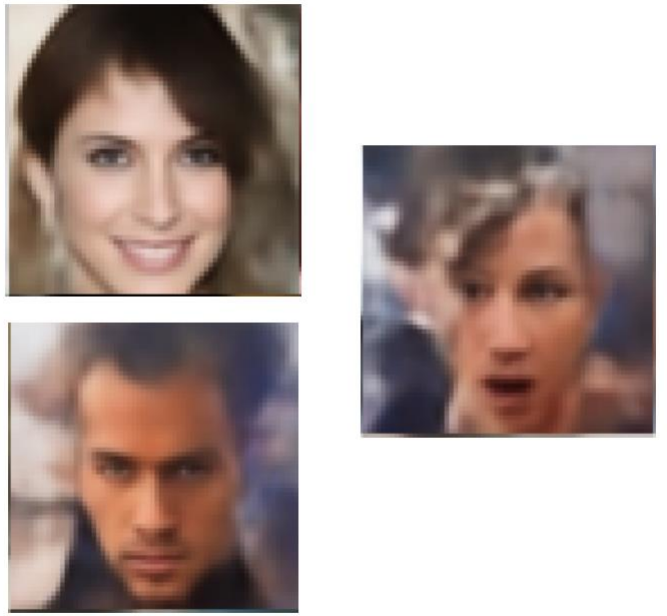
Generative auto-encoders:

- Learn compressed representations of the data
- prior distribution on latent space
- Can generate new data



sample



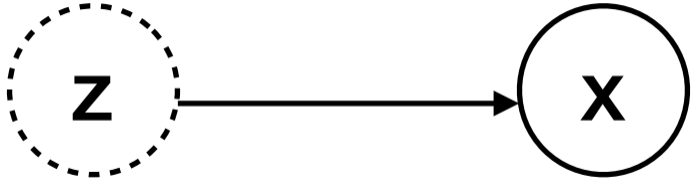


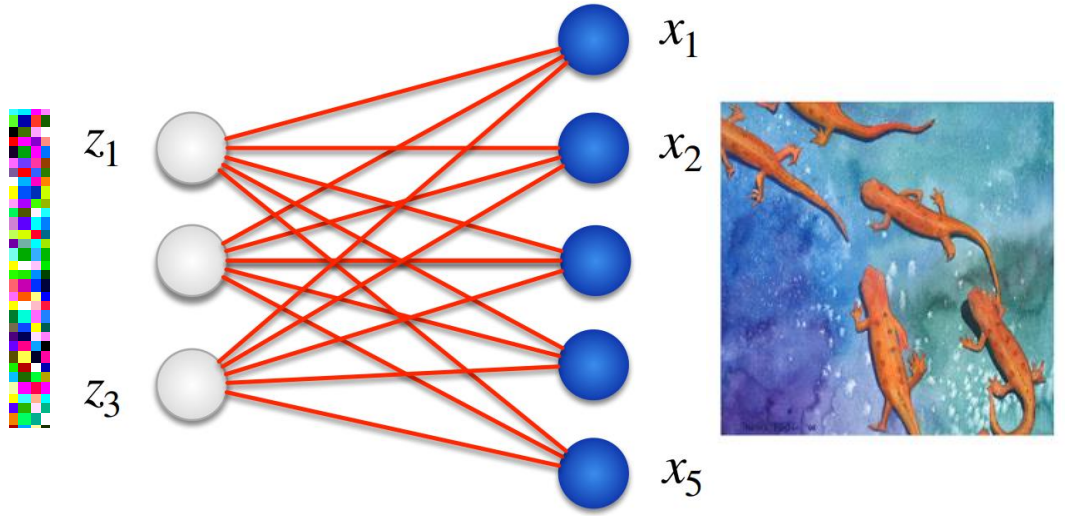
Face image source: Tolstikhin et al., ICLR 2018

Variational auto-encoders

Kingma & Welling 2014; Rezende, Mohamed, Wierstra, 2014

Joint model $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$





Desired objective: $\log p_\theta(x) = \log \mathbb{E}_{z\sim p(z)}[p_\theta(x|z)]$

Surrogate objective:

$$\log p_\theta(x) \geq \mathbb{E}_{q(z|x)}[\log p_\theta(x|z)] - KL[q(z|x)||p_\theta(z)] = ELBO\ (\theta)$$

$q(z|x)$: encoding distribution

Amortized variational inference

Surrogate objective $\arg \max_{q \in Q, \theta} \mathbb{E}_{q(z|x)}[\log p_\theta(x|z)] - KL[q(z|x)||p_\theta(z)]$

Q: family of approximate posteriors.

Variational inference: optimize parameters of q(z|x) for each x separately

Amortized variational inference: model parameters of q(z|x) with a neural network $\mathrm{nn}_\theta(x)$

$$q(z|x) = q_\phi(z|x)$$

$$\arg \max_{\phi, \theta} \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL[q_\phi(z|x)||p_\theta(z)]$$

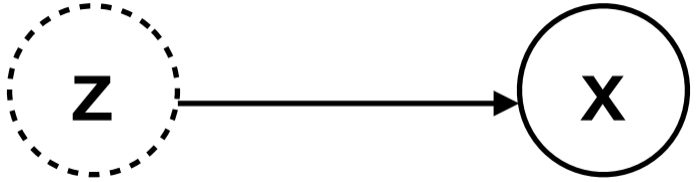
Frequently used approximate posterior: $q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \mathrm{diag}\left(\sigma_\phi^2(x)\right))$

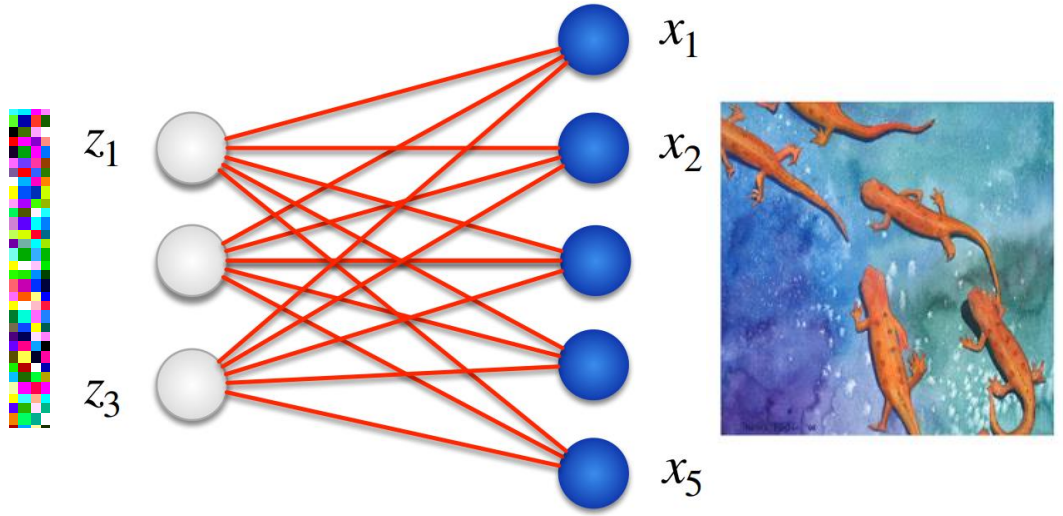
Frequently used prior: $p_\theta(z) = p(z) = \mathcal{N}(z|0, I)$

Variational auto-encoders

Kingma & Welling 2014; Rezende, Mohamed, Wierstra, 2014

Joint model $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$

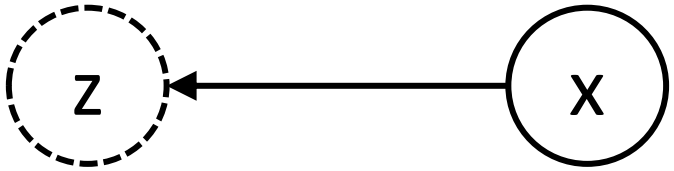


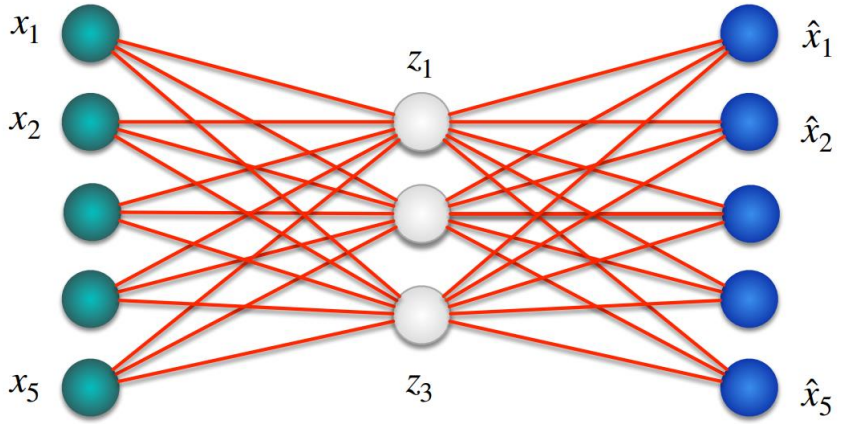


Surrogate objective:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL[q_\phi(z|x)||p_\theta(z)] = ELBO\,(\theta, \phi)$$

$q(z|x)$: encoding distribution





VAE objective: two for one

An introduction to variational auto-encoders, Kingma & Welling, 2019

$$\arg \max_{q \in Q, \theta} \mathbb{E}_{q(z|x)}[\log p_\theta(x|z)] - KL[q_\phi(z|x)||p_\theta(z)]$$

$$\log p_\theta(x) = KL(q_\phi(z|x)||p_\theta(z|x)) + ELBO(\theta, \phi)$$

Maximizing the ELBO w.r.t. parameters θ and ϕ leads to optimization of two quantities:

1. $\log p_\theta(x)$ does not depend on ϕ so maximizing $ELBO(\theta, \phi)$ wrt ϕ makes $q_\phi(z|x)$ a better approximation to the intractable $p_\theta(z|x)$
2. We are approximately maximizing $\log p_\theta(x)$ → improves generative model.

Optimizing the parameters of the generative model

An introduction to variational auto-encoders, Kingma & Welling, 2019

Unbiased gradients of the ELBO w.r.t. the generative model parameters $\boldsymbol{\theta}$ are simple to obtain:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right] \tag{2.14}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\boldsymbol{\theta}} (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \right] \tag{2.15}$$

$$\simeq \nabla_{\boldsymbol{\theta}} (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \tag{2.16}$$

$$= \nabla_{\boldsymbol{\theta}} (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})) \tag{2.17}$$

The last line (eq. (2.17)) is a simple Monte Carlo estimator of the second line (eq. (2.15)), where $\mathbf{z}$ in the last two lines (eq. (2.16) and eq. (2.17)) is a random sample from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$.

Optimizing the parameters of the encoder

An introduction to variational auto-encoders, Kingma & Welling, 2019

Unbiased gradients w.r.t. the *variational* parameters $\boldsymbol{\phi}$ are more difficult to obtain, since the ELBO's expectation is taken w.r.t. the distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, which is a function of $\boldsymbol{\phi}$. I.e., in general:

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right] \tag{2.18}$$

$$\neq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\boldsymbol{\phi}} (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \right] \tag{2.19}$$

Reparameterization: separating randomness & parameters

Kingma & Welling 2014; Rezende, Mohamed, Wierstra, 2014

Rewrite $z \sim q_\phi(z|x)$ as a transformation of another random variable ϵ that does not depend ϕ and x:

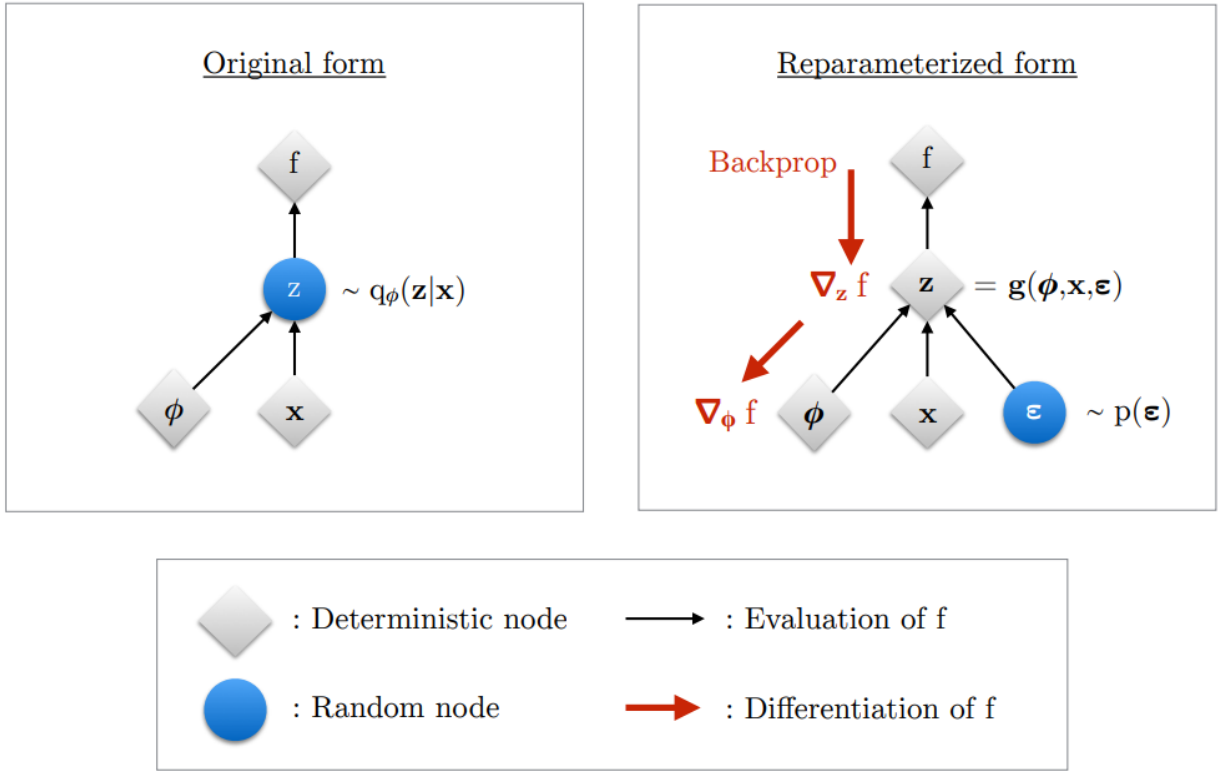
$$z = g(\epsilon, \phi, x) \qquad \epsilon \sim p(\epsilon)$$

