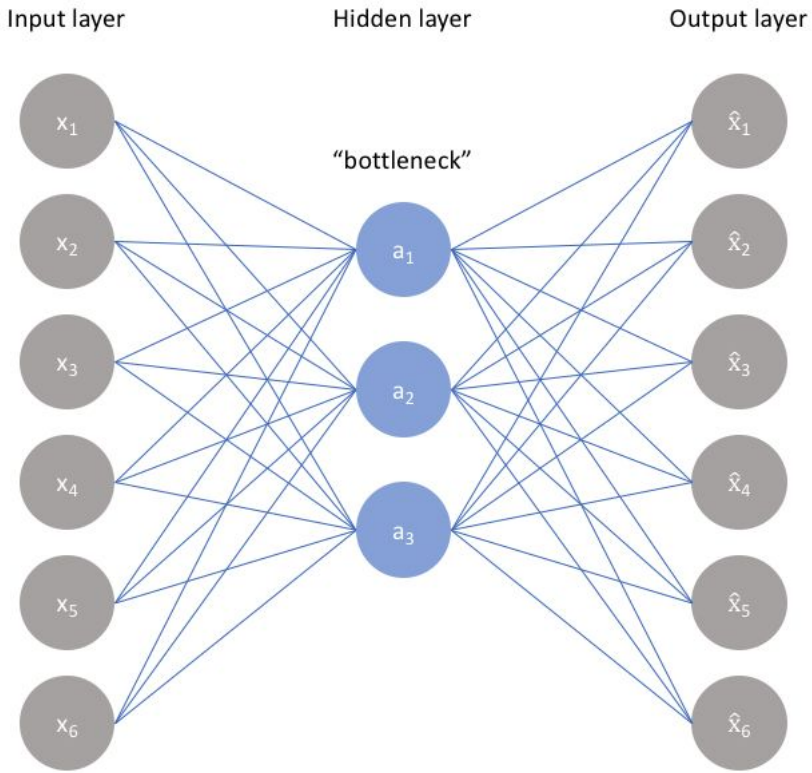
Auto-Encoding Variational Bayes

Diederik P. Kingma, Max Welling

Presented by Jack Nugent, Sol Park

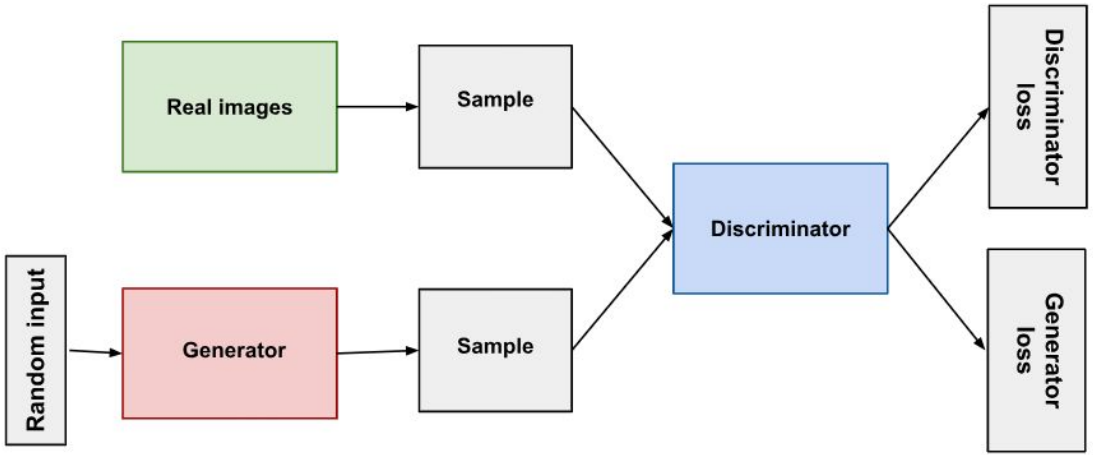
Outline

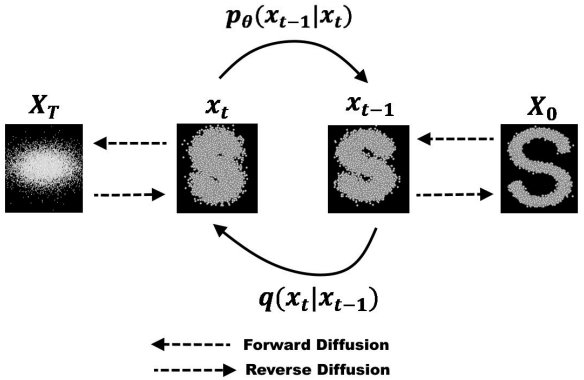
1. Preliminaries
2. Background
3. Motivation
4. ELBO
5. SGVB + AEVB
6. Reparameterization Trick
7. VAE Architecture
8. Experiments
9. VAE Developments
10. Discussion



