Variational Autoencoders

Denis Derkach, Maksim Artemev, Artem Ryzhikov

CS HSE faculty, Generative Models, spring 2020

Reminder: Autoencoders

Autoencoders are a type of neural networks with two distinct parts:

- **Encoder** f: Perform a dimensionality reduction step on the data, $x \in \mathbb{R}^n$ to obtain features $h \in \mathbb{R}^d$.
- **Decoder** g: Maps the features $h \in \mathbb{R}^d$ to reproduce input $x \in \mathbb{R}^n$.

Thus, we want to construct:

$$\hat{x} = f(g(x));$$

with $x \in \mathbb{R}^n$, such that

$$\mathcal{L}(\hat{x}, x) \to 0.$$

Reminder: PCA as an autoencoder

In general, PCA can be thought of as an linerarised version of AE:

- $h = f(x) = W^T x$
- $g(h) = Wh$

with $W \in \mathbb{R}^{n \times d}$. Then the loss function is:

$$\mathcal{L} = ||x - WW^T x||^2.$$

The matrix W will have the following key property:
If $W = U\Sigma V^T$ is the singular value decomposition of W, then U will span the same subspace as the top d eigenvectors of $\mathrm{cov}\,(x)$.
If we introduce some nonlineraity (or even deep neural network), we will have a nonlinear analogue of PCA.

Reminder: AE in the wild

The autoencoder may be extended in several ways. Some of them are:

- sparse autoencoders ($L(x, \hat{x}) + \sum_i^{N_h} |h_i|$);
- contractive autoencoders ($L(x, \hat{x}) + ||\nabla_x h_i||^2$);
- denoising autoencoders ($L(\tilde{x}, \hat{x})$).

In general, denoising autoencoders can be used to generate events (in combination with Markov-Chain Monte Carlo).
A more complicated example is to create Masked Autoencoder to mimic Autoregression.

Motivation

- We have seen already that the NNs cannot produce a viable generative model (if trained in the same manner as usual).
- The main problem is the absence of probabilistic algorithm that can properly generate.
- We tried to solve this with autoregressive models.
- What else can be done?

Reminder: Latent variables

- Allow us to define complex models $p(x)$ in terms of simple building blocks $p(x|z)$.
- Natural for unsupervised learning tasks (clustering, unsupervised representation learning, etc.)
- No free lunch: much more difficult to learn compared to fully observed, autoregressive models.

Variational Inference: idea

- We have some m latent variables.
- let's pick a family of distributions $q(z_{1:m}|\nu)$ over the latent variables with its own variational parameters ν.
- find the algorithm for parameter fits that makes q close to the posterior of interest.
- Use q with the fitted parameters as a proxy for the posterior.

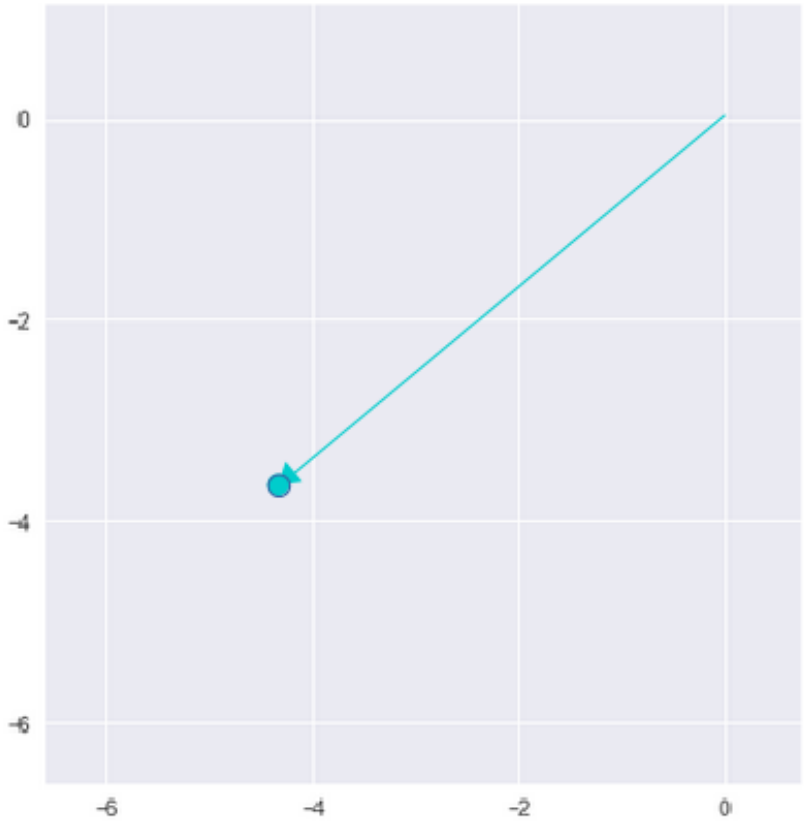
Variational Autoencoders: idea

Instead of learning $f()$ and $g()$, we could learn distributions of the latent features given the input and the input features given the activations, i.e., probabilistic versions of $f()$ and $g()$. Concretely, the VAE will learn:

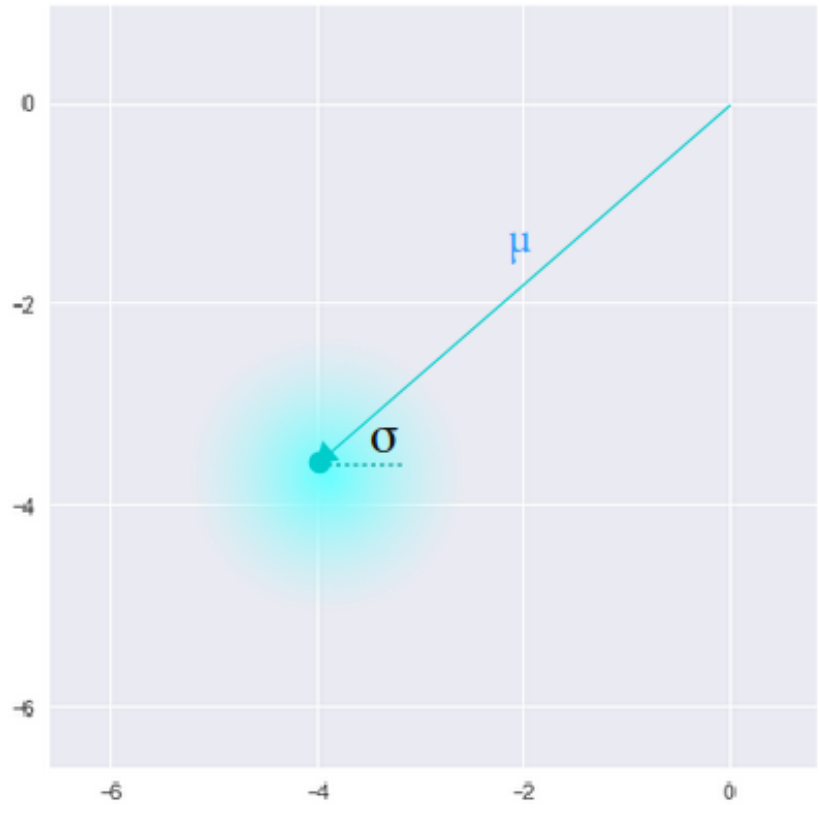
- $p(z|x)$: the distribution of the features given the input.
- $p(x|z)$: the distribution of the input given the features.