Then for any $f(z)$ we can do the following

$$\begin{aligned} \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] &= \nabla_\phi \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(\mathbf{z})] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_\phi f(\mathbf{z})] \\ &\simeq \nabla_\phi f(\mathbf{z}) \end{aligned}$$

An introduction to variational auto-encoders, Kingma & Welling, 2019

Reparameterization trick



An introduction to variational auto-encoders, Kingma & Welling, 2019

Reparameterizing the ELBO

$$ELBO(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z|x)\right]$$
$$= \mathbb{E}_{p(\epsilon)}[\log p_\theta(x, z(\epsilon, \phi, x)) - \log q_\phi(z(\epsilon, \phi, x)|x)]$$

$$z = g(\epsilon, \phi, x) \qquad \epsilon \sim p(\epsilon)$$

Gradients wrt encoder parameters ϕ can be computed with automatic backprop

$$\nabla_\phi \mathrm{ELBO}(\theta, \phi) = \mathbb{E}_{p(\epsilon)}[\nabla_\phi \log p_\theta(x, z(\epsilon, \phi, x)) - \nabla_\phi \log q_\phi(z(\epsilon, \phi, x)|x)]$$

Reparameterization for factorized Gaussian distributions

$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_i q_\phi(z_i|\mathbf{x}) = \prod_i \mathcal{N}(z_i; \mu_i, \sigma_i^2)$$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$$

Reparameterization:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \mathrm{EncoderNeuralNet}_\phi(\mathbf{x})$$

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$

Reparameterization for full-covariance Gaussian distributions

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Reparameterization:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$
$$\mathbf{z} = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}$$

Cholesky decomposition of the covariance matrix: $\Sigma = LL^T$

(L is lower triangular and has nonzero diagonal entries)

Results from Kingma & Welling 2014

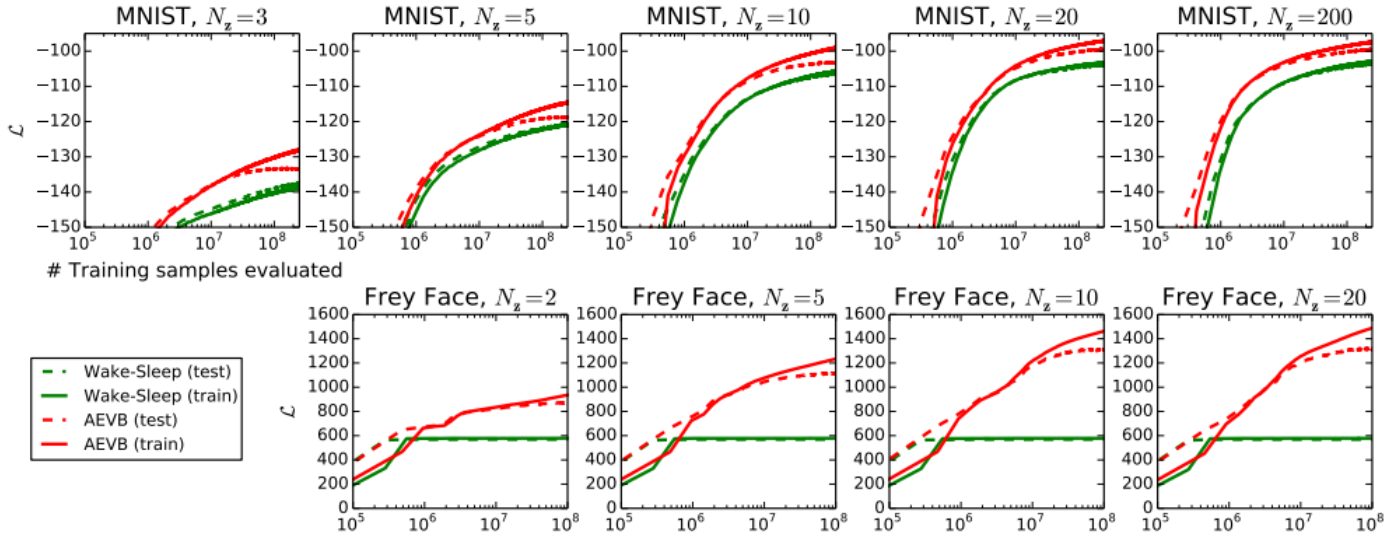


Figure 2: Comparison of our AEVB method to the wake-sleep algorithm, in terms of optimizing the lower bound, for different dimensionality of latent space ($N_{\mathbf{z}}$). Our method converged considerably faster and reached a better solution in all experiments. Interestingly enough, more latent variables does not result in more overfitting, which is explained by the regularizing effect of the lower bound. Vertical axis: the estimated average variational lower bound per datapoint. The estimator variance was small (< 1) and omitted. Horizontal axis: amount of training points evaluated. Computation took around 20-40 minutes per million training samples with a Intel Xeon CPU running at an effective 40 GFLOPS.

Extra slides VAEs

Approximate variational inference

Jordan et al, 1998, Jaakkola and Jordan 2000; Jaakola 2001; Wainwright and Jordan 2008

Variational inference: a review for statisticians, Blei et al. 2018

"The goal of variational inference is to approximate a conditional density of latent variables given observed variables."

Observed data: $X = \{x_1, \ldots, x_N\}$

We assume that the data was produced by the generative model

$$p(z, x) = p(z)p(x|z)$$

Latent variables / model parameters

Approximate variational inference

Example: Bayesian mixture of K Gaussians

K cluster means: $\mu_k \sim p(\mu_k) = \mathcal{N}(0, \sigma^2) \quad k = 1, \dots K$

N cluster assignments: $\boldsymbol{z}_i \sim p(\boldsymbol{z}_i) = \text{Categorical}\left(\frac{1}{K}, \dots, \frac{1}{K}\right) \quad i = 1, \dots, N$

N datapoints: $x_i \sim p(x_i|\boldsymbol{z}_i, \boldsymbol{\mu}) = \mathcal{N}\left(x_i \middle| z_i^T \mu, 1\right) \quad i = 1, \dots, N$

$$p(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu}) = p(\boldsymbol{\mu}) \prod_{i=1}^{N} p(\boldsymbol{z}_i) p(x_i|\boldsymbol{z}_i, \boldsymbol{\mu})$$

Goal: infer $p(\mu, Z|X) = p(X, Z, \mu)/p(X)$

Evidence: intractable

Approximate variational inference

Pick a family of distributions Q, and find the best candidate $q(z) \in Q$ such that

$$q^*(z) = \arg\min_{q(z)\in Q} KL[q(z)||p(z|x)]$$

$$KL[q(z)||p(z|x)] = \mathbb{E}[\log q(z)] - \mathbb{E}[\log p(z|x)]$$

But this KL is not tractable because we don't know $p(z|x)$ …

$$KL[q(z)||p(z|x)] = \mathbb{E}[\log q(z)] - \mathbb{E}[\log p(z,x)] + \log p(x)$$

Instead, we can maximize an alternative objective:

$$ELBO(q) = \mathbb{E}[\log p(z,x)] - \mathbb{E}[\log q(z)]$$

Approximate variational inference

$$ELBO(q) = \mathbb{E}[\log p(z, x)] - \mathbb{E}[\log q(z)]$$

ELBO: Evidence lower bound

$\log p(x) = KL[q(z)||p(z|x)] + ELBO(q)$ → ELBO $\leq \log p(x)$

Optimizing the ELBO with respect to q(z) ensures that

1. q(z) approximates the unknown p(z|x)
2. ELBO(q) becomes a tighter lower bound to log p(x)

Mean-field variational inference

Mean-field variational family: all latent variables are independent and each variable has a distinct factor describing its distribution

$$q(z) = \prod_{j=1}^{m} q_j(z_j)$$

K cluster means: $\mu_k \sim p(\mu_k) = \mathcal{N}(0, \sigma^2) \quad k = 1, \dots K$

N cluster assignments: $\boldsymbol{z}_i \sim p(\boldsymbol{z}_i) = \text{Categorical}\left(\frac{1}{K}, \dots, \frac{1}{K}\right) \quad i = 1, \dots, N$

N datapoints: $x_i \sim p(x_i | \boldsymbol{z}_i, \boldsymbol{\mu}) = \mathcal{N}\left(x_i \middle| z_i^T \mu, 1\right) \quad i = 1, \dots, N$

$p(\mu, Z|X) = ?$

$$q(\mu, Z) = \prod_{k=1}^{K} q\left(\mu_k \middle| m_k, s_k^2\right) \prod_{i=1}^{N} q(z_i | \phi_i)$$

KL and reverse KL: zero avoiding and zero forcing

Figure 10.2 Comparison of the two alternative forms for the Kullback-Leibler divergence. The green contours corresponding to 1, 2, and 3 standard deviations for a correlated Gaussian distribution $p(\mathbf{z})$ over two variables z_1 and z_2, and the red contours represent the corresponding levels for an approximating distribution $q(\mathbf{z})$ over the same variables given by the product of two independent univariate Gaussian distributions whose parameters are obtained by minimization of (a) the Kullback-Leibler divergence $\mathrm{KL}(q\|p)$, and (b) the reverse Kullback-Leibler divergence $\mathrm{KL}(p\|q)$.