Preliminaries

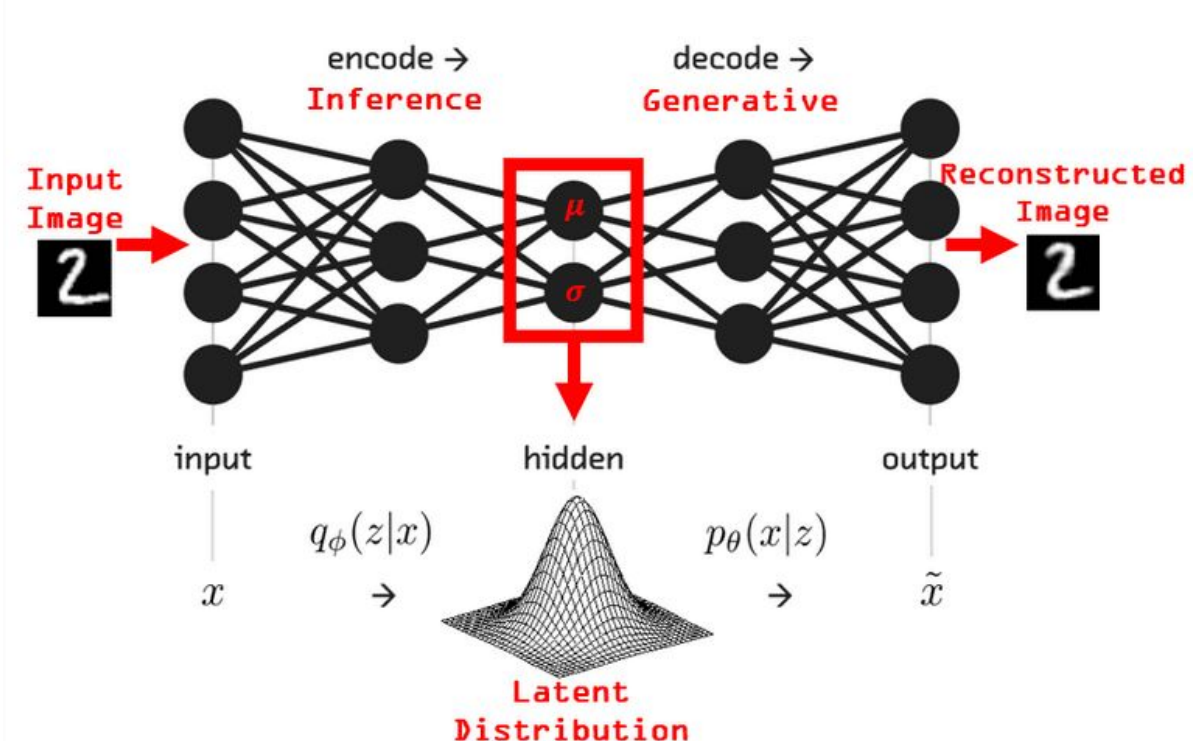
Generative Models





GANs and Diffusion models both “generate” new examples given data.

What is a variational autoencoder?



Food for thought: *Should* a VAE predict σ?

Problem Setup

The Model

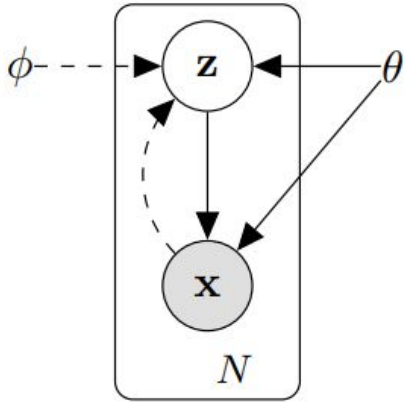


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. The variational parameters $\boldsymbol{\phi}$ are learned jointly with the generative model parameters $\boldsymbol{\theta}$.

Problem A: Intractability

$$X = \{x^i\}_{i=1}^N$$

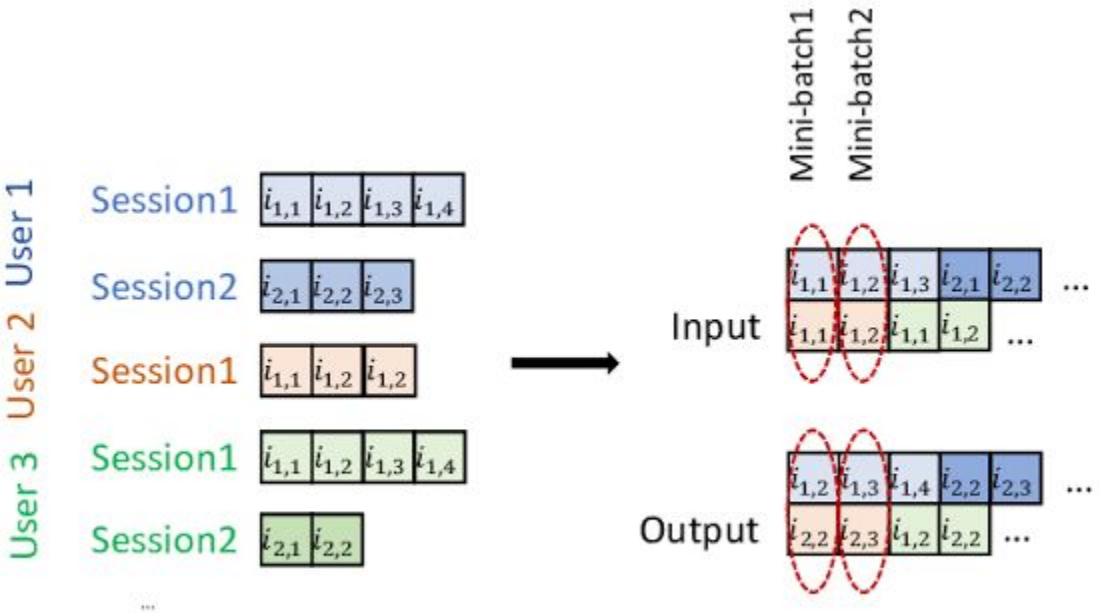
$p_\theta(z)$: continuous (high-dimensional) random var. z

$p_\theta(x|z)$: image generated from latent variables z.