These distributions are parameterized by neural networks, meaning that they can express nonlinear transformations, and be trained via stochastic gradient descent.

Graphical motivation



Standard Autoencoder
(direct encoding coordinates)



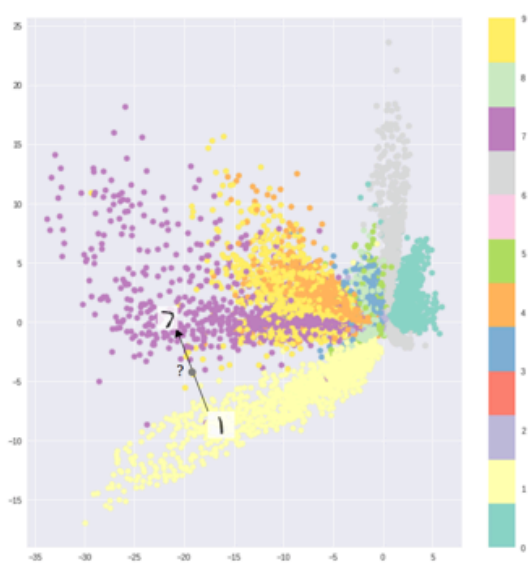
Variational Autoencoder
(μ and σ initialize a probability distribution)

The idea is to ”blur” the exact position in the latent space.

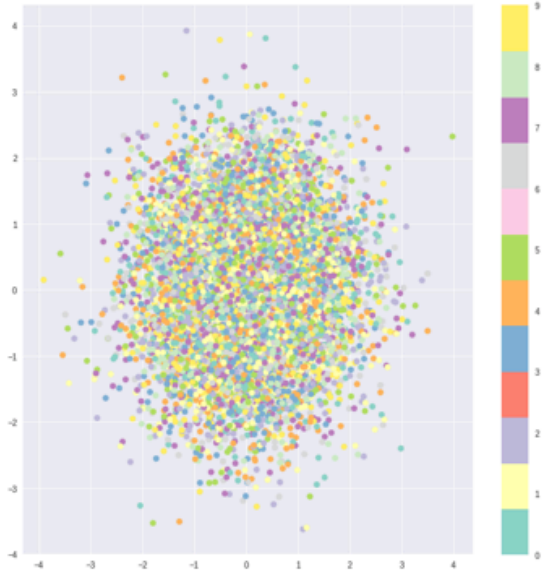
from J. Jordan’s blog

Graphical motivation

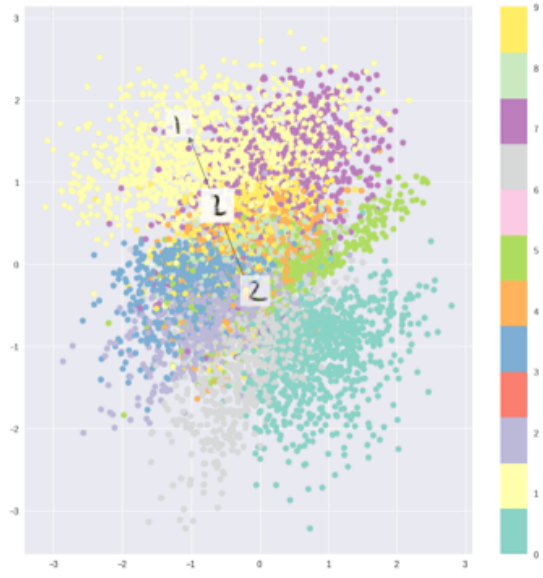
Only reconstruction loss



Only KL divergence



Combination



We need something that is able to show a smooth transition in latent space, but still separate classes.

from J. Jordan's blog

VAE as a generative model

Why do we want to learn distribution?

- Data is noisy - it is better to give some freedom.
- Latent and observed variables connection is quite nonlinear.
- AE becomes a generative model; we sample z and then sample x (using $p(x|z)$). And thus reconstruct a true probability:

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

Generating samples from $p(x)$

Knowing $p(z)$ and $p(x|z)$ we can:

1. Generate a sample randomly from $z_s \sim p(z)$. Call this sample z_s.
2. Then generate a sample $x_s \sim p(x|z = z_s)$. This sample of x_s is from $p(x)$.

If we do it many times, we will get:

$$p(x) \approx \frac{1}{n} \sum_{i=1}^{n} p(x, z_s^{(i)}),$$

which approximates the pdf of x.
We can use ancestral sampling here.

Designing the prior distribution, $p(z)$

How to choose $p(z)$?

- We can take $p(z) \sim \mathcal{N}(0, \mathcal{I})$;
- this can be done, since we have an expressive model as a decoder;
- in general, we can take any distribution, but the tightness of the lower bound depends on the specific choice of q.