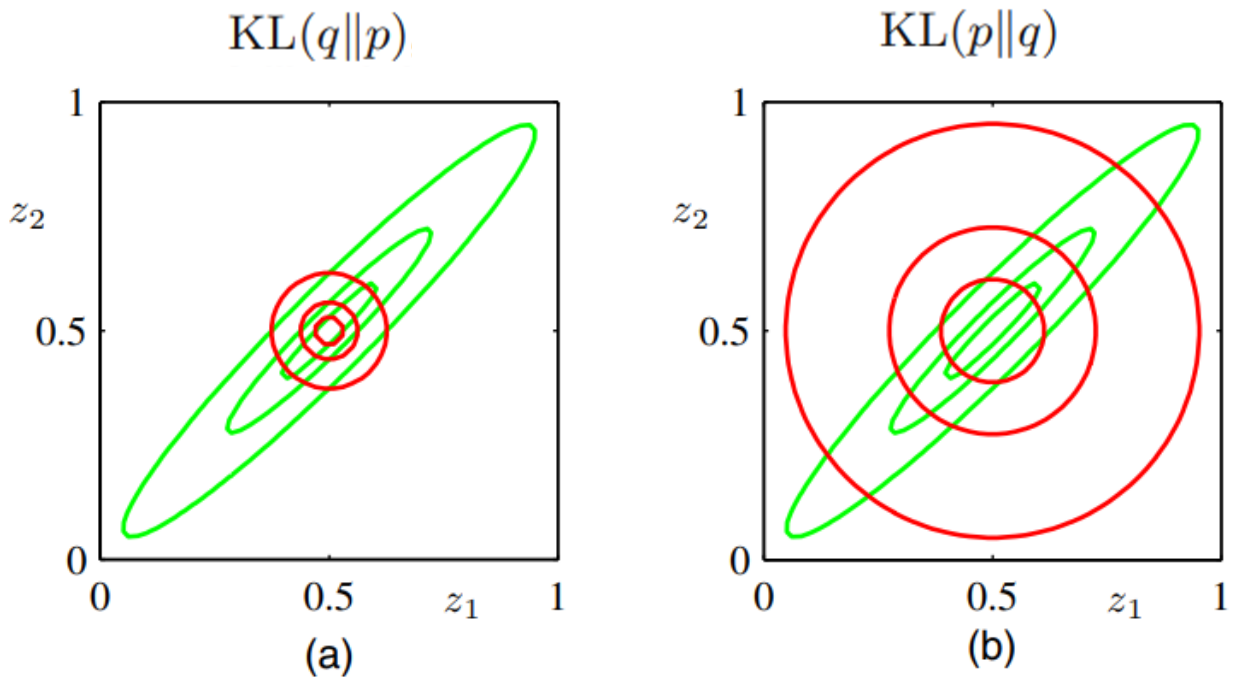


Figure source: PRML, Bishop, 2006

Mode seeking versus mode covering

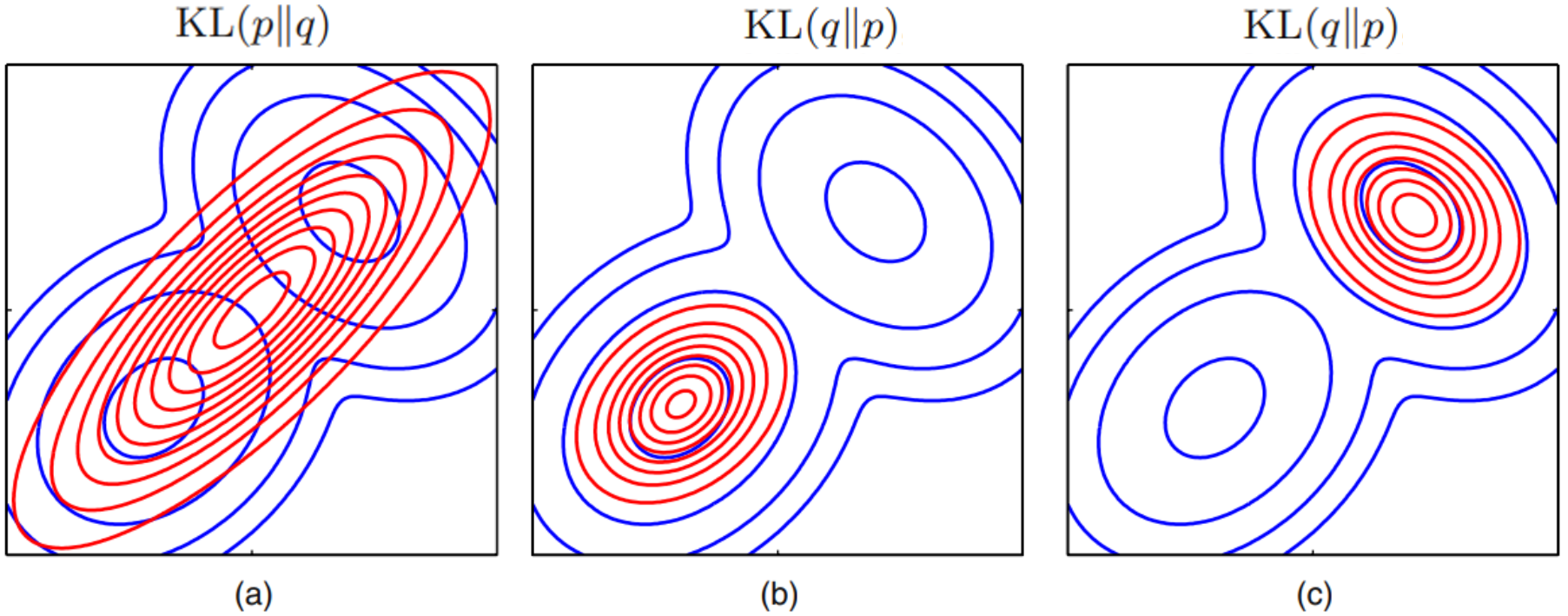


Figure 10.3 Another comparison of the two alternative forms for the Kullback-Leibler divergence. (a) The blue contours show a bimodal distribution $p(\mathbf{Z})$ given by a mixture of two Gaussians, and the red contours correspond to the single Gaussian distribution $q(\mathbf{Z})$ that best approximates $p(\mathbf{Z})$ in the sense of minimizing the Kullback-Leibler divergence $\mathrm{KL}(p\|q)$. (b) As in (a) but now the red contours correspond to a Gaussian distribution $q(\mathbf{Z})$ found by numerical minimization of the Kullback-Leibler divergence $\mathrm{KL}(q\|p)$. (c) As in (b) but showing a different local minimum of the Kullback-Leibler divergence.

Figure source: PRML, Bishop, 2006

α-divergences

Black-box α-Divergence minimization, Hernandez-Lobato et al, 2015

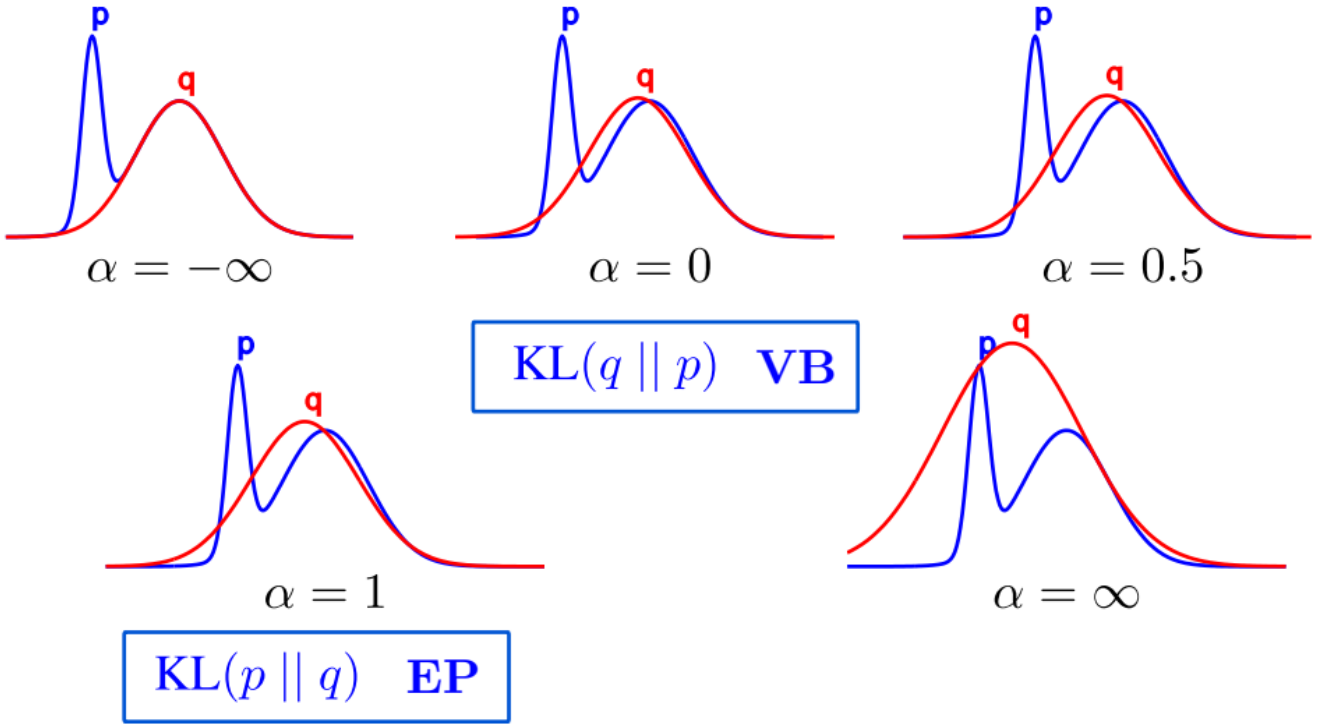
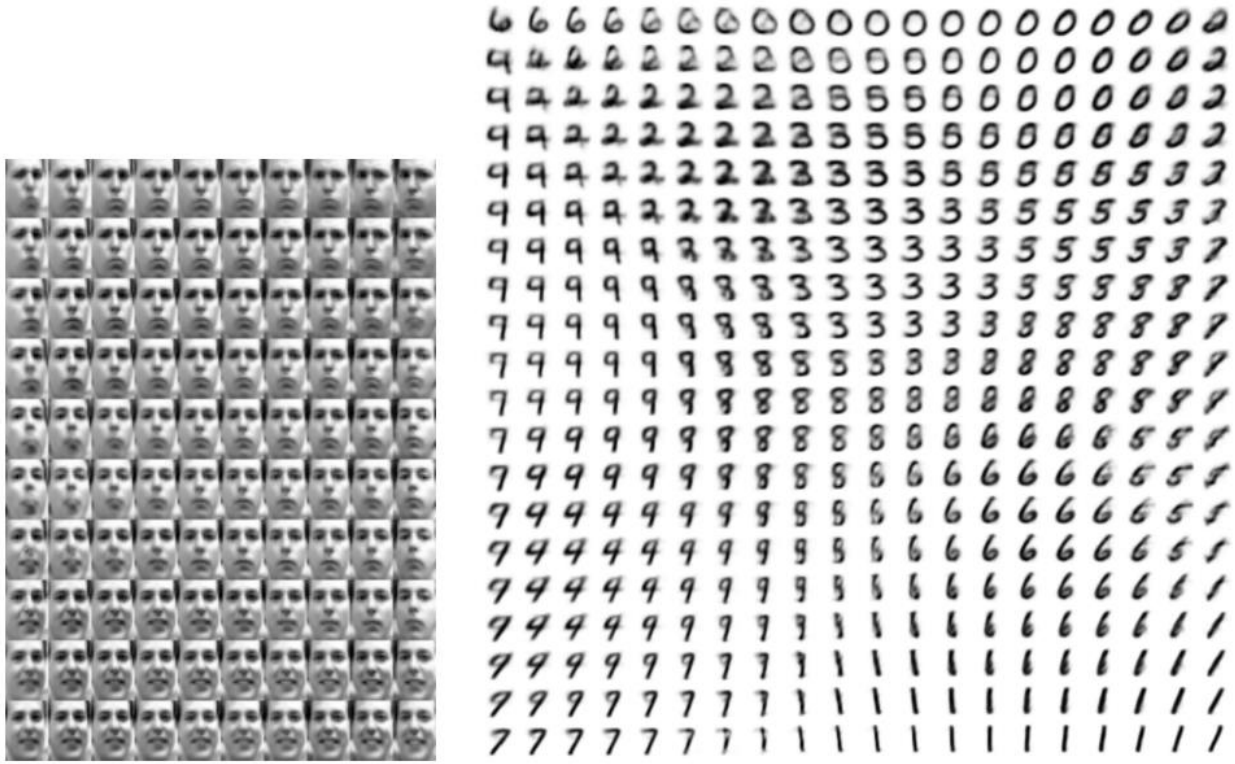


Figure 2. An illustration of approximating distributions by α-divergence minimization. Here p and q shown in the graphs are unnormalized probability densities. Reproduced from Minka (2005). Best viewed in color.

Results from Kingma & Welling 2014



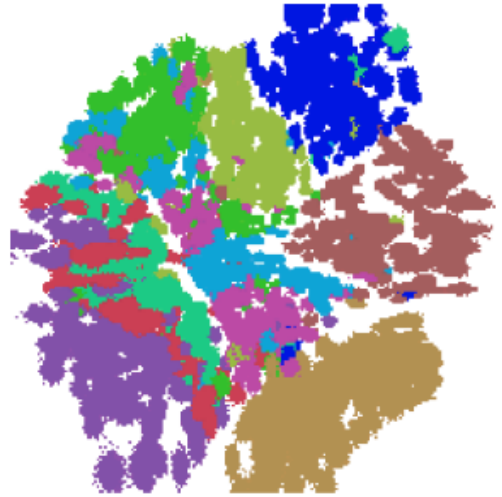
(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Results from Rezende et al. 2014

(a) Left: Training data. Middle: Sampled pixel probabilities. Right: Model samples



(b) 2D embedding.

Figure 3. Performance on the MNIST dataset. For the visualisation, each colour corresponds to one of the digit classes.

Marginal likelihood estimation: importance sampling

Rezende et al. 2014, Burda et al. 2016

$$ELBO(\theta,\phi) = \mathbb{E}_{q_\phi(z|x)}[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}] \approx \frac{1}{K}\sum_{k=1}^{K} \log \frac{p_\theta(x,z^{(k)})}{q_\phi(z^{(k)}|x)}$$

Tighter bound is obtained by importance sampling:

$$ELBO_{IW}^{K}(\theta,\phi) = \mathbb{E}_{z^{(1)},\ldots,z^{(K)}\sim q_\phi(z|x)}[\log(\frac{1}{K}\sum_{k=1}^{K} \frac{p_\theta\left(x,z^{(k)}\right)}{q_\phi\left(z^{(k)}\middle|x\right)})] \leq \log p_\theta(x)$$

A very incomplete selection of advances in VAEs: Ladder VAEs

Ladder Variational autoencoders, Sonderby et al. 2016

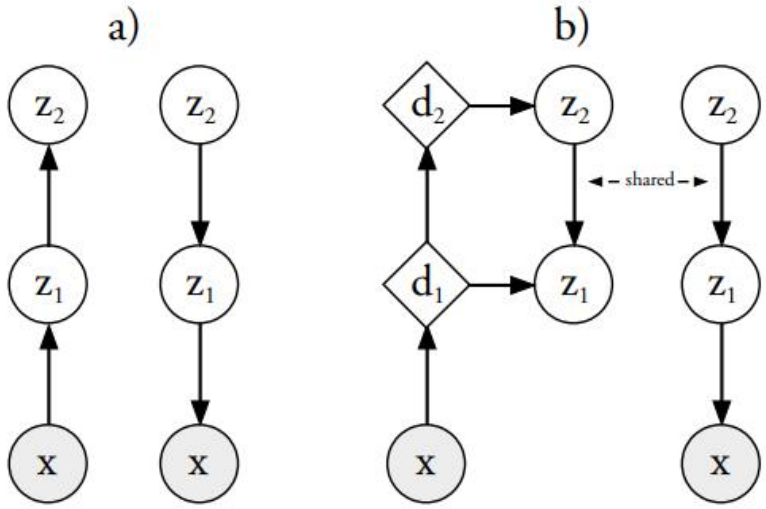


Figure 1: Inference (or encoder/recognition) and generative (or decoder) models for a) VAE and b) LVAE. Circles are stochastic variables and diamonds are deterministic variables.

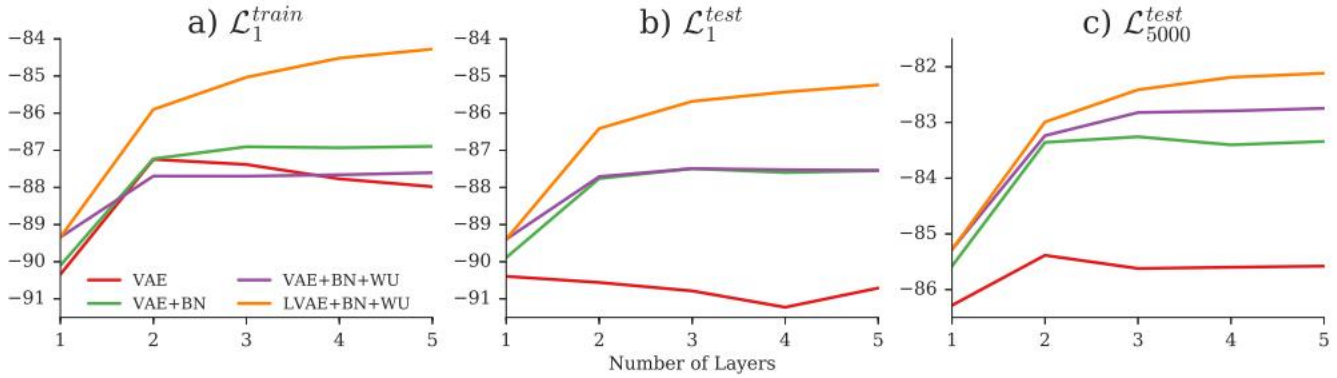


Figure 3: MNIST log-likelihood values for VAEs and the LVAE model with different number of latent layers, Batch normalization (*BN*) and Warm-up (*WU*). a) Train log-likelihood, b) test log-likelihood and c) test log-likelihood with 5000 importance samples.