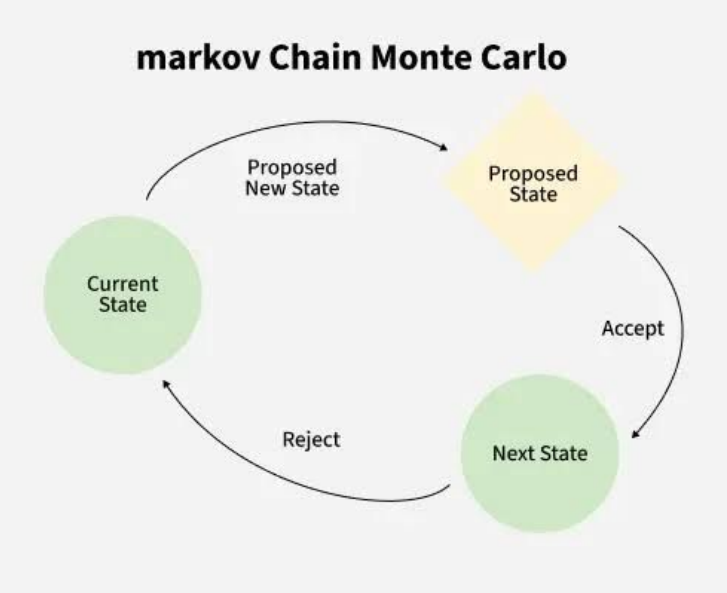
$$p_\theta(x) = \int p_\theta(z) \cdot p_\theta(x|z) dz \qquad p_\theta(z|x) = \frac{p_\theta(x|z) p_\theta(z)}{p_\theta(x)}$$

$p_\theta(x)$ is intractable, and by extension, the posterior is also intractable.

Problem B: Large Datasets



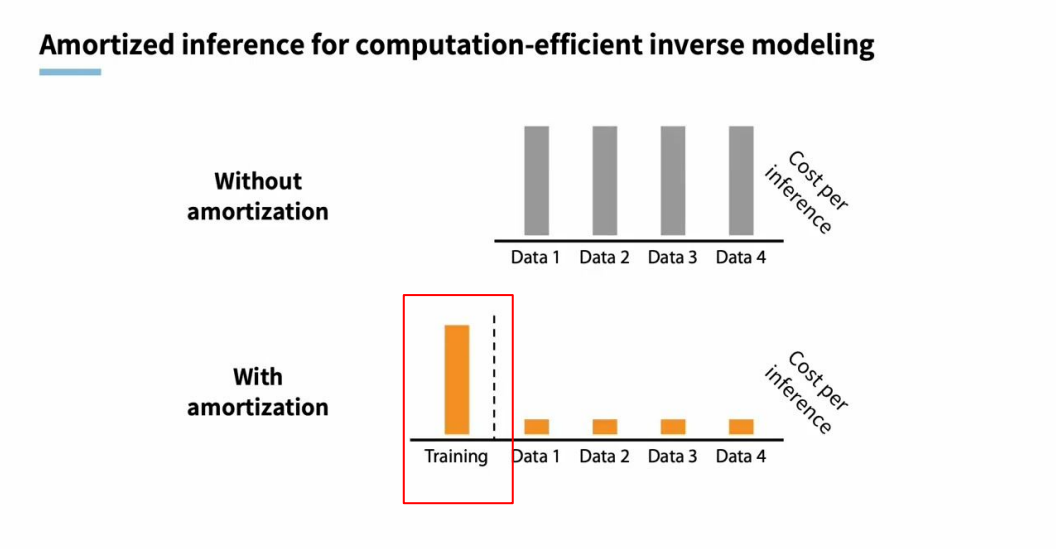
Issues with existing solutions



Expensive inference to sample from the posterior

AEVB: Auto-Encoding Variational Bayes

By using a SGVB (Stochastic Gradient Variational Bayes) estimator to optimize a recognition model (neural network).



Step One: Variational Bound (ELBO)

Recognition Model

Encoder

Our recognition model:

$$q_\varphi(z|x)$$

The parameters φ will be learned with the global parameters θ.

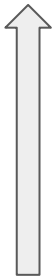
Decoder

$$p_\theta(x|z)$$

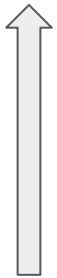
The latent variables are used to generate an image.