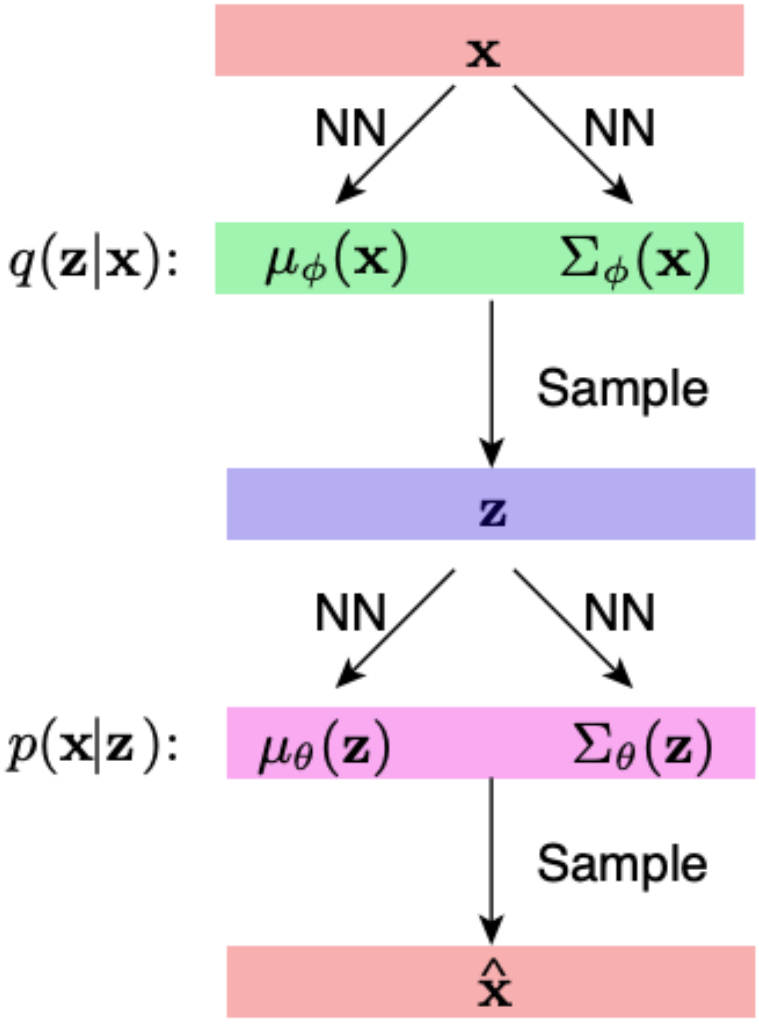
Modeling the distribution of x

We want a distribution of x but we want to do it via latent variable.
We can take:

$$x|z \sim \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))$$

Here, $\mu_\theta(z)$ is a neural network that takes z and outputs a vector in $\mathbb{R}^n$ that represents the mean of $x|z$. Similarly, $\Sigma_\theta(z)$ is a neural network that takes z and outputs a matrix in $\mathbb{R}^{n\times n}$ that represents the covariance of $x|z$.
Note: $x|z$ is normal, z is normal, but x is not normal due to nonlinear transformation of z.

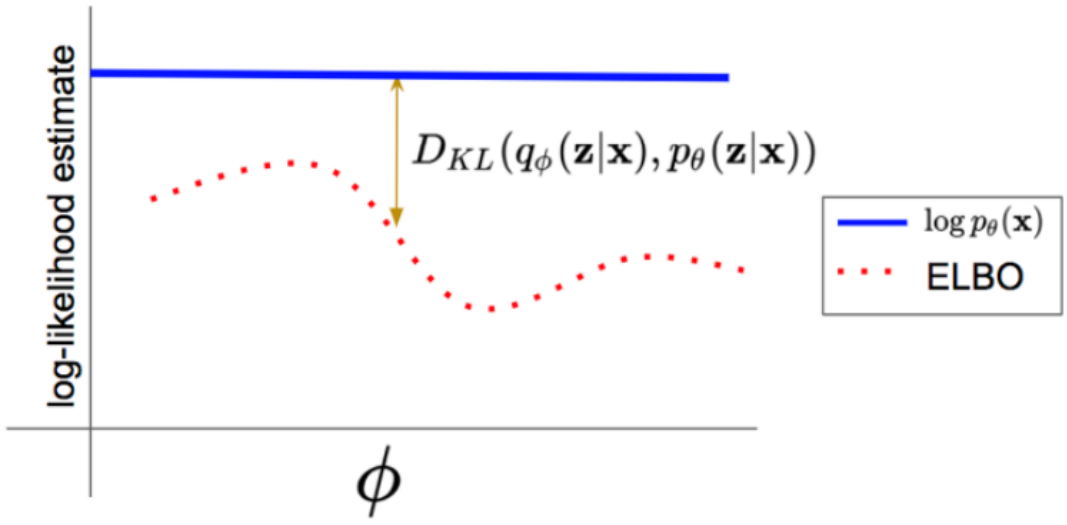
Conceptual diagram of the VAE



- looks similar to AE;
- inference changes from learning $z = f(x)$ to learning $q(z|x)$ and from learning $\hat{x} = g(z)$ to learning $p(x|z)$.
- instead of learning $\hat{x}$, we sample it.

from UCLA DL lectures by prof. Kao

Reminder: ELBO



Interpretation of:

$$KL(q_\phi(z|x)||p_\theta(z|x))$$

- The KL divergence of the approximate posterior from the true posterior;
- The better $q_\phi(z|x)$ approximates the true (posterior) distribution $p_\theta(z|x)$, in terms of the KL divergence, the smaller the gap.

Reminder: ELBO

Evidence lower bound (ELBO) holds for any q in canonical form:

$$\log p(x;\theta) \geq \sum_z q(z;\phi) \log p(z,x;\theta) + H(q(z;\phi)) \equiv \mathcal{L}(x;\theta,\phi).$$

We maximize the bound in order to maximize the log likelihood, l, for the entire dataset D:

$$l(D) = \sum_{x^i \in D} \log(p(x^i;\theta) \geq \sum_{x^i \in D} \mathcal{L}(x^i;\theta,\phi^i)$$

Note that in general variational parameters ϕ^i should be calculated for every data point x^i because the true posterior $p(z|x^i;\theta)$is different across datapoints x.

from Stanford S. Ermon and A. Grover Write-up

Maximum ELBO: intuition

Maximization of the ELBO brings two objectives:

- It will approximately maximize the marginal likelihood $p_\theta(x)$. Thus better generative moderl.
- It will minimize the KL divergence of the approximation $q_\phi(z|x)$ from the true posterior $p_\theta(z|x)$, so encoding part becomes better.

pre-SGD for ELBO

We can estimate gradients for ELBO:

- gradient wrt θ is easy:

$$\nabla_\theta \mathcal{L}(x; \theta, \phi) = \nabla_\theta \log p(x, z; \theta)$$

  and can be done using a simple MC estimation.

- for ϕ situation is much more tricky, since we calculate expectation over q:

$$\nabla_\phi \mathcal{L}(x; \theta, \phi) = \nabla_\phi \mathbb{E}_{q(z|x;\phi)}[\log p(x, z; \theta) - \log q(z|x; \phi)].$$

  luckily, we can use reparameterization.