Follow up work: Biva: a very deep hierarchy of latent variables for generative modeling, Maaloe et al. 2019

A very incomplete selection of advances in VAEs: NVAE

NVAE: a deep hierarchical variational autoencoder, 2020

In summary, we make the following contributions: i) We propose a novel deep hierarchical VAE, called NVAE, with depthwise convolutions in its generative model. ii) We propose a new residual parameterization of the approximate posteriors. iii) We stabilize training deep VAEs with spectral regularization. iv) We provide practical solutions to reduce the memory burden of VAEs. v) We show that deep hierarchical VAEs can obtain state-of-the-art results on several image datasets, and can produce high-quality samples even when trained with the original VAE objective. To the best of our knowledge, NVAE is the first successful application of VAEs to images as large as 256×256 pixels.

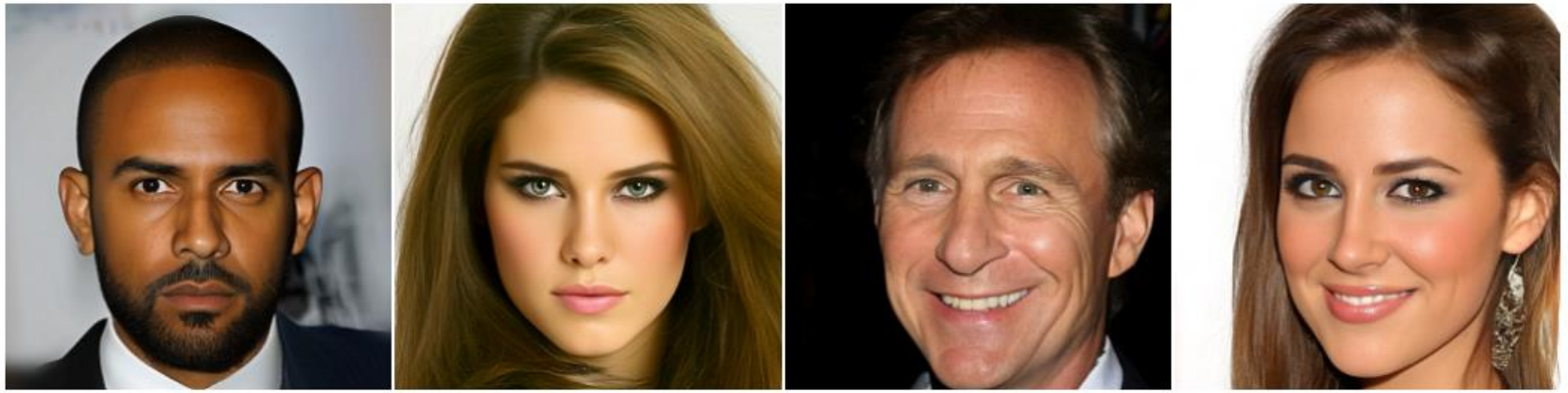


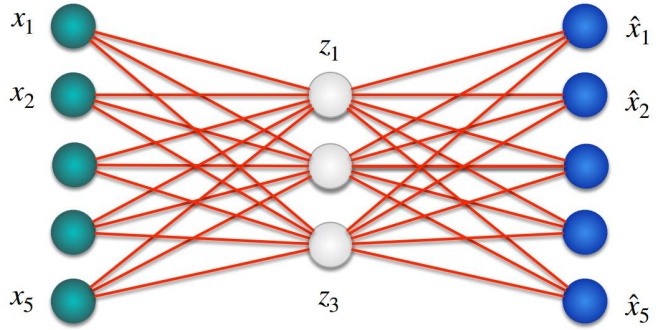
Figure 1: 256×256-pixel samples generated by NVAE, trained on CelebA HQ [28].

Morning program: lecture

Probabilistic generative modeling:

1. Variational auto-encoders
2. **Normalizing flows**
3. Denoising diffusion models

Variational auto-encoders



Surrogate objective:

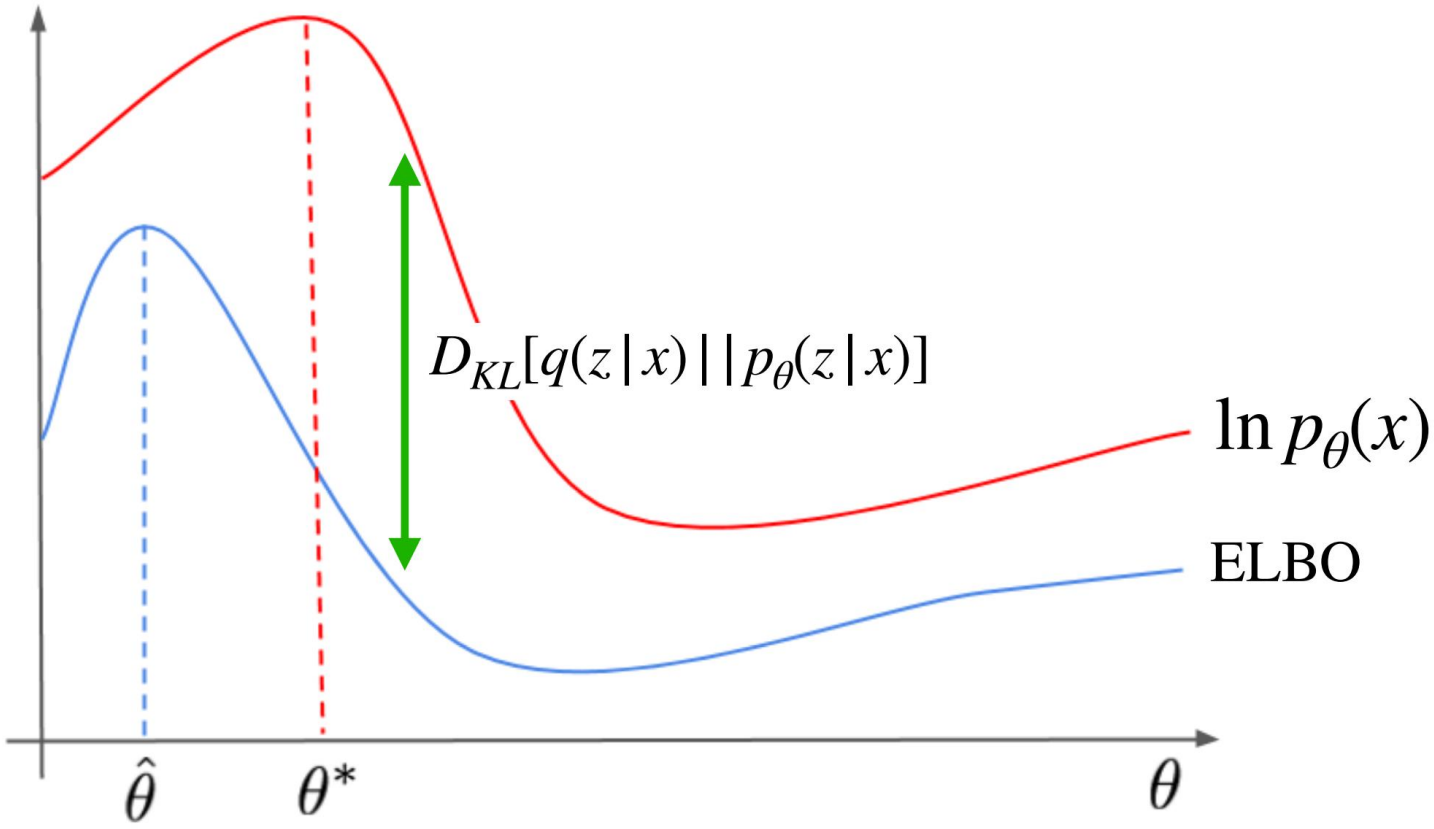
$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL[q_\phi(z|x)||p_\theta(z)] = ELBO\,(\theta, \phi)$$

Equivalent to:

$$\text{ELBO}(\theta, \phi) = \log p_\theta(x) - KL(q_\phi(z|x)||p_\theta(z|x))$$

- $q_\phi(z|x)$ tries to approximate $p_\theta(z|x)$
- Bad approximations lead to a large gap!

Approximate posteriors



Looser lower bounds do not share the same local maxima as the log likelihood

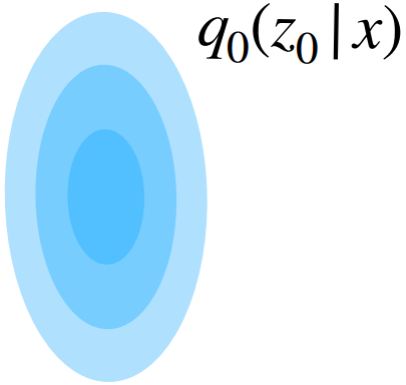
Figure source: Sylvester normalizing flows for variational inference, R van den Berg et al. 2018

Variational inference with normalizing flows

Rezende & Mohamed, ICML 2015

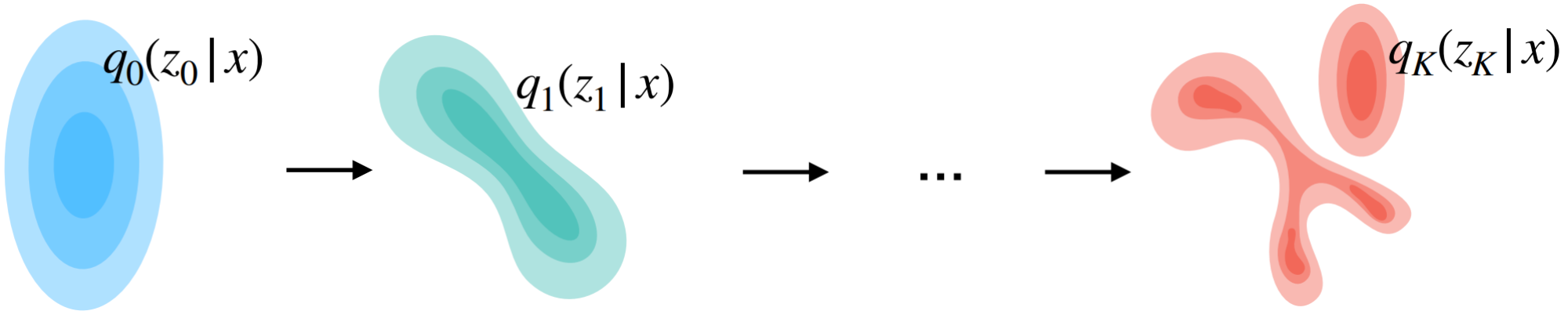
Sample from simple distribution $z_0 \in \mathbb{R}^D$

$$z_0 \sim q_0(z \mid x) = \mathcal{N}(z \mid \mu(x), \mathrm{diag}(\sigma^2(x)))$$



Apply a sequence of invertible transformations $f_k : \mathbb{R}^D \mapsto \mathbb{R}^D$

$$z_K = f_K \circ \ldots \circ f_2 \circ f_1(z_0) \qquad z_0 \to z_1 \to \ldots \to z_K$$

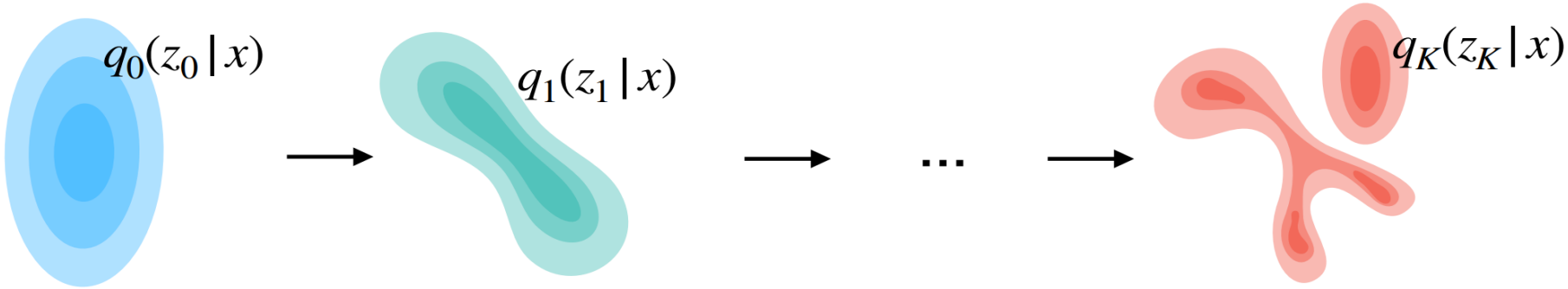


Variational inference with normalizing flows

Rezende & Mohamed, ICML 2015

Apply a sequence of invertible transformations $f_k : \mathbb{R}^D \mapsto \mathbb{R}^D$

$$z_K = f_K \circ \ldots \circ f_2 \circ f_1(z_0) \qquad z_0 \rightarrow z_1 \rightarrow \ldots \rightarrow z_K$$