Marginal Likelihood Estimator

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)}))$$



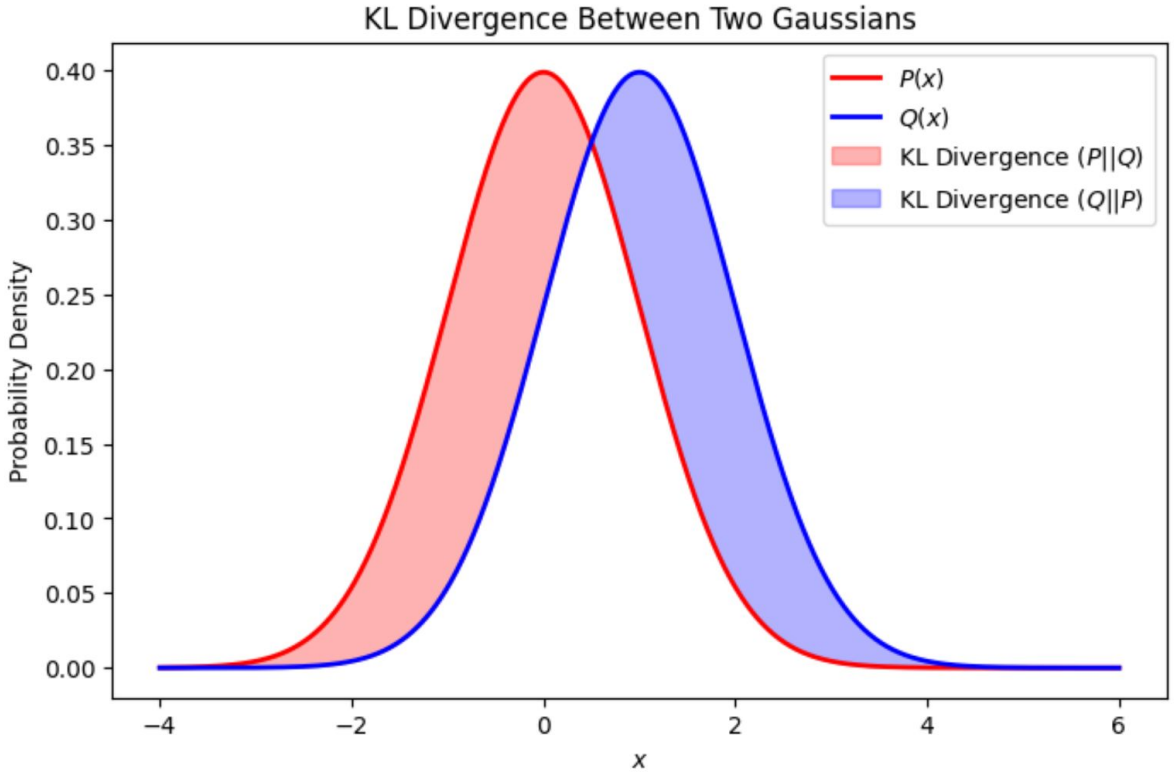
Marginal Likelihood



KL Divergence

What function does the KL divergence serve?

KL Divergence



Marginal Likelihood Estimator

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = \square + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

Marginal Likelihood

ELBO

Evidence (Variational) Lower Bound (ELBO)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right]$$

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

Regularizer

Measures how well the decoder reconstructs x.

Marginal Likelihood

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

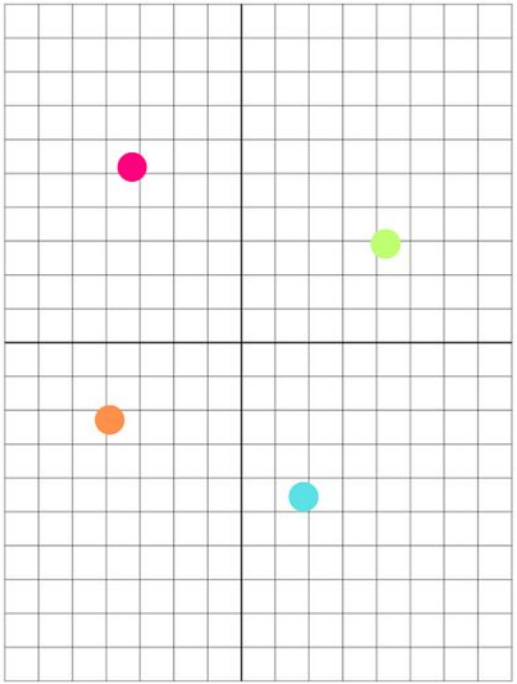
D_{KL} will always be greater than or equal to 0.

Thus, we can prove that:

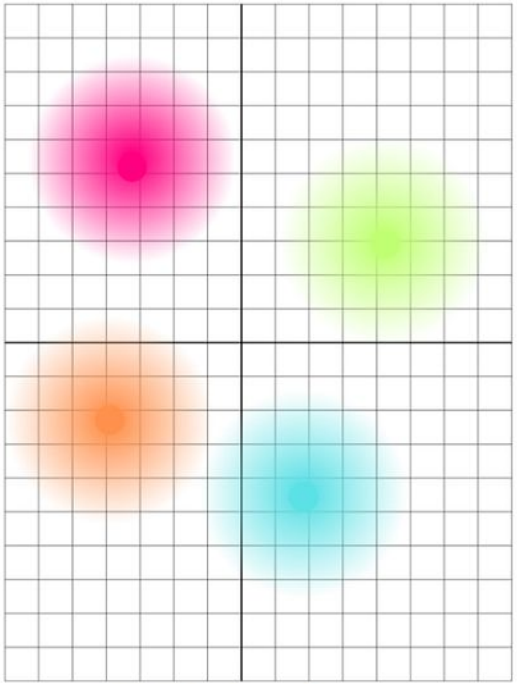
$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

KL Divergence allows for a smooth latent space (z). Why is this important?