The reparameterization trick

To get around the sampling inside backpropagation loop, we use something called the reparameterization trick. In the sampling operation, we want to sample:

$$z \sim q(z|x^i)$$

To do so, we sample $\varepsilon \sim \mathcal{N}(0, I)$ and calculate:

$$z = \mu_\phi(x^i) + \Sigma_\phi^{1/2}(x^i)\varepsilon$$

Then, z will be a sample from $q(z|x(i))$ as its a linear transformation of ε with mean $\mu_\phi(x(i))$ and covariance $\Sigma_\phi(xi)$. The sampling operation now occurs only for ε, which we do not need to backpropagate through.

Reparameterisation for Discrete Latent Observables

In fact, we can also use the reparameterisation trick for discrete variables:

- for $z \sim \{1; ..; K\}$ discrete rv with π_k - success probability;
- we define Gumbel distribution:

$$p(\varepsilon; \mu, \beta) = \exp(-\exp((-\varepsilon - \mu)/\beta)).$$

- then we define softmax:

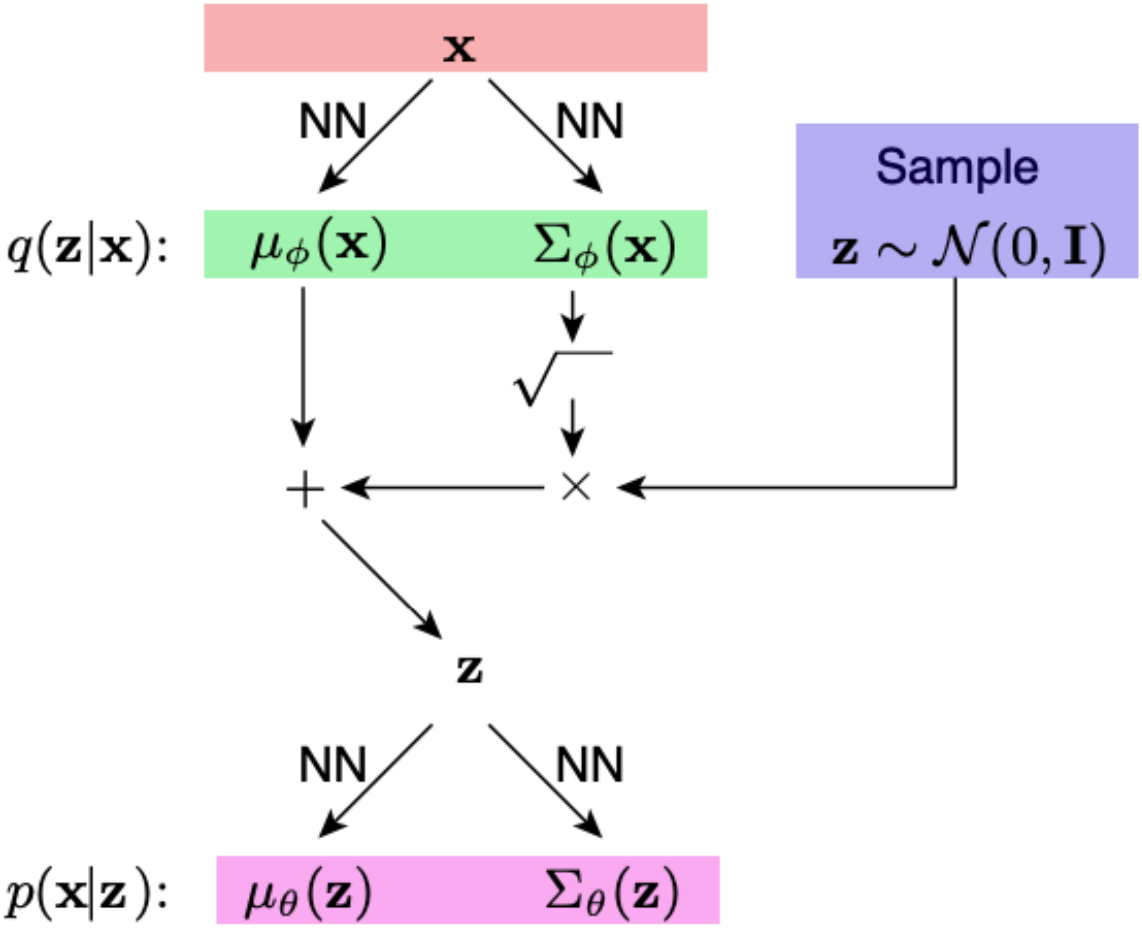
$$z_k = \exp(\ln \pi_k + \varepsilon_k) / \sum_{k=1}^{K} (\ln \pi_k + \varepsilon_k).$$

- proceed like in the continuous case.

E. Jang et al. Categorical Reparameterization with Gumbel-Softmax, ICLR 2017

C. Maddison et al. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables, ICLR 2017

Updated computational graph to calculate the ELBO



Updated Gradients

Under the reparameterisation, we can use:

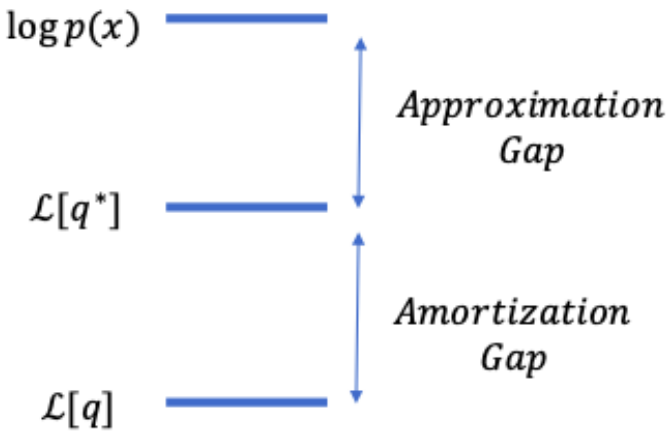
$$\mathcal{L}(x;\theta,\phi) = \mathbb{E}_{q(z|x;\phi)}[\log p(x,z;\theta)\log q(z|x;\phi)] = \\ = \mathbb{E}_{p(\varepsilon)}[\log p(x,z;\theta) - \log q(z|x;\phi)].$$

where $z = g(\varepsilon,\phi,x)$.
As a result we can form a simple Monte Carlo estimator $\mathcal{L}(x^i;\theta,\phi)$ of the individual-datapoint ELBO where we use a single noise sample ε from $p(\varepsilon)$:

$$\mathcal{L}(x^i;\theta,\phi) = \log p(x^i,z) - \log q(z|x;\phi)$$

Note that the gradient estimates from per-event ELBO will be unbiased.