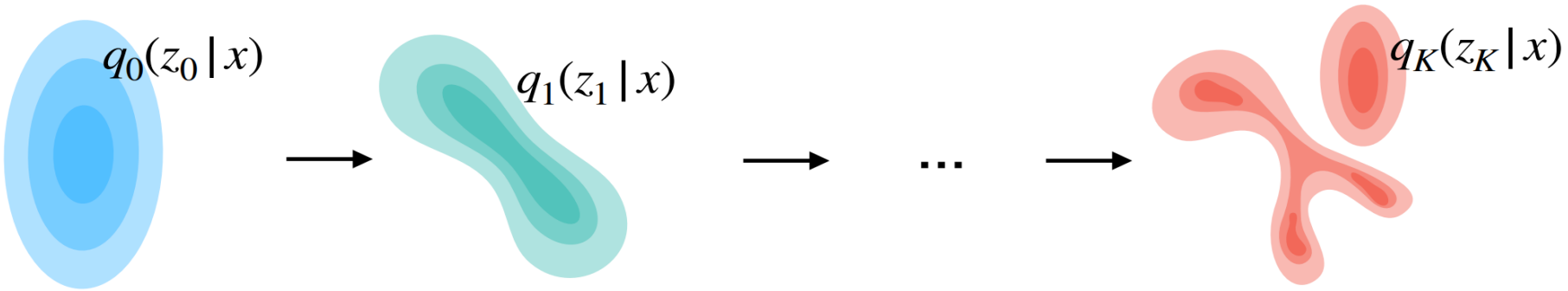
For each transformation $z_k = f_k(z_{k-1})$

$$q_k(z_k) = q_{k-1}(z_{k-1}) \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|^{-1} \longrightarrow \ln q_K(z_K) = \ln q_0(z_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|$$

Practical normalizing flows for variational inference

1. Flexible invertible transformations

Inflexible transformations lead to long sequences of flows for flexible posteriors



2. Easily computable Jacobian determinants:

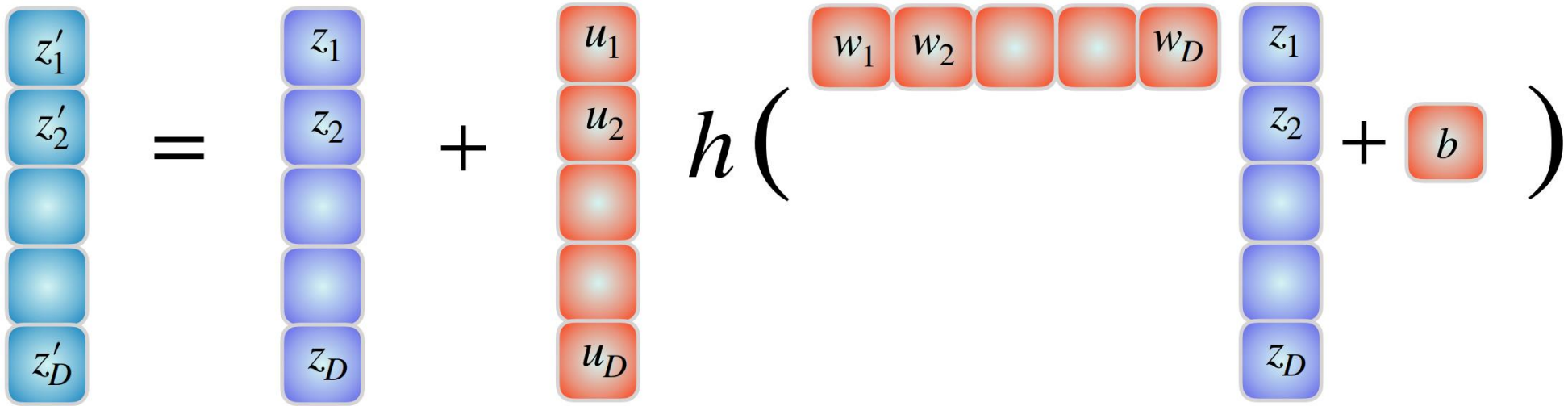
For a D x D matrix in general $O(D^3)$

Planar flows

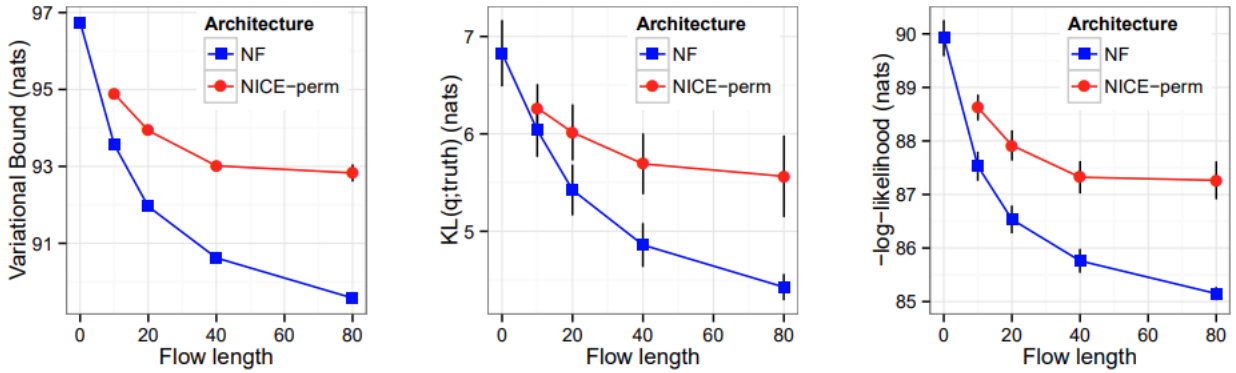
Rezende & Mohamed, ICML 2015

: learnable parameter

$$z' = z + u\, h(w^T z + b)$$



Results Rezende et al.



(a) Bound $\mathcal{F}(\mathbf{x})$ (b) $\mathbb{D}_{\mathrm{KL}}(q; p(z|x))$ (c) $-\ln p(\mathbf{x})$

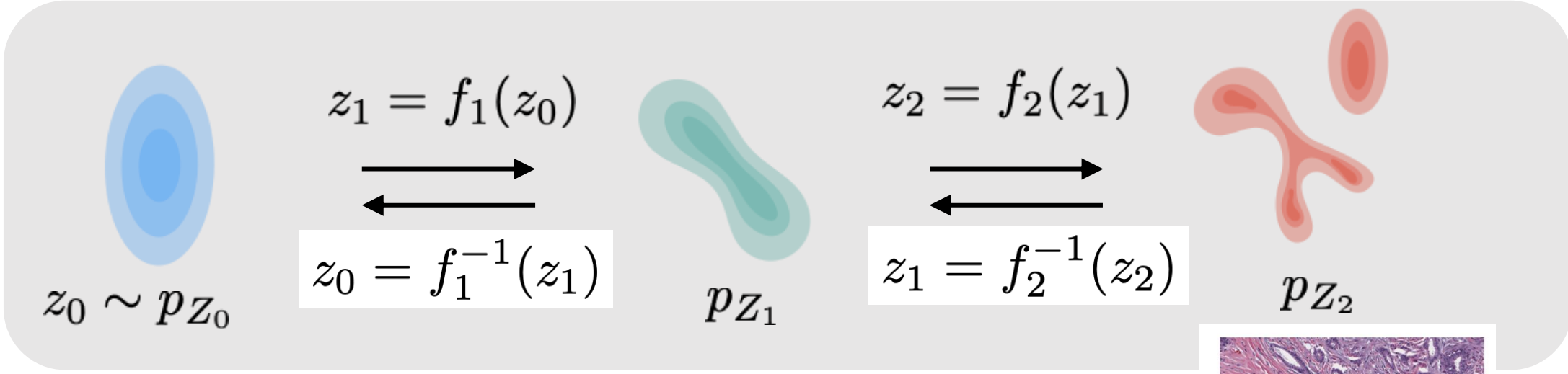
Figure 4. Effect of the flow-length on MNIST.

Table 2. Comparison of negative log-probabilities on the test set for the binarised MNIST data.

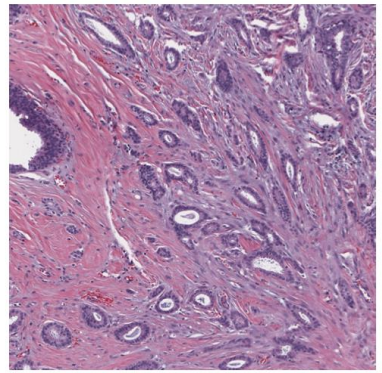
Model	$-\ln p(\mathbf{x})$
DLGM diagonal covariance	$\leq$ 89.9
DLGM+NF (k = 10)	$\leq$ 87.5
DLGM+NF (k = 20)	$\leq$ 86.5
DLGM+NF (k = 40)	$\leq$ 85.7
DLGM+NF (k = 80)	$\leq$ 85.1
DLGM+NICE (k = 10)	$\leq$ 88.6
DLGM+NICE (k = 20)	$\leq$ 87.9
DLGM+NICE (k = 40)	$\leq$ 87.3
DLGM+NICE (k = 80)	$\leq$ 87.2
Results below from (Salimans et al., 2015)	
DLGM + HVI (1 leapfrog step)	88.08
DLGM + HVI (4 leapfrog steps)	86.40
DLGM + HVI (8 leapfrog steps)	85.51
Results below from (Gregor et al., 2014)	
DARN $n_h = 500$	84.71
DARN $n_h = 500$, adaNoise	84.13

Generative normalizing flows

Idea: apply a sequence of invertible transformations to a random variable



Additional practical requirement: The inverse function also needs to be easy to compute!



Density estimation using Real NVP

Dinh, Sohl-Dickstein, Bengio 2016

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ s^\theta(x_1) \odot x_2 + t^\theta(x_1) \end{bmatrix}$$

Non-volume preserving

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} z_1 \\ (z_2 - t^\theta(z_1)) \oslash s^\theta(z_1) \end{bmatrix}$$

Image generation with RealNVP

Dinh, Sohl-Dickstein, Bengio 2016

Figure 5: On the left column, examples from the dataset. On the right column, samples from the model trained on the dataset. The datasets shown in this figure are in order: CIFAR-10, Imagenet (32×32), Imagenet (64×64), CelebA, LSUN (bedroom).

Masked autoregressive flow for density estimation

Papamakarios et al. 2017

Latent z to data x: autoregressive

$$x_i = \mu_i(x_{1:i-1}) + z_i \odot \sigma_i(x_{1:i-1})$$

Data x to latent z: parallel

$$z_i = (x_i - \mu_i(x_{1:i-1}))\sigma_i^{-1}(x_{1:i-1})$$

Glow: generative flow with invertible 1x1 convolutions

Kingma & Dhariwal 2018

1x1 convolution: invertible matrix multiplication along the channel dimension

Figure 7: Samples from model trained on 5-bit LSUN bedrooms, at temperature 0.875. Resolutions 64, 96 and 128 respectively [4]

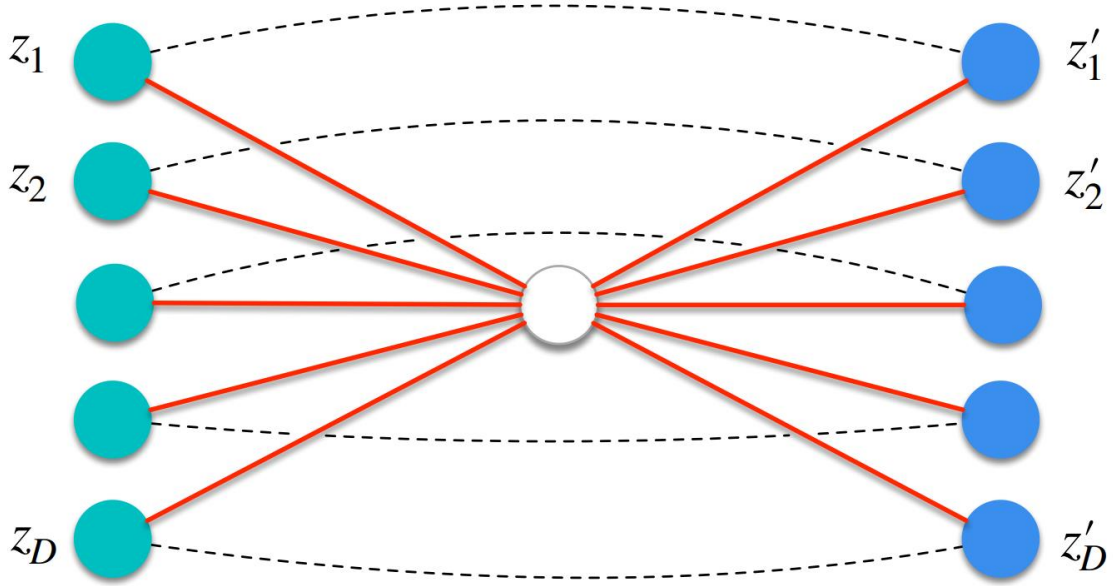
Extra NF slides

Planar flows

Rezende & Mohamed, ICML 2015

$$z' = z + u\, h(w^T z + b)$$

2 layer block single hidden unit with a residual connection



Planar flows: very simple jacobian determinant

$$z' = z + u\, h(w^T z + b)$$

$h(z) = \tanh(z)$

$w^T u \geq -1$

Simple determinant:

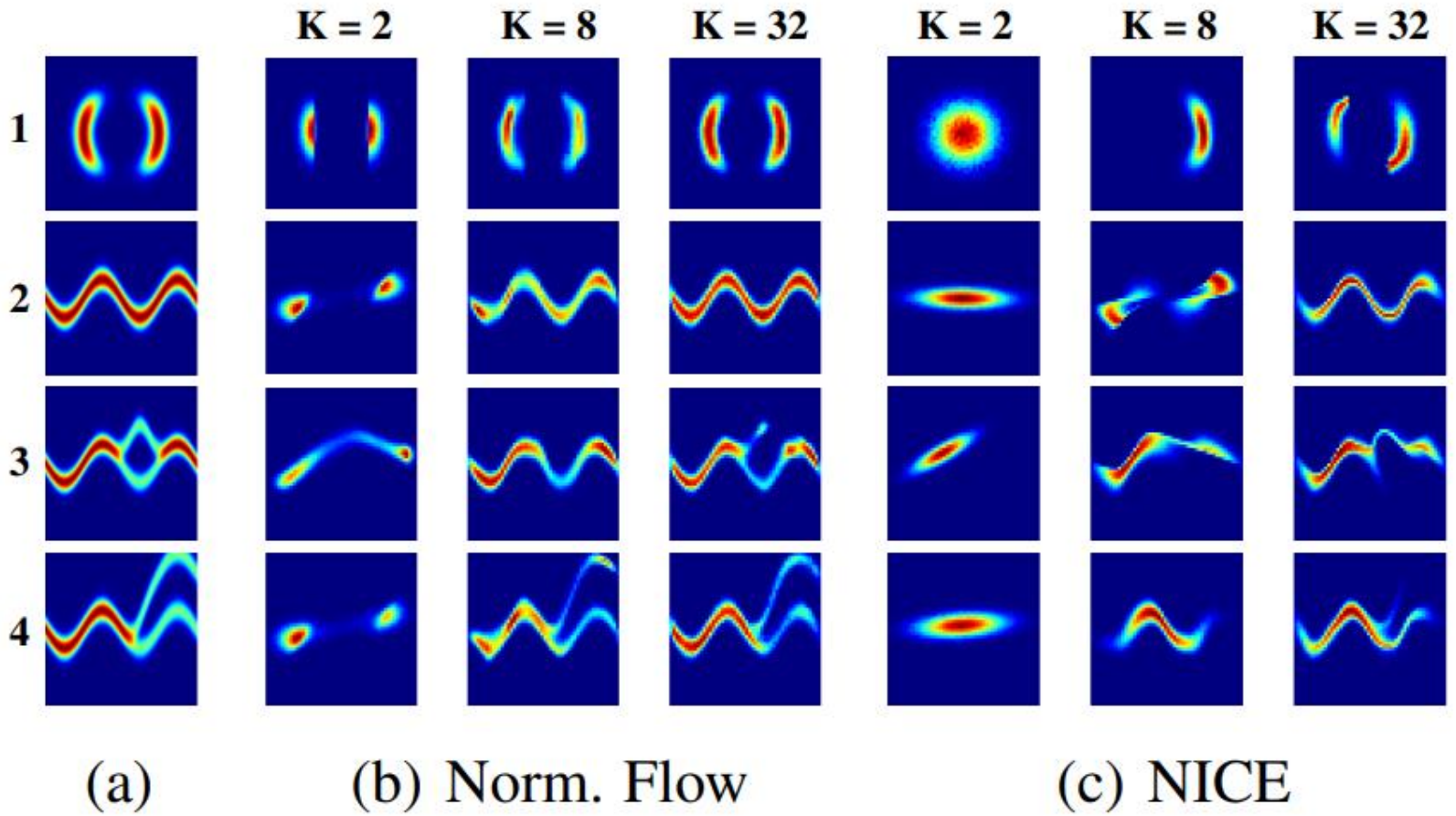
$$\det \frac{\partial z'}{\partial z} = \det \left(\mathbf{1}_D + u h'(w^T z + b) w^T \right)$$
$$= 1 + h'(w^T z + b) w^T u$$

Matrix determinant lemma:

$$\det(A + uv^T) = (1 + v^T A^{-1} u) \det(A)$$

Planar flows

Rezende & Mohamed, ICML 2015



IAF results

Table 2: Our results with ResNet VAEs on CIFAR-10 images, compared to earlier results, in *average number of bits per data dimension* on the test set. The number for convolutional DRAW is an upper bound, while the ResNet VAE log-likelihood was estimated using importance sampling.

Method	bits/dim $\leq$
Results with tractable likelihood models:	
Uniform distribution (van den Oord et al., 2016b)	8.00
Multivariate Gaussian (van den Oord et al., 2016b)	4.70
NICE (Dinh et al., 2014)	4.48
Deep GMMs (van den Oord and Schrauwen, 2014)	4.00
Real NVP (Dinh et al., 2016)	3.49
PixelRNN (van den Oord et al., 2016b)	**3.00**
Gated PixelCNN (van den Oord et al., 2016c)	**3.03**
Results with variationally trained latent-variable models:	
Deep Diffusion (Sohl-Dickstein et al., 2015)	5.40
Convolutional DRAW (Gregor et al., 2016)	3.58
ResNet VAE with IAF (Ours)	**3.11**

Morning program: lecture

1. Variational auto-encoders
2. Normalizing flows
3. **Denoising diffusion models**

Slides based on presentations by

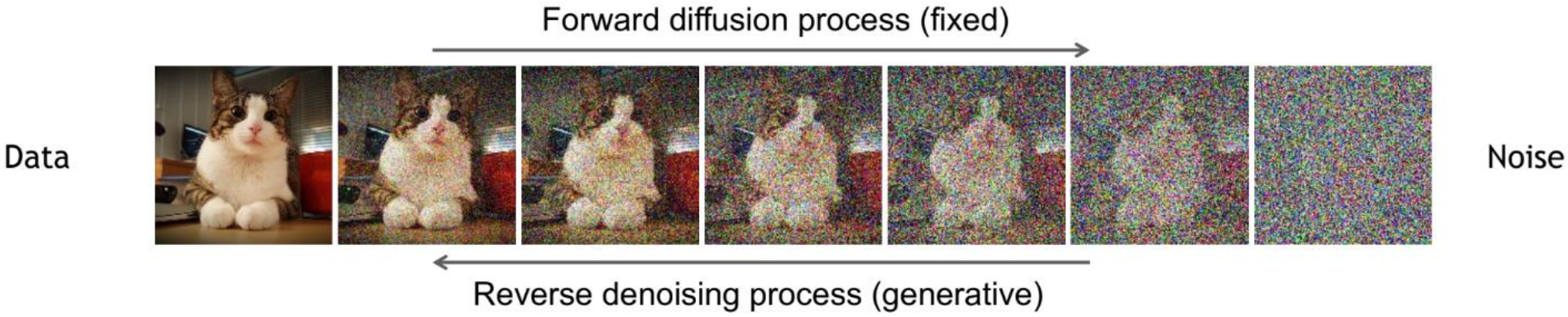
Daniel Zuegner
Senior Researcher
AI4Science, Microsoft Research
Berlin

Sarah Lewis
Senior RSDE
AI4Science, Microsoft Research
Cambridge, UK

What is a diffusion model?

Sohl-Dickstein et al., ICML 2015, Ho et al., NeurIPS 2020, Song et al., ICLR 2021

- A generative model that learns to **revert** a **diffusion process**.
- Forward diffusion process gradually destroys information in the data.
- **Alternative perspectives** on diffusion models:
  - Deep generative models with latent variables & ELBO maximization
  - (Denoising) score matching



Recap: Latent Variable Generative Model

A latent (unobserved) random variable

$$p_\theta(x) = \int \mathrm{d}z \, p_\theta(x, z)$$

Goal: sample $x \sim p_\theta(x)$ & maximize $\mathrm{KL}[q(x) || \, p_\theta(x)]$

Defining joint distribution as $p_\theta(x, z) = p_\theta(x|z)p(z)$ allows ancestral sampling of $(x, z) \sim p_\theta(x, z)$.

Introducing (a lot of) latent variables

$$\log p_\theta(\boldsymbol{x}) = \log \int \dots \int p_\theta(\boldsymbol{x}, \boldsymbol{z}_1, \dots, \boldsymbol{z}_T)\, \mathrm{d}\boldsymbol{z}_1 \dots \mathrm{d}\boldsymbol{z}_T$$

$$\log p_\theta(\boldsymbol{x}) = \log \int \dots \int \frac{q(\boldsymbol{z}_1, \dots, \boldsymbol{z}_T|\boldsymbol{x})}{q(\boldsymbol{z}_1, \dots, \boldsymbol{z}_T|\boldsymbol{x})} p_\theta(\boldsymbol{x}, \boldsymbol{z}_1, \dots, \boldsymbol{z}_T)\, \mathrm{d}\boldsymbol{z}_1 \dots \mathrm{d}\boldsymbol{z}_T$$

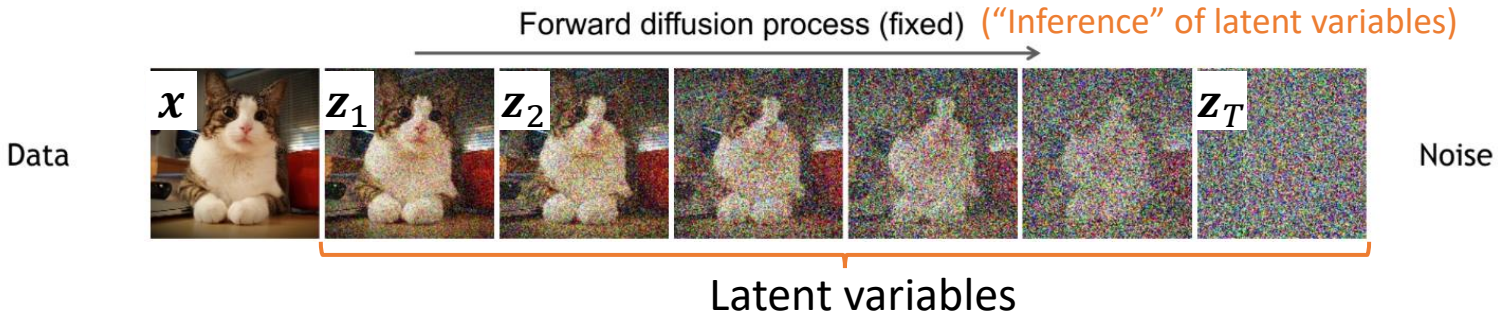
$$\log p_\theta(\boldsymbol{x}) = \log \mathbb{E}_{q(\boldsymbol{z}_1,\dots,\boldsymbol{z}_T|\boldsymbol{x})} \left[\frac{p_\theta(\boldsymbol{x}, \boldsymbol{z}_1, \dots, \boldsymbol{z}_T)}{q(\boldsymbol{z}_1, \dots, \boldsymbol{z}_T|\boldsymbol{x})} \right]$$

$$\log p_\theta(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{z}_1,\dots,\boldsymbol{z}_T|\boldsymbol{x})} \log \left[\frac{p_\theta(\boldsymbol{x}, \boldsymbol{z}_1, \dots, \boldsymbol{z}_T)}{q(\boldsymbol{z}_1, \dots, \boldsymbol{z}_T|\boldsymbol{x})} \right]$$

How to factorize $p_\theta(\boldsymbol{x}, \boldsymbol{z}_1, \dots, \boldsymbol{z}_T)$ and $q(\boldsymbol{z}_1, \dots, \boldsymbol{z}_T|\boldsymbol{x})$?

How to factorize $q(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{x})$ & $p_\theta(\boldsymbol{z}_T, \ldots \boldsymbol{z}_1, \boldsymbol{x})$?

Sohl-Dickstein et al., ICML 2015



Forward: Define a Markov Chain to "infer" the latents given the data:

$$q(\boldsymbol{x}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T) = q(\boldsymbol{x})q(\boldsymbol{z}_1, \ldots \boldsymbol{z}_T|\boldsymbol{x}) = \; q(\boldsymbol{x}) \cdot q(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$$

Backward:

$$p_\theta(\boldsymbol{z}_T, \ldots \boldsymbol{z}_1, \boldsymbol{x}) = p_\theta(\boldsymbol{z}_T)p_\theta(\boldsymbol{x}|\boldsymbol{z}_1)\prod_{t=2}^{T} p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$$

Deriving the negative ELBO

Sohl-Dickstein et al., ICML 2015, Ho et al., NeurIPS 2020, Song et al., ICLR 2021

Definitions of p and q:

$$q(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{x}) = q(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$$

$$p_\theta(\boldsymbol{z}_T, \ldots \boldsymbol{z}_1, \boldsymbol{x}) = p_\theta(\boldsymbol{z}_T) p_\theta(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$$

Plugging it all in:

$$-\log p_\theta(\boldsymbol{x}) \leq -\mathbb{E}_{q(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{x})} \log \left[\frac{p_\theta(\boldsymbol{x}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T)}{q(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{x})} \right]$$

$$= -\mathbb{E}_{q(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{x})} \log \left[\frac{p_\theta(\boldsymbol{z}_T) p_\theta(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{q(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})} \right]$$

$$= -\mathbb{E}_{q(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{x})} \left[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z}_1)}{q(\boldsymbol{z}_1|\boldsymbol{x})} + \sum_{t=2}^{T} \log \frac{p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})} + \log p(\boldsymbol{z}_T) \right]$$

Alternative version of -ELBO

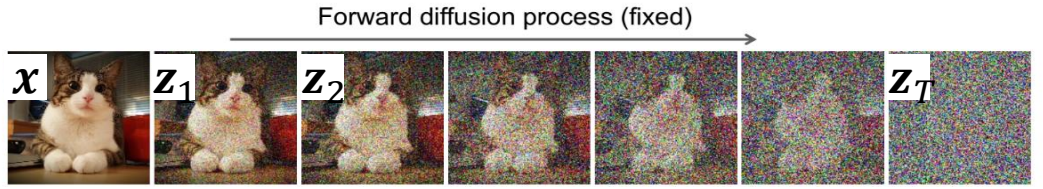
Sohl-Dickstein et al., ICML 2015 ; Ho et al., NeurIPS 2020

$$-\log p_\theta(x) \leq -\mathbb{E}_{q(z_1, \dots, z_T|x)}\left[\log \frac{p_\theta(x|z_1)}{q(z_1|x)} + \sum_{t=2}^{T} \log \frac{p_\theta(z_{t-1}|z_t)}{q(z_t|z_{t-1})} + \log p(z_T)\right]$$