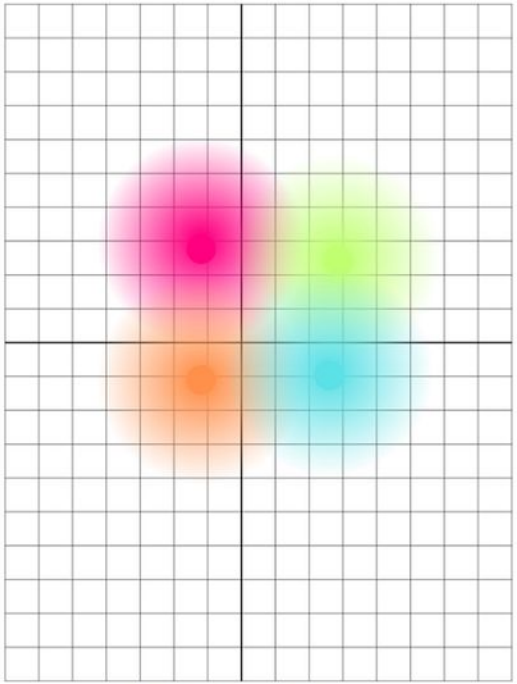
Significance of ELBO



Autoencoder Latent Space



VAE Latent Space
(Unregularised)



VAE Latent Space
(Regularised)

SGVB estimator / AEVB Algorithm

Path to SGVB Estimator

Turns out, we could simplify this the ELBO by simplifying z:

$$\widetilde{\mathbf{z}} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$

Noise

which utilizes the reparameterization trick.

SGVB Estimator

$$\widetilde{\mathcal{L}}^{A}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)} | \mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

Joint probability of image (datapoint) $x^{(i)}$ and latent variable $z^{(i, l)}$.

Likelihood of accurate generation of x + likelihood of z.

SGVB Estimator

$$\widetilde{\mathcal{L}}^{A}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)} | \mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

L samples of z.

We are sampling z from g_{φ}, not from q_{φ}.

Mini-Batch

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \simeq \widetilde{\mathcal{L}}^{M}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^{M}) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

N data points, M mini-batches

AEVB (Auto-encoding Variational Bayes) Algorithm

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

Reparameterization Trick

The problem

$$\mathcal{L}(\theta, \phi; x) = -D_{\mathrm{KL}}\big(q_\phi(z|x) \parallel p_\theta(z)\big) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$$

∇_θ: sampling has no dependency on θ

∇_ϕ: we need to backpropagate through $z \sim q_\phi$. Sampling is not differentiable.

In general, how do we compute $\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)]$?

Try to differentiate exactly

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z) f(z)\, dz$$

Try to differentiate exactly

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z) f(z)\, dz$$

$$= \int \nabla_\phi q_\phi(z)\, f(z)\, dz$$

Problem: ∇q is not a density — nothing to sample from

(Aside) REINFORCE estimator

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z) f(z)\, dz$$

$$= \int \nabla_\phi q_\phi(z)\, f(z)\, dz$$

$$= \int q_\phi(z)\, \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}\, f(z)\, dz$$

multiply and divide by q_ϕ

(Aside) REINFORCE estimator

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z) f(z)\, dz$$

$$= \int \nabla_\phi q_\phi(z)\, f(z)\, dz$$

$$= \int q_\phi(z)\, \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}\, f(z)\, dz$$

$$= \mathbb{E}_{q_\phi(z)}\left[f(z)\, \nabla_\phi \log q_\phi(z)\right]$$

Unbiased, but variance is very large

Reparameterization Trick

$$z = \mu_\phi + \sigma_\phi \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

Randomness is now a constant not dependent on ϕ

Reparameterization Trick

$$z = \mu_\phi + \sigma_\phi \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

$$\mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{p(\varepsilon)}\left[f\left(\mu_\phi + \sigma_\phi \odot \varepsilon\right)\right]$$

probability distribution $p(\varepsilon)$ does not depend on ϕ

Reparameterization Trick

$$z = \mu_\phi + \sigma_\phi \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

$$\mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{p(\varepsilon)}\left[f\left(\mu_\phi + \sigma_\phi \odot \varepsilon\right)\right]$$

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{p(\varepsilon)}\left[\nabla_\phi f\left(\mu_\phi + \sigma_\phi \odot \varepsilon\right)\right]$$

We can differentiate as usual.