Amortized Inference



One last problem remains

- variational parameters ϕ^i should be calculated for every data point x^i;
- for big datasets this is unsustainable;
- idea: forget about it and make a single model.

C. Cremer et al. Inference Suboptimality in Variational Autoencoders, ICML-2018

Amortization: problem

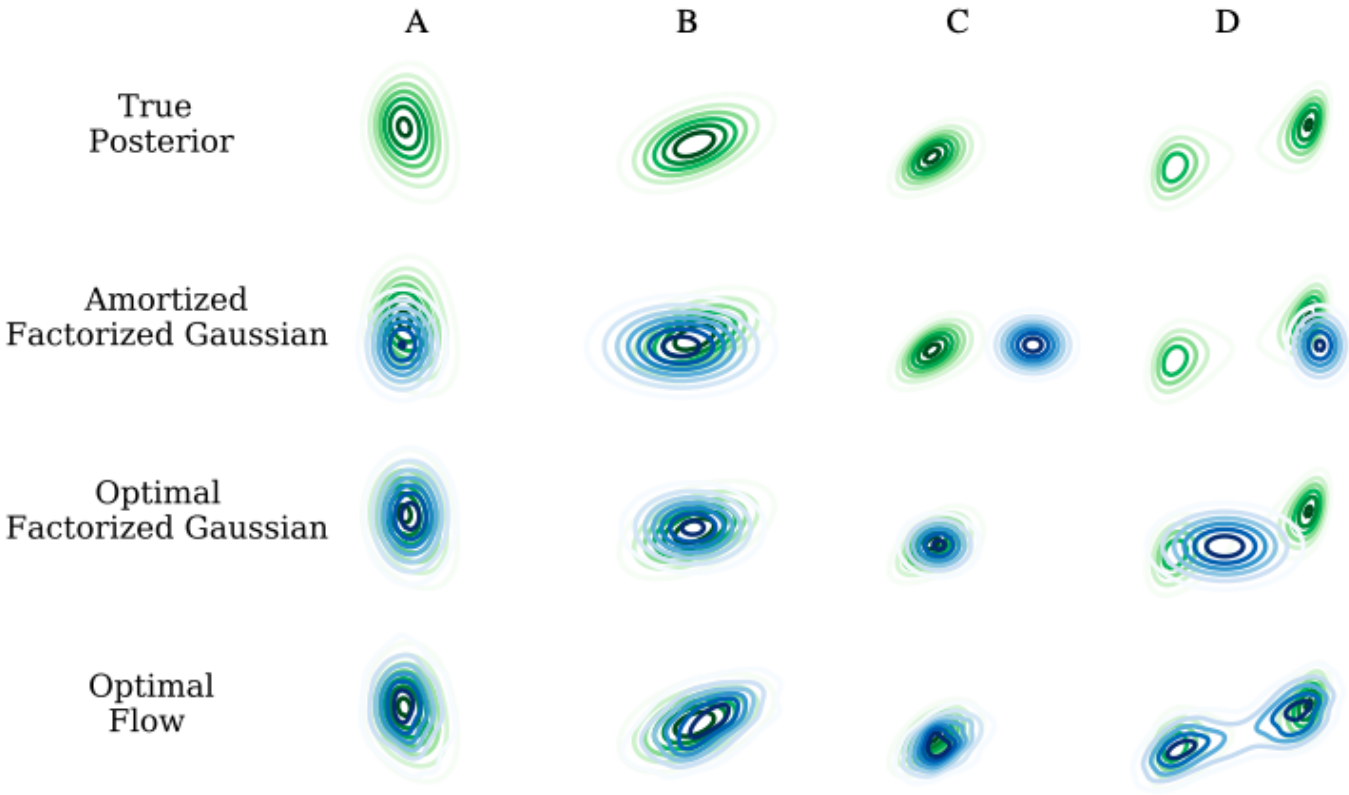


Figure 2. True Posterior and Approximate Distributions of a VAE with 2D latent space. The columns represent four different datapoints. The green distributions are the true posterior distributions, highlighting the mismatch with the blue approximations. Amortized: Variational parameters learned over the entire dataset. Optimal: Variational parameters optimized for each individual datapoint. Flow: Using a flexible approximate distribution.

Amortization: solution

Term	Definition	VAE Formulation
Inference	$\log p(x) - \mathcal{L}[q]$	$\mathrm{KL}\left(q(z\|x)\|\|p(z\|x)\right)$
Approximation	$\log p(x) - \mathcal{L}[q^*]$	$\mathrm{KL}\left(q^*(z\|x)\|\|p(z\|x)\right)$
Amortization	$\mathcal{L}[q^*] - \mathcal{L}[q]$	$\mathrm{KL}\left(q(z\|x)\|\|p(z\|x)\right) - \mathrm{KL}\left(q^*(z\|x)\|\|p(z\|x)\right)$

It appeared that making a more expressive encoder will help the problem.

C. Cremer et al. Inference Suboptimality in Variational Autoencoders, ICML-2018

Learning with Amortized Inference

1. Initialise θ and ϕ.
2. Randomly sample a data point x^i from D.
3. Compute gradients (we know how!).
4. Update θ and ϕ in a gradient direction.

VAE: results

Left: 1st epoch of VAE, center 9th epoch, right: true distribution.

VAE: results 2

Celeb dataset results.