Use $q(z_t|z_{t-1}) = q(z_t|z_{t-1}, x) = \frac{q(z_{t-1}|z_t, x)q(z_t|x)}{q(z_{t-1}|x)}$

$$-\log p_\theta(x) \leq -\mathbb{E}_{q(z_1|x)} \log p_\theta\,(x|z_1) + \sum_{t=2}^{T} \mathbb{E}_{q(z_t|x)}[KL[q(z_{t-1}|z_t, x)||p_\theta(z_{t-1}|z_t)]]$$
$$+ KL[q(z_T|x)||p_\theta(z_T)]$$

Interesting properties



$$-\log p_\theta(x) \leq -\mathbb{E}_{q(z_1|x)} \log p_\theta\,(x|z_1) + \sum_{t=2}^{T} \mathbb{E}_{q(z_t|x)}[KL[q(z_{t-1}|z_t, x)||p_\theta(z_{t-1}|z_t)]]$$
$$+ KL[q(z_T|x)||p_\theta(z_T)]$$

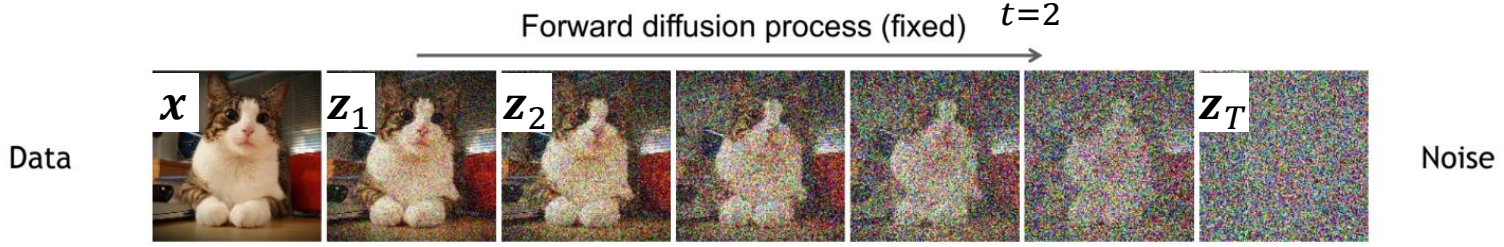
When $T \to \infty$:

1. Functional form of forward process and reverse process is the same. [*W. Feller. 1949*]

2. For many choices of $q(z_t|z_{t-1})$, we get a stationary distribution $\lim_{t\to\infty} q(z_t|x) = \pi(z)$

Training diffusion models

$$q(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{x}) = q(\boldsymbol{z}_1|\boldsymbol{x}) \prod_{t=2}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$$



$$p_\theta(\boldsymbol{z}_T, \ldots \boldsymbol{z}_1, \boldsymbol{x}) = p_\theta(\boldsymbol{z}_T) p_\theta(\boldsymbol{x}|\boldsymbol{z}_1) \prod_{t=2}^{T} p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$$

$$L_{vb} = \mathbb{E}_{q(x)}[-\mathbb{E}_{q(z_1|x)} \log p_\theta(x|z_1) + \sum_{t=2}^{T} \mathbb{E}_{q(z_t|x)}[KL[q(z_{t-1}|z_t, x)||p_\theta(z_{t-1}|z_t)]] + KL[q(z_T|x)||p_\theta(z_T)]]$$

Practical requirements for $q(x_t|x_{t-1})$ to allow for efficient training of p_θ:

1. Efficient sampling of x_t from $q(x_t|x_0)$ for arbitrary time t
2. Tractable expression for $q(x_{t-1}|x_t, x_0)$.

For Gaussian or binomial $q(x_t|x_{t-1})$ (and $p_\theta(x_{t-1}|x_t)$): ☑ ☑

Diffusion models with Gaussian distributions

Gaussian forward distributions: $q(z_t|z_{t-1}) = \mathcal{N}(z_t|\sqrt{\beta_t}z_{t-1}, (1-\beta_t)\mathrm{I})$

1. Sampling arbitrary timesteps in one shot:

$$q(z_t|x) = \mathcal{N}\left(z_t|\sqrt{\bar{\alpha}_t}x, (1-\bar{\alpha}_t)\mathrm{I}\right)$$

$$\alpha_t = 1 - \beta_t$$

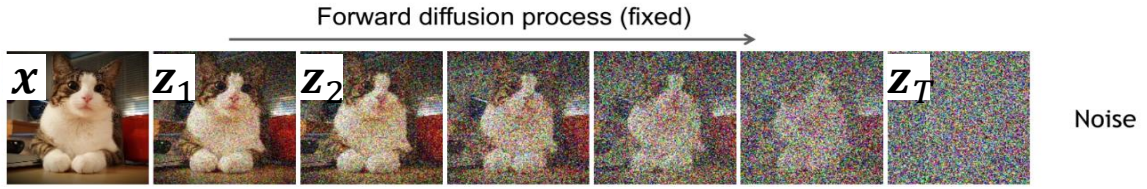
$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

2. Tractable expression for posterior:

$$q(z_{t-1}|z_t, x) = \mathcal{N}\left(z_{t-1}|\tilde{\mu}_t(z_t, x), \tilde{\beta}_t\mathrm{I}\right)$$

$$\tilde{\mu}_t(z_t, x) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}z_t \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Recap: Interesting properties



$$-\log p_\theta(x) \leq -\mathbb{E}_{q(z_1|x)} \log p_\theta(x|z_1) + \sum_{t=2}^{T} \mathbb{E}_{q(z_t|x)}[KL[q(z_{t-1}|z_t, x)||p_\theta(z_{t-1}|z_t)]]$$

$$+ KL[q(z_T|x)||p_\theta(z_T)]$$

When $T \to \infty$:

1. Functional form of forward process and reverse process is the same
   → $p_\theta(z_{t-1}|z_t) = \mathcal{N}(z_{t-1}|\mu_\theta(z_t, t), \sigma_t I)$

2. For many choices of $q(z_t|z_{t-1})$, we get a stationary distribution $\lim_{t\to\infty} q(z_t|x) = \pi(z)$
   → for large T and small $\beta_1 < \beta_2 < \cdots < \beta_T$ $q(z_T|x) \approx \mathcal{N}(0, I)$
   → pick $p_\theta(z_T) = \mathcal{N}(0, I)$

Parameterizing $\mu_\theta(z_t, t)$

Ho et al., NeurIPS 2020

$$\mathbb{E}_{q(z_t|x)}[KL[q(z_{t-1}|z_t, x)||p_\theta(z_{t-1}|z_t)] = \mathbb{E}_{q(z_t|x)}\left[\frac{1}{2\sigma_t^2}\left|\left|\tilde{\mu}_t(z_t, x) - \mu_\theta(z_t, t)\right|\right|^2\right] + C$$

Simplest option: $\mu_\theta(z_t, t) = \mathrm{nn}_\theta(z_t, t)$

Recall:

$$\boldsymbol{z}_t = \sqrt{\alpha_t}\boldsymbol{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}$$

Alternative:

$$\tilde{\mu}_t(z_t, x) = \frac{1}{\sqrt{\alpha_t}}\left(z_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon(x, z_t)\right)$$

Predict the noise:

$$\mu_\theta(z_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(z_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(z_t, t)\right)$$

Loss, training and sampling

Ho et al., NeurIPS 2020

$$L_{vlb} = \mathbb{E}_{q(x)}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\mathbb{E}_{t\sim U(2,T)}\left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2\alpha_t\,(1-\bar{\alpha}_t)}}_{\lambda_t}\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\boldsymbol{x}+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t\right)\right\|_2^2\right] + \text{reconstruct}$$

Ho et al. 2020 found that setting $\lambda_t = 1$ improves sample quality, i.e., the training loss is:

$$L_{simple} = \mathbb{E}_{q(x)}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\mathbb{E}_{t\sim U(2,T)}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\boldsymbol{x}+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t\right)\right\|_2^2\right] + \text{reconstruct}$$

Algorithm 1 Training

1: **repeat**
2: $\boldsymbol{x} \sim q(\boldsymbol{x})$
3: $t \sim \text{Uniform}(\{1,\dots,T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\alpha_t}\,\boldsymbol{x}+\sqrt{1-\alpha_t}\,\boldsymbol{\epsilon},t)\right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
2: **for** $t = T,\dots,1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{z}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\epsilon}_\theta(\mathbf{z}_t,t)\right) + \sigma_t\mathbf{z}$
5: **end for**
6: **return** $\boldsymbol{z}_0$

Results from Ho et al. 2020

Sohl-Dickstein et al., ICML 2015, Ho et al., NeurIPS 2020

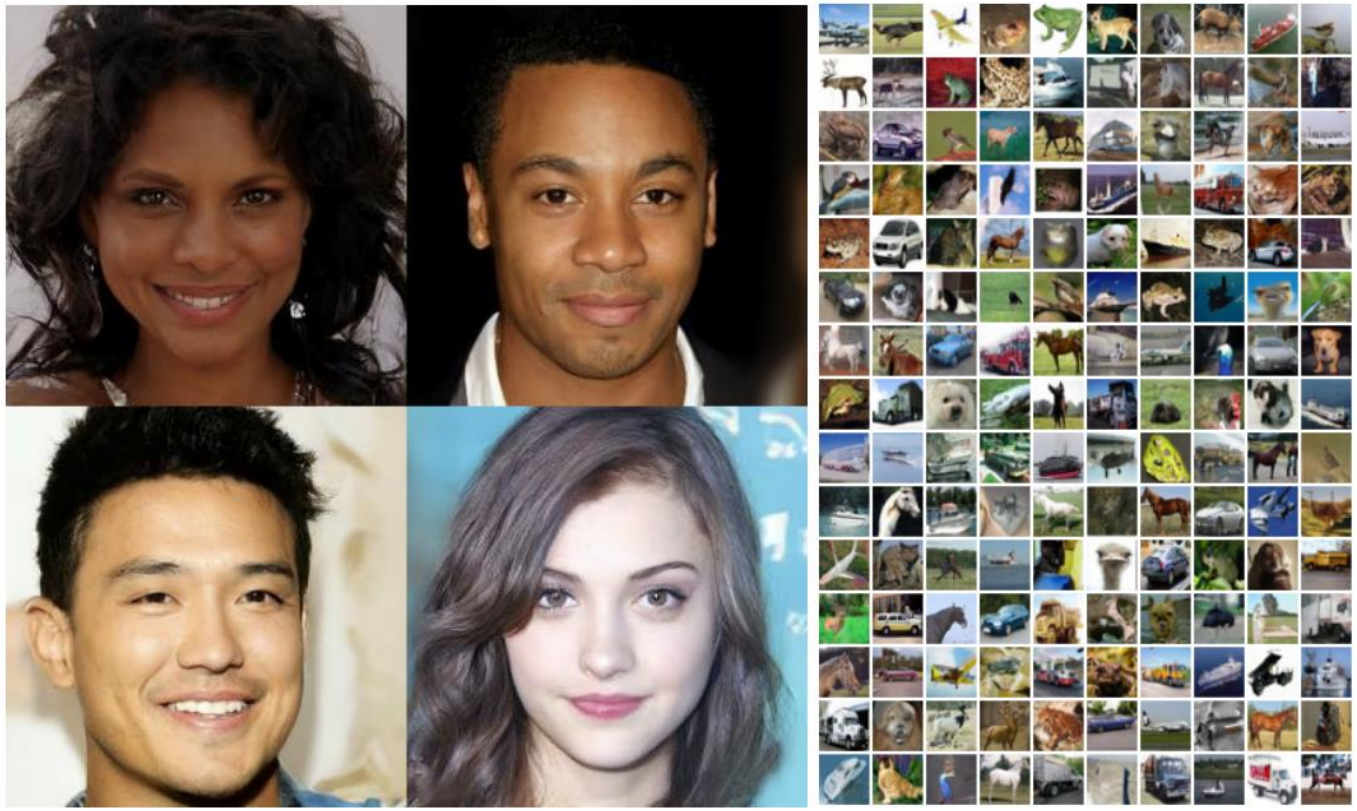


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			$\leq$ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87$\pm$0.12	25.32	
SNGAN [39]	8.22$\pm$0.05	21.7	
SNGAN-DDLS [4]	9.09$\pm$0.10	15.42	
StyleGAN2 + ADA (v1) [29]	**9.74** $\pm$ 0.05	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67$\pm$0.13	13.51	$\leq$ 3.70 (3.69)
Ours (L_{simple})	9.46$\pm$0.11	**3.17**	$\leq$ 3.75 (3.72)

Score-matching perspective

Energy-based models:

$$p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta}, \qquad Z_\theta = \int e^{-f_\theta(x)} dx$$

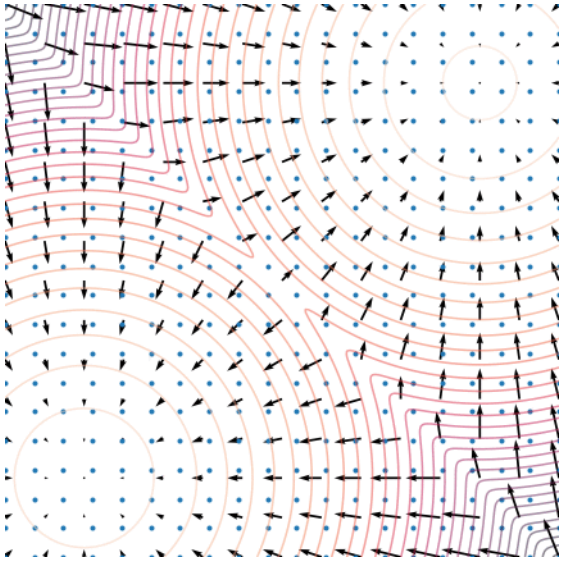
Intractable in general

Score-based modeling: define the **score function** instead:

$$\boldsymbol{s}_\theta(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p_\theta(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log Z_\theta = 0$$

