

# Variational Autoencoders

CPSC 540 - Advanced Machine Learning  
University of British Columbia

April 27, 2022

## Ordinary autoencoders

- Convolutional layers reduce input data  $x$  into **latent vector** representation  $z$ .
- Generate new data: pick a point  $z$  in **latent space**, push through **decoder**.
- **Potential for overfitting.**
- Nearby points in latent space can **produce different results.**

## Variational Autoencoders (VAEs)

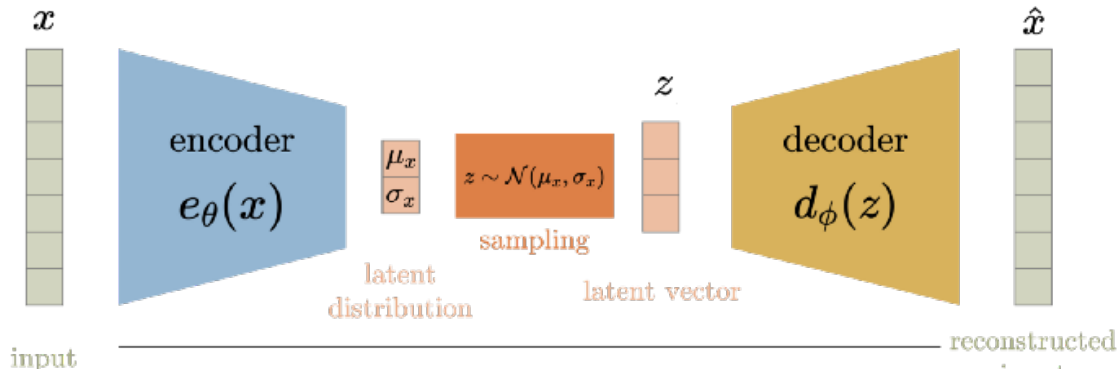
- Represent the latent variable  $z$  as a *distribution*

$$z \sim p(z \mid x)$$

- Draw a random sample  $z$ , then push it through the decoder.
- Nearby points in latent space **produce similar data** when pushed through decoder.

# Model Definition

- Similar to autoencoders, VAEs have an **encoder**, **decoder**, and **bottleneck layer**.
- The latent vector  $z$  is not learned directly.
- Instead the **parameters of the distribution** are learned.



- VAEs are **deep generative models**.
- Use evidence maximization, i.e. maximizing  $p(x)$ .
- Evidence can be calculated with

$$p(x) = \int p(x | z)p(z) dz$$

where the integral is over the possible values of  $z$  in the latent space.

- Main Problem: **Exponential time to calculate  $p(x)$**  (must evaluate all configurations of latent variables).

## Simplifying Assumptions:

- Decoder (likelihood):

$$p_{\theta}(x \mid z) = \mathcal{N}(f(z, \theta), \sigma^2 I)$$

Function  $f$  to be learned by decoder network weights/biases  $\theta$ .  $\sigma$  is hyper-parameter.

- Encoder:  $p(z \mid X)$  can be approximated by

$$q_{\phi}(z \mid x) = \mathcal{N}(\mu(x, \phi), \Sigma(x, \phi))$$

Weights/biases  $\phi$  are learned by encoder network.

- Latent Prior:  $p(z) = \mathcal{N}(0, 1)$ .

**Q:** How can we force  $p(z | x) \approx q_\phi(z | x)$ ?

**A:** The **Kullback-Leibler** (KL) - divergence **measures how different** two distributions  $p_1$  and  $p_2$  are:

$$\mathcal{KL}(p_1 \parallel p_2) := E_{x \sim p_1}(\log(p_1(x)) - \log(p_2(x)))$$

We will try to force  $p(z | x) \approx q_\phi(z | x)$  by minimizing the KL-divergence between them.

# Loss Function

---

Want to learn best  $\mu$  and  $\Sigma$  for the encoder distribution  $q_\phi(z | x^i) = \mathcal{N}(\mu(x^i, \phi), \Sigma(x^i, \phi))$ :

$$q_\phi^*(z | x) = \operatorname{argmin}_\phi \mathcal{KL}(q_\phi(z | x) || p(z | x))$$

After some manipulation (to be done on the assignment):

$$\mathcal{KL}(q_\phi(z | x) || p(z | x)) = E_{z \sim q_\phi(z | x)}[\log(q_\phi(z | x)) - \log(p(x, y))] + \log(p(x))$$

The log-evidence to be maximized is therefore

$$\log(p(x)) = E_{z \sim q_\phi(z | x)}[\log(p(x, z)) - \log(q_\phi(z | x))] + \mathcal{KL}(q_\phi(z | x) || p(z | x))$$



(Continued) Log-evidence to be **maximized**:

$$\log(p(x)) = E_{z \sim q_\phi(z | x)}[\log(p(x, z)) - \log(q_\phi(z | x))] + \mathcal{KL}(q_\phi(z | x) || p(z | x))$$

- Expectation on RHS is called “**ELBO**” function.
- Issue:  $p(z | x)$  still appears! (Hard to calculate).

ELBO can be manipulated (to be done on assignment) so that  $\text{grep}(z | x)$  goes away:

$$\text{ELBO}_i(\theta, \phi) = E_{z \sim q_\phi(z | x^i)}[\log(p_\theta(x^i | z))] - \mathcal{KL}(q_\phi(z | x^i) || p(z))$$

The **negative of this is the loss** for a single training example  $i$ .

- “**E**vidence **L**ower **B**ound” - Satisfies  $\text{ELBO}(x) \leq \log(p(x))$ .
- Instead of maximizing  $\log(p(x))$  directly like in MLE, we maximize its lower bound ELBO.
- Can be viewed as *modified* or *approximate* “MLE”.

# Interpretation of the Loss Function

---

Loss function:

$$-\text{ELBO}_i(\theta, \phi) = -E_{z \sim q_\phi(z | x^i)}[\log(p_\theta(x^i | z))] + \mathcal{KL}(q_\phi(z | x^i) || p(z))$$

## Interpretation:

- First term: **reconstruction