I. Shafkat's blog

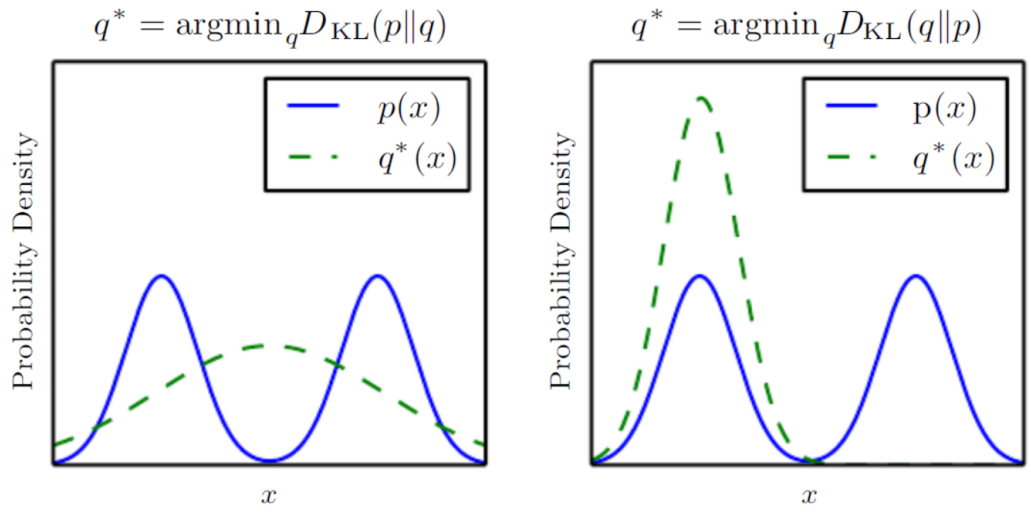
VAE: optimization problem

Stochastic optimization can get stuck:

- in the beginning, likelihood term $\log p(x|z)$ is relatively weak, such that an initially attractive state is where $q(z|x) \approx p(z)$;
- Several ideas possible:
  - optimization schedule (C. Sonderby, ICML 2016) - gradually increase weight of KL.
  - free bits (Knigma, ICML 2016) - ensure that on average, a certain minimum number of bits of information are encoded per latent variable.

VAE: blurred problem

Since optimising ELBO is equivalent to minimizing $KL(q_D(x, z; \phi)||p(x, z; \theta))$.



This can be countered by choosing a sufficiently flexible inference model, and/or a sufficiently flexible generative model.

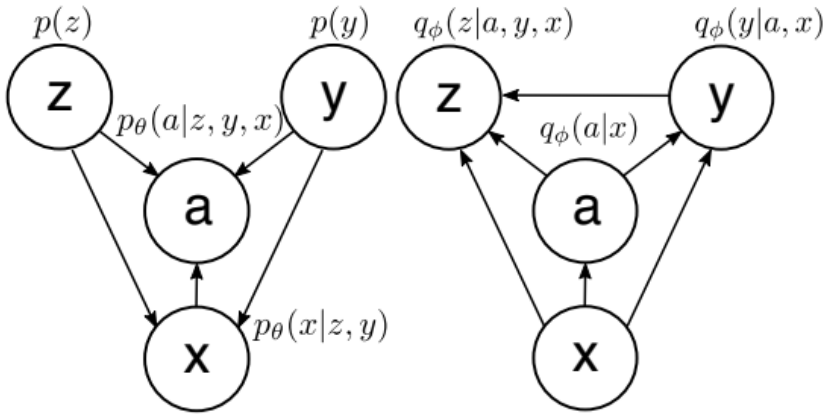
Applications

- Representation learning: learning better representations of the data. Some uses of this are:
  - Data-efficient learning, such as semi-supervised learning
  - Visualisation of data as low-dimensional manifolds
- Artificial creativity: plausible interpolation between data and extrapolation from data.

D. Knigma, M. Welling An Introduction to Variational Autoencoders, Foundations and Trends in Machine Learning

Combination with representation

One can augment the semi-supervised classification by introducing auxiliary variable a both in generative and discriminative model



(a) Generative model P. (b) Inference model Q.

Figure 1. Probabilistic graphical model of the ADGM for semi-supervised learning. The incoming joint connections to each variable are deep neural networks with parameters θ and ϕ.

This gives a higher exclusivity to discriminative model with a possibility to learn with smaller amount of train data.

L. Maaloe Auxiliary Deep Generative Models, ICML 2016

VAE: artificial creativity

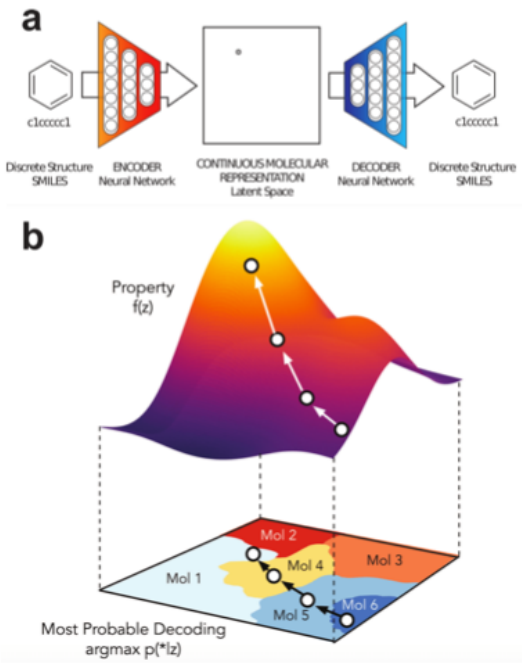


Figure 4.2: (a) Application of a VAE to chemical design in (Gómez-Bombarelli *et al.*, 2018). A latent continuous representation $\mathbf{z}$ of molecules is learned on a large dataset of molecules. (b) This continuous representation enables gradient-based search of new molecules that maximizes $f(\mathbf{z})$, a certain desired property.

- interested in latent space;
- look for the representation that has the best property $f(z)$;
- generates a new molecule.