General Approach

Write $z = g_\phi(\varepsilon, x)$ with $\varepsilon \sim p(\varepsilon)$ (not dependent on ϕ)

$$\mathbb{E}_{q_\phi(z|x)}[f(z)] = \mathbb{E}_{p(\varepsilon)}\left[f\left(g_\phi(\varepsilon, x)\right)\right]$$

This can be used for many distributions – Gumbel-softmax is a common reparametrization of a discrete choice

Variational Autoencoder

Variational Autoencoder

$$\mathcal{L}(\theta, \phi; x) = \underbrace{-D_{\mathrm{KL}}\big(q_\phi(z|x) \parallel p(z)\big)}_{\text{regularizer}} + \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{reconstruction}}$$

- KL regularizer $\quad - x \overset{\text{enc}}{\to} z$
  - pull the encoder's output distribution towards a gaussian
- Reconstruction $\quad - x \overset{\text{enc}}{\to} z \quad \overset{\text{sample}}{\to} \quad \tilde{z} \overset{\text{dec}}{\to} \tilde{x}$
  - How likely was x?
  - similar to an autoencoder

VAE model

	distribution	network outputs
prior $p(z)$	$\mathcal{N}(0, I)$	(no parameters)
encoder $q_\phi(z\|x)$	$\mathcal{N}(\mu, \sigma^2 I)$	$\mu(x)$ and $\log \sigma^2(x)$
decoder, binary x	Bernoulli	$y(z)$
decoder, continuous x	$\mathcal{N}(\mu, \sigma^2 I)$	$\mu(z)$ and $\log \sigma^2(z)$

- for continuous outputs, they decode a mean and deviation – many implementations instead fix the decoder $\sigma = 1$

KL term: encoder tries to match the prior

$$-D_{\mathrm{KL}} = \mathbb{E}_q[\log p(z) - \log q(z)]$$

Both are Gaussian; reduces to

$$-D_{\mathrm{KL}} = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2\right)$$

Maximized with $\mu = 0$ and $\sigma = 1$ (the prior)

Reconstruction term: decoder tries to recover the data

The decoder is $\mathcal{N}(\hat{x}(z), \sigma_x^2 I)$, so

$$\log p_\theta(x|z) = \log \prod_{i=1}^{D} \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{(x_i - \hat{x}_i)^2}{2\sigma_x^2}\right)$$

Reconstruction term: decoder tries to recover the data

The decoder is $\mathcal{N}(\hat{x}(z), \sigma_x^2 I)$, so

$$\log p_\theta(x|z) = \log \prod_{i=1}^{D} \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{(x_i - \hat{x}_i)^2}{2\sigma_x^2}\right)$$

$$= -\frac{1}{2\sigma_x^2} \sum_{i=1}^{D} (x_i - \hat{x}_i)^2 - \frac{D}{2} \log(2\pi\sigma_x^2)$$

$$= -\frac{1}{2\sigma_x^2} \|x - \hat{x}(z)\|^2 - \frac{D}{2} \log(2\pi\sigma_x^2)$$

Experiments

Wake-sleep Algorithm (Hinton)

Wake phase

- infer $z \sim q_\phi(z|x)$ from real data
- update the decoder $p_\theta(x|z)$

Sleep phase

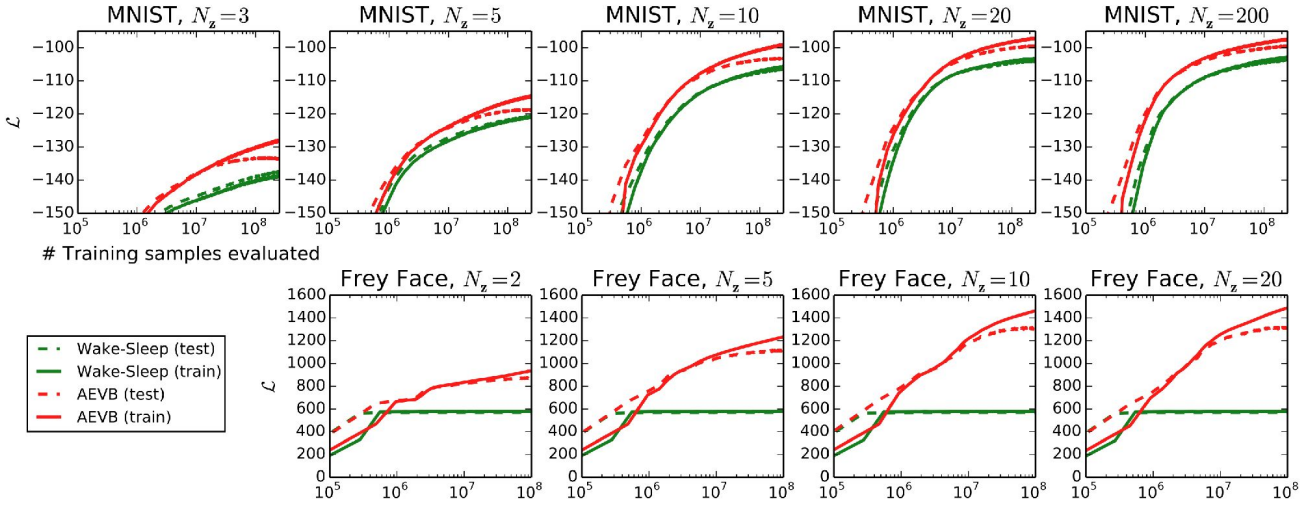
- select $z \sim N(0, 1)$ and generate x with the decoder
- train the encoder to recover z from x

Key difference

- alternates between two objectives; it does not directly optimize the ELBO
- same cost per datapoint as VAE