Optimization: score matching objective

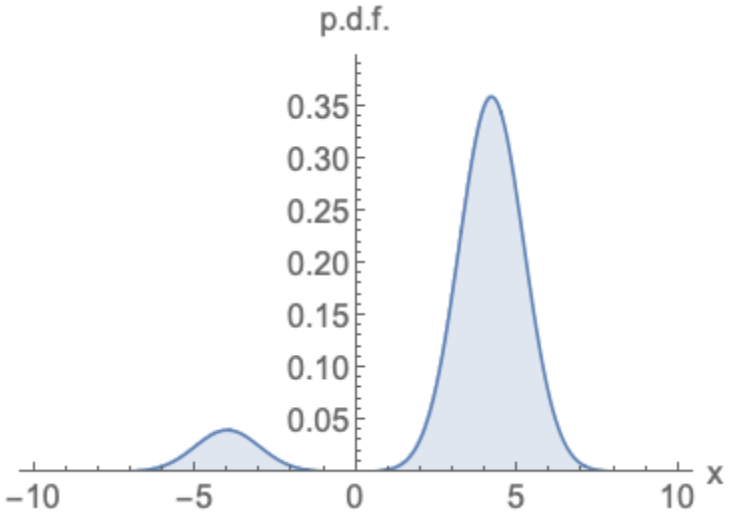
Sampling: Langevin dynamics

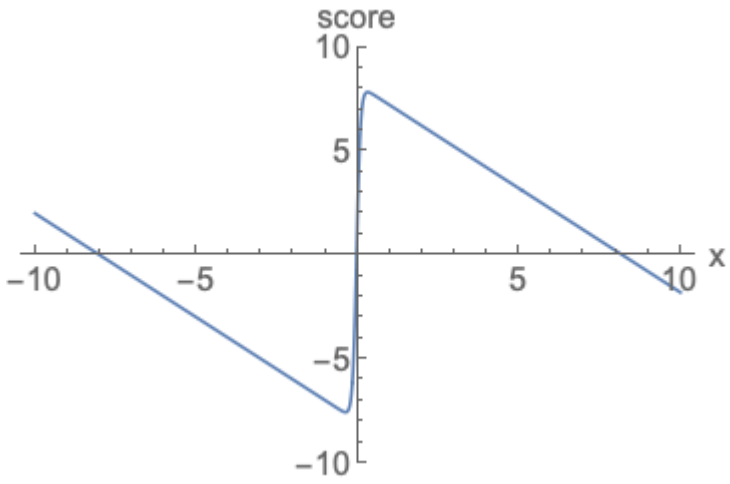


Sampling via the score function of a Gaussian mixture

[Generative Modeling by Estimating Gradients of the Data Distribution | Yang Song (yang-song.net)]

Score function example





[Generative Modeling by Estimating Gradients of the Data Distribution | Yang Song (yang-song.net)]

Score matching training

Explicit score matching

$$L_{ESM} = \mathbb{E}_{q(\boldsymbol{x})}[\, \| \nabla_{\boldsymbol{x}} \log q(\boldsymbol{x}) - \boldsymbol{s}_\theta(\boldsymbol{x}) \|_2^2 \,]$$

Unknown ☹

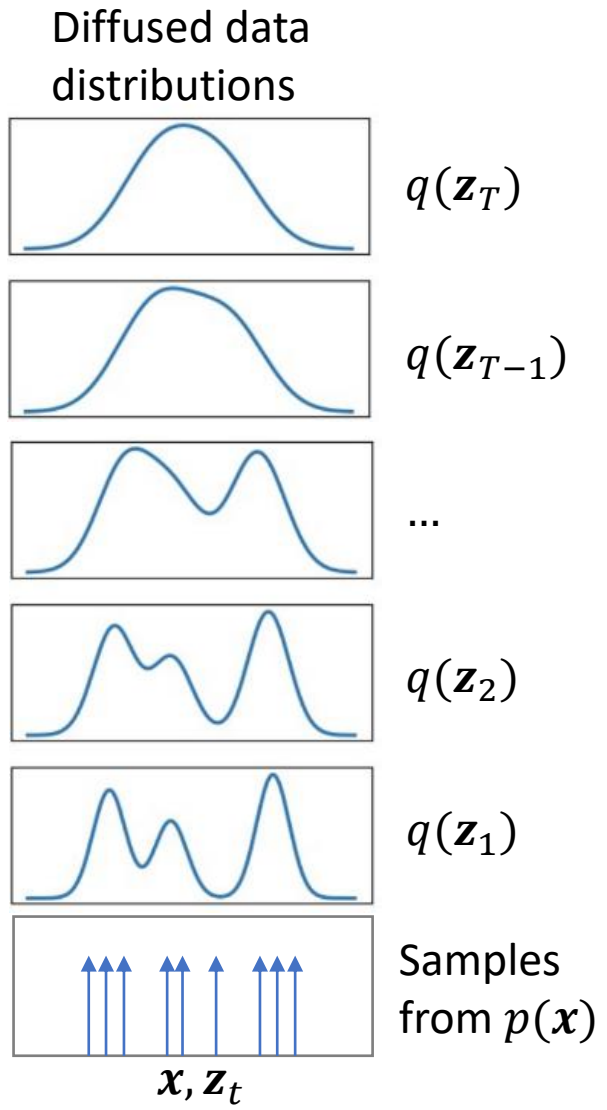
Explicit score matching with tractable non-parametric estimator of data dist:

$$q_\sigma(\tilde{\boldsymbol{x}}) = \int q(\boldsymbol{x})\, q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x} = \mathbb{E}_{q(\boldsymbol{x})}[q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})]$$

where $q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \mathcal{N}(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}, \sigma \mathbf{I})$.

If $\sigma \approx 0 \Rightarrow q_\sigma(\tilde{\boldsymbol{x}}) \approx q(\boldsymbol{x})$

$$\mathrm{L}_{\mathrm{ESM}_\sigma} = \mathbb{E}_{\tilde{\boldsymbol{x}} \sim q_\sigma(\tilde{\boldsymbol{x}})}[\, \| \nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}}) - \boldsymbol{s}_\theta(\tilde{\boldsymbol{x}}) \|_2^2 \,]$$ [Vincent, 2011]



Adapted from
Denoising Diffusion-based Generative Modeling: Foundations and Applications (cvpr2022-tutorial-diffusion-models.github.io)

Score matching training

Plugging in the score of q_σ into the objective:

$$\mathrm{L}_{\mathrm{ESM}_\sigma} = \mathbb{E}_{\tilde{\boldsymbol{x}}\sim q_\sigma(\tilde{\boldsymbol{x}})}[\, \|\nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}}) - \boldsymbol{s}_\theta(\tilde{\boldsymbol{x}})\|_2^2 \,]$$

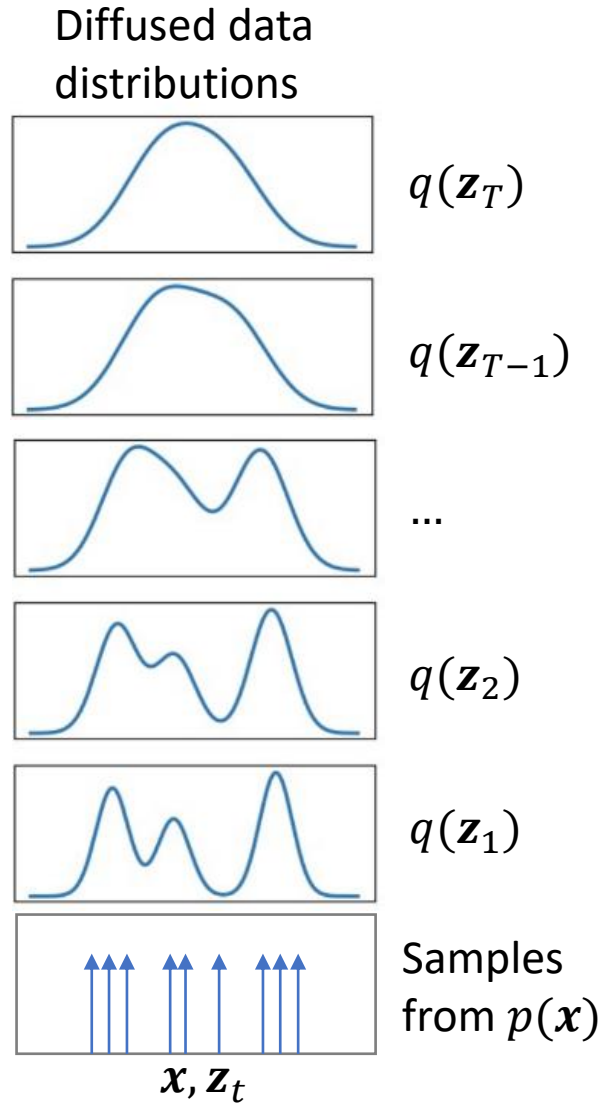
Very expensive to evaluate

Recall: $q_\sigma(\tilde{\boldsymbol{x}}) = \mathbb{E}_{q(\boldsymbol{x})}[q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})], \qquad q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \mathcal{N}(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}, \sigma\mathbf{I}).$

Denoising score matching

[Vincent, 2011]: the following objective is equivalent to the above

$$L_{\mathrm{DSM}_\sigma} = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}),\tilde{\boldsymbol{x}}\sim q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})}[\, \|\nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) - \boldsymbol{s}_\theta(\tilde{\boldsymbol{x}})\|_2^2 \,]$$



Adapted from
Denoising Diffusion-based Generative Modeling: Foundations and Applications (cvpr2022-tutorial-diffusion-models.github.io)

Diffusion models and Score Matching

- Denoising score matching objective: min $\mathbb{E}_{q(\boldsymbol{x})}[\|\nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) - \boldsymbol{s}_\theta(\tilde{\boldsymbol{x}})\|_2^2]$
- Choose $q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \boldsymbol{\mathcal{N}}\left(\tilde{\boldsymbol{x}}\middle|\sqrt{\bar{\alpha}}\boldsymbol{x}, (1-\bar{\alpha})\mathbf{I}\right)$ DDPM noise distribution
- $\nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \frac{-(\overbrace{\sqrt{\bar{\alpha}}\boldsymbol{x}+\sqrt{1-\bar{\alpha}}\boldsymbol{\epsilon}}^{\tilde{\boldsymbol{x}}} - \overbrace{\sqrt{\bar{\alpha}}\boldsymbol{x}}^{\boldsymbol{\mu}})}{1-\bar{\alpha}} = \frac{-\boldsymbol{\epsilon}}{\sqrt{1-\bar{\alpha}}}$
- Plugging in:

$$L_{\text{DSM}} = \frac{1}{1-\bar{\alpha}} \mathbb{E}_{\boldsymbol{x}\sim q(x), \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{s}_\theta\left(\sqrt{\bar{\alpha}}\boldsymbol{x} + \sqrt{1-\bar{\alpha}}\boldsymbol{\epsilon}\right)\right\|_2^2\right]$$

Compare with **DDPM** loss:

$$L_{\text{simple}} = \mathbb{E}_{\boldsymbol{x}\sim q, \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}), t\sim\mathcal{U}(2,T)}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\boldsymbol{x} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|_2^2\right] + \ldots$$

“Weird” corruption

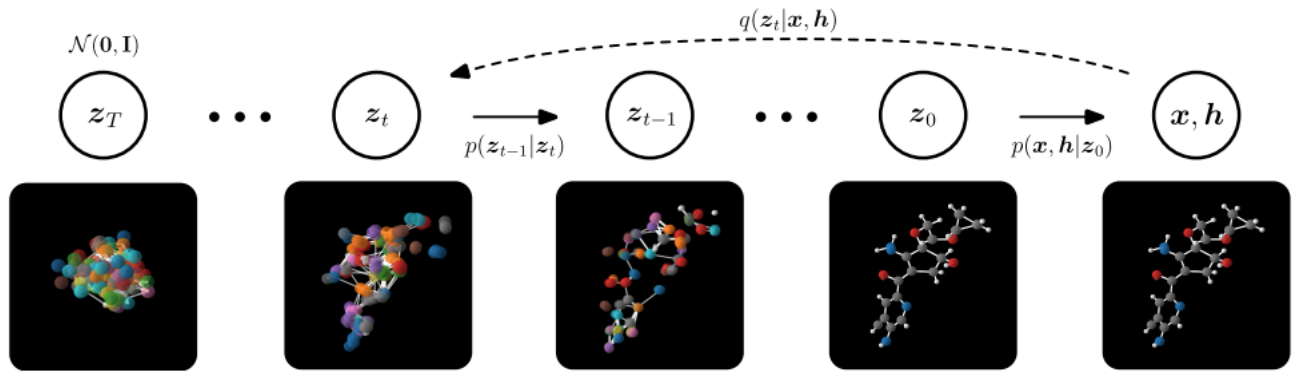
- [2107.03006] Structured Denoising Diffusion Models in Discrete State-Spaces (arxiv.org)
- [2112.10752] High-Resolution Image Synthesis with Latent Diffusion Models (arxiv.org)
- [2208.09392] Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise (arxiv.org)

Afternoon program

Workshop on Diffusion Models for Molecule Generation

Equivariant Diffusion for Molecule Generation in 3D

Emiel Hoogeboom [*1] Victor Garcia Satorras [*1] Clément Vignac [*2] Max Welling [1]



Extra slides

Why AI4Science at MSR?

A unique convergence of:

- Fundamental Science
- Heavy-duty computational Modeling
- Timely applications