R. Gomez-Bombarelli et al., Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules

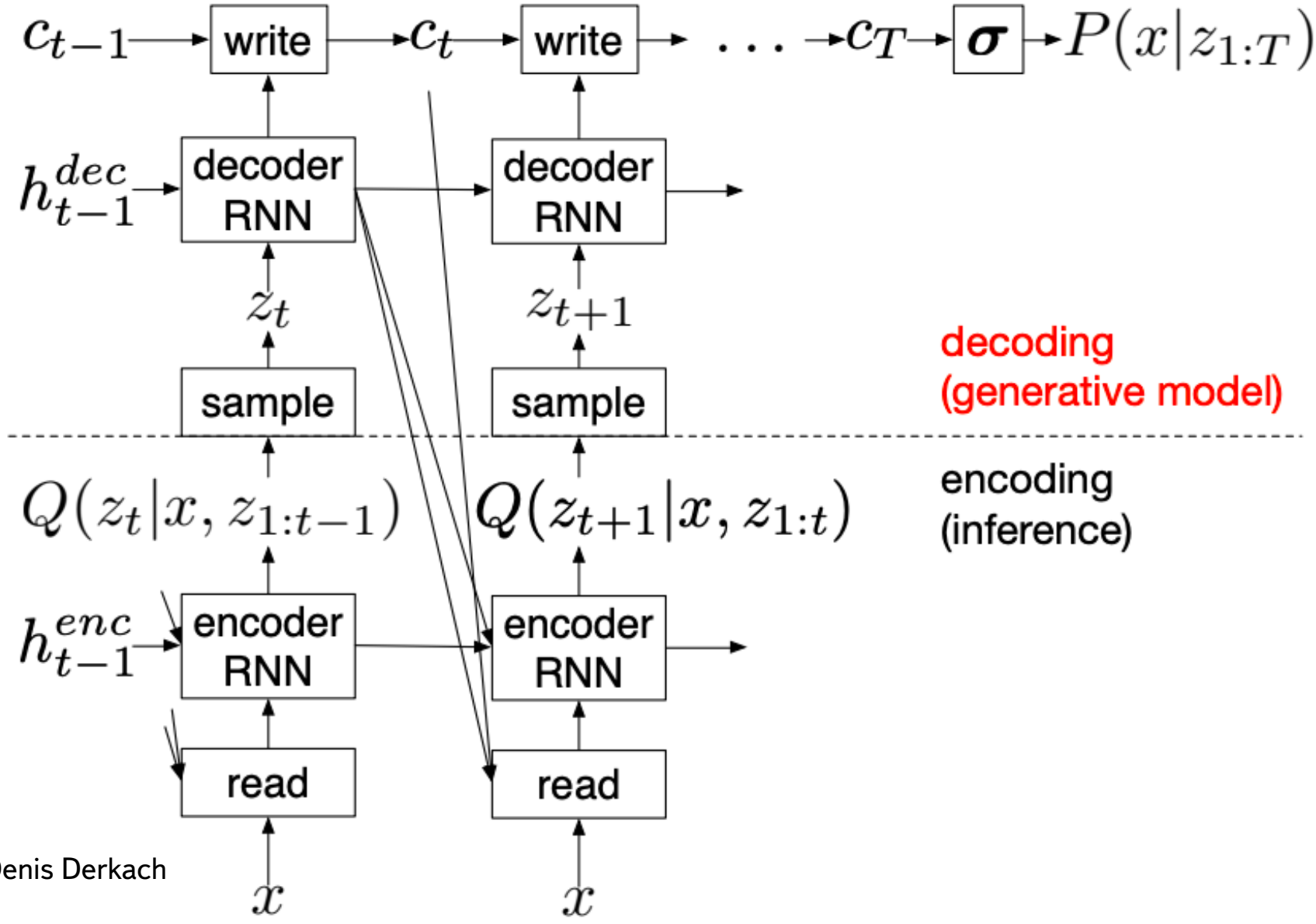
Deep Recurrent Attentive Writer (DRAW)

Idea: if you look at the human artists, they (almost) never produce the painting at once. Rather, it is a successive small steps.
Let's create a network that does the same:

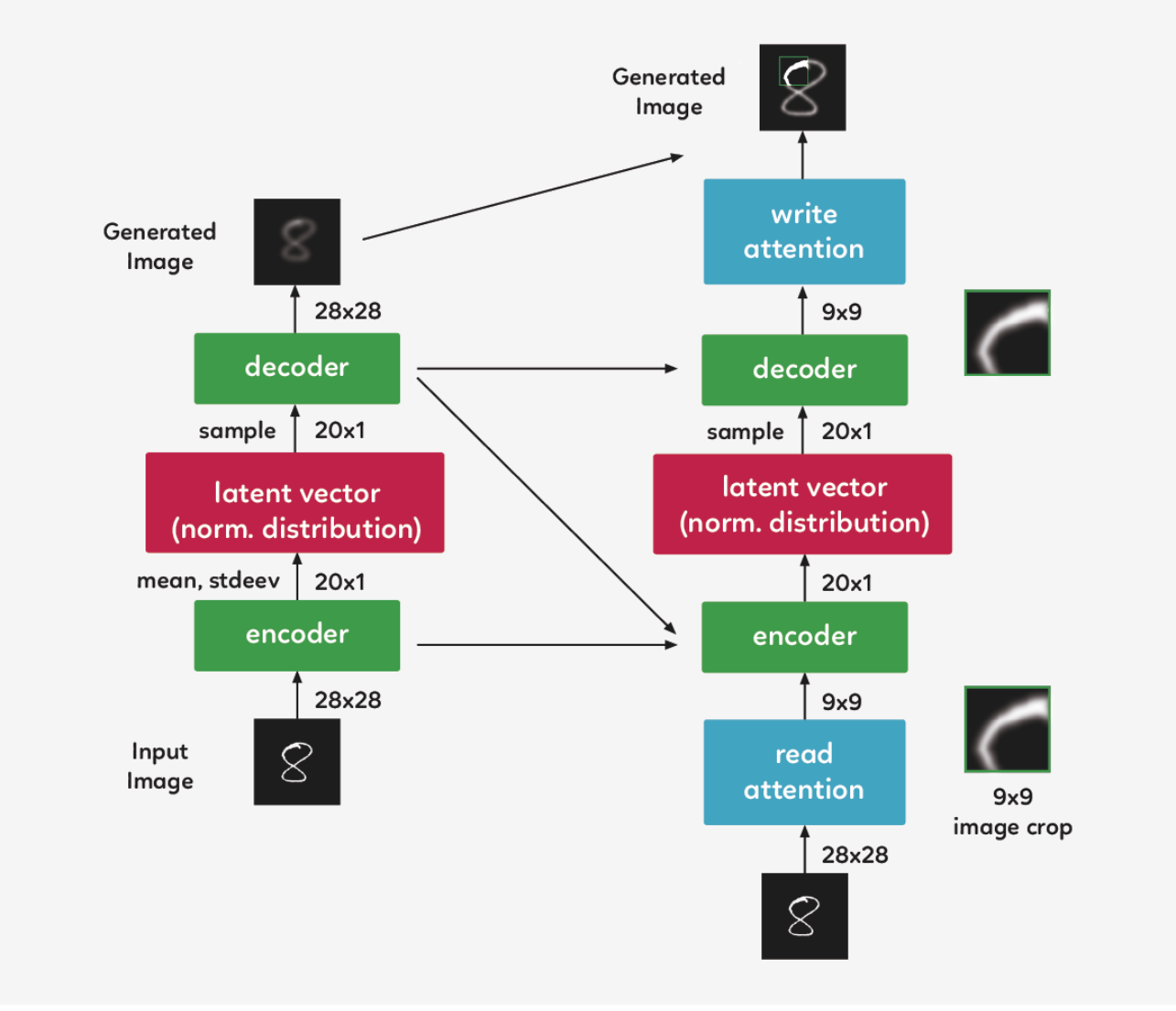
- Recurrent (both encoders and decoders).
- Decoder's output are added successively.
- A dynamically updated attention mechanism for input and output.