Generated MNIST samples

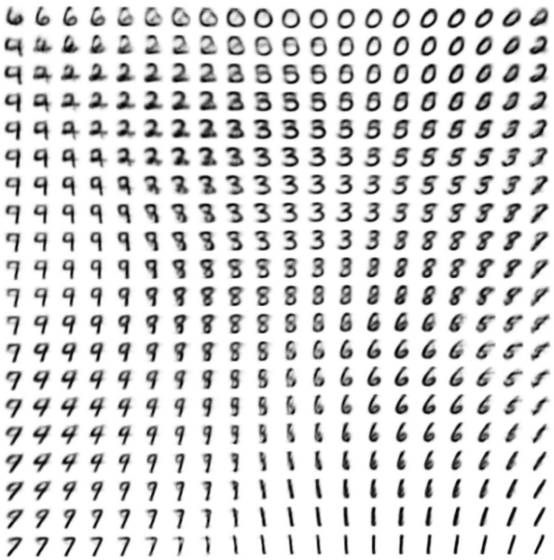
Variational Lower bound



- Converges well and scales with latent dimension

What is the structure of the latent space?

(a) Learned Frey Face manifold



(b) Learned MNIST manifold

2-D latent space, decoded on a grid of points transformed through the inverse Gaussian CDF

Computational cost

- The encoder and decoder were each an MLP with a single hidden layer of dimension 500, 200, 100 depending on dataset

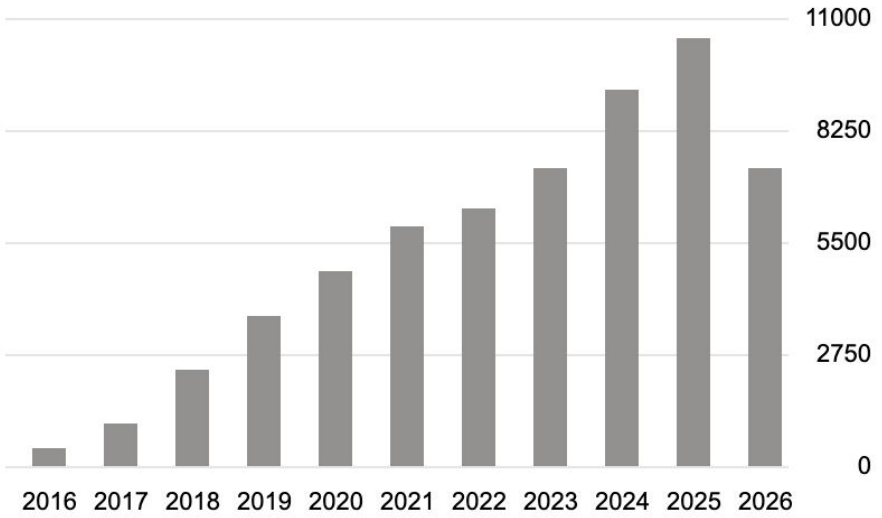
	throughput	one 10^8-sample run
Xeon (2013)	40 GFLOP/s	33–67 h
H100 (2022)	989 TFLOP/s	5–10 s

- one H100 can do $25000\times$ computation as their CPU

VAE Developments

Impact

- Nearly 60k citations



Impact

- Nearly 60k citations

Influential papers using VAE

- Neural Discrete Representation Learı
  (VQVAE)

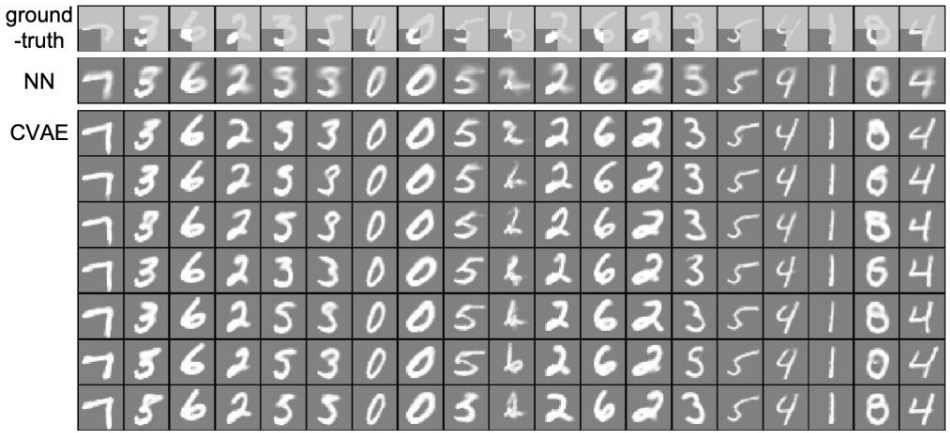
Random ImageNet samples

Impact

- Nearly 60k citations

Influential papers using VAE

- Neural Discrete Representation Learning (VQVAE)
- Learning Structured Output Representation using Deep Conditional Generative Models (CVAE)



Conditional generation
(image infilling)

Impact

- Nearly 60k citations

Influential papers using VAE

- Neural Discrete Representation Learning (VQVAE)
- Learning Structured Output Representation using Deep Conditional Generative Models (CVAE)
- High-Resolution Image Synthesis with Latent Diffusion Models

ImageNet Samples

β-VAE

Optimize reconstruction with fixed prior error:

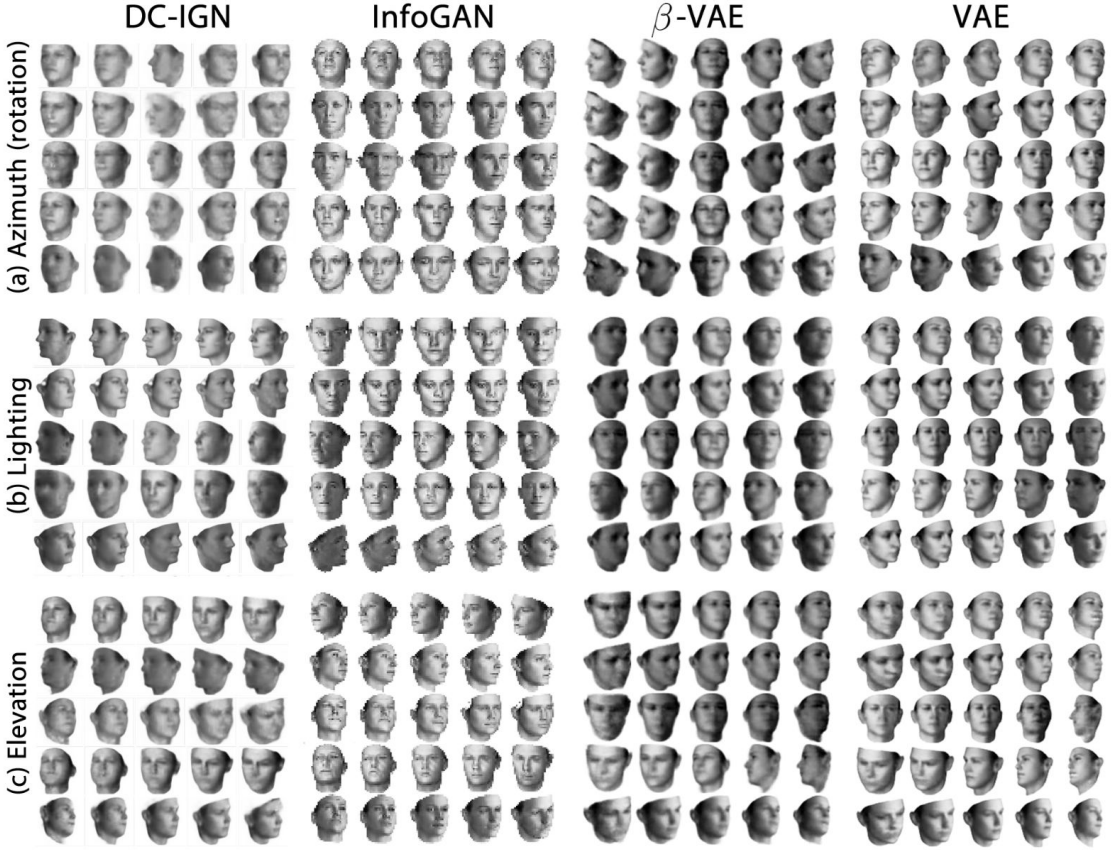
$$\max_{\theta,\phi} \quad \mathbb{E}_{q_{\phi(z|x)}}\left[\log p_{\theta(x|z)}\right] \quad \text{subject to} \quad D_{\mathrm{KL}}\left(q_{\phi(z|x)} \parallel p(z)\right) \leq \varepsilon$$

Rewrite with “Lagrange multiplier”

$$\mathcal{L}_\beta = \mathbb{E}_q\left[\log p_{\theta(x|z)}\right] - \beta\, D_{\mathrm{KL}}\left(q_{\phi(z|x)} \parallel p(z)\right)$$

- β controls how strongly we enforce adherence to the prior
- authors claim higher β enables natural emergence of structure

β-VAE – model learns identifiable latent factors



Diffusion — a hierarchical VAE

q: fixed noising (encoder) p_θ: learned denoising (decoder)

component	VAE	diffusion
prior	$p(z) = \mathcal{N}(0, I)$	$p(x_T) = \mathcal{N}(0, I)$, then $p_\theta(x_{t-1}\|x_t)$
encoder	$q_{\phi(z\|x)}$, learned	$q(x_{1:T}\|x_0)$, fixed noising
decoder	$p_{\theta(x\|z)}$	$p_\theta(x_{0:T-1}\|x_T)$, learned denoising

- replace latent z with a series of latents $x_1, x_2, ..., x_T$

Discussion Questions

Discussion Questions

1. What is the generative model assumed by a VAE, and why is maximum-likelihood learning of this model difficult?
2. What is the ELBO? Why do they optimize this?
3. What is the reparameterization trick, and why is it necessary?
4. What assumptions does the VAE make about its latent representation, and when might those assumptions be problematic for scientific data?

Discussion Questions

1. Do VAEs learn latent spaces with semantic meaning?
2. What determines nearness in VAE latent space?
3. Should a VAE decoder predict μ or μ and σ? What are the advantages/disadvantages of each?
4. There are various unsupervised learning methods – VAEs, masked autoencoding, momentum contrastive learning, JEPA, SIGREG. Alphafold uses a form of masked autoencoding. Given unlabeled data, how would you decide what method to use?
5. Is reconstruction in pixel space a good measure of how well you can represent data?