K. Gregor et al., DRAW: A Recurrent Neural Network For Image Generation

DRAW-Architecture



Choose important portion of an image



DRAW: results

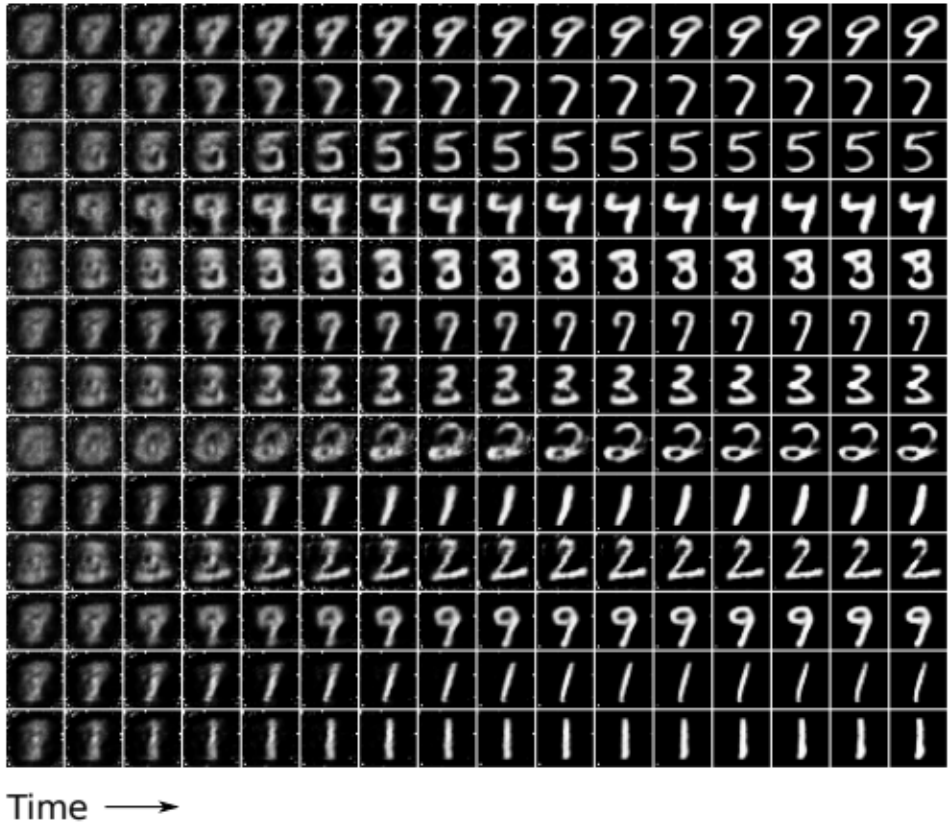
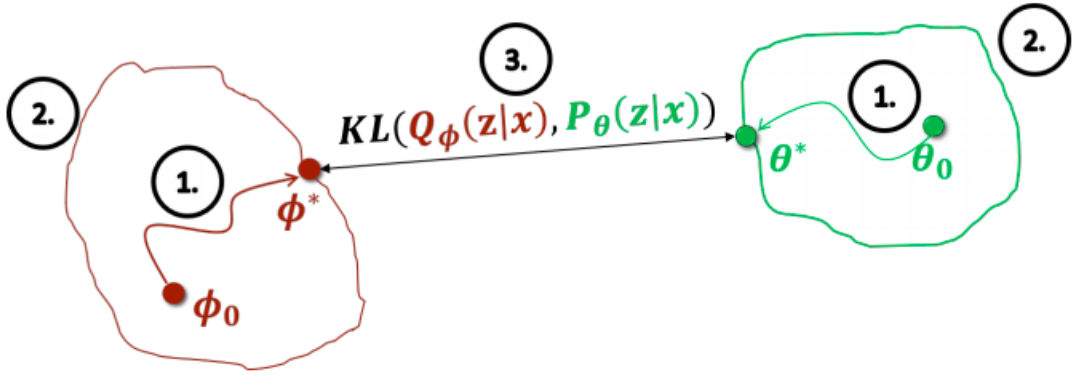


Figure 7. **MNIST generation sequences for DRAW without attention.** Notice how the network first generates a very blurry image that is subsequently refined.

DRAW: results

Figure 8. **Generated MNIST images with two digits.**

Research directions



1. Better optimization techniques
2. More expressive approximating families
3. Alternate loss functions

from Stanford S. Ermon and A. Grover Write-up

Improving Encoder

- Amortization
  - Scalability: Efficient learning and inference on massive datasets
  - Regularization effect: Because of joint training, it also implicitly regularizes the decoder (Shu et al., 2018)
- Augmenting variational posteriors
  - Monte Carlo methods: Importance Sampling (Burda et al., 2015), MCMC (Salimans et al., 2015, Hoffman, 2017, Levy et al., 2018), Sequential Monte Carlo (Maddison et al., 2017, Le et al., 2018, Naesseth et al., 2018), Rejection Sampling (Grover et al., 2018)
  - Normalizing flows (Next lectures)

Improving Decoder

- Powerful decoders $p(x|z;\theta)$ such as DRAW (Gregor et al., 2015), PixelCNN (Gulrajani et al., 2016)
- Parameterized, learned priors $p(z;\theta)$ (Nalusnick et al., 2016, Tomczak and Welling, 2018, Graves et al., 2018)

Variational objectives

- Tighter ELBO does not imply:
  - Better samples: Sample quality and likelihoods are uncorrelated (Theis et al., 2016)
  - Informative latent codes: Powerful decoders can ignore latent codes due to tradeoff in minimizing reconstruction error vs. KL prior penalty (Bowman et al., 2015, Chen et al., 2016, Zhao et al., 2017, Alemi et al., 2018)
- Alternatives to the reverse-KL divergence:
  - Renyis alpha-divergences (Li and Turner, 2016)
  - Integral probability metrics such as maximum mean discrepancy, Wasserstein distance (Dziugaite et al., 2015; Zhao et. al, 2017; Tolstikhin et al., 2018)

Wrap-up

- Variational Autoencoder is a powerful modern tool.
- Latent representations can be used as a separate instrument.
- Many good results.
- Still some things are to be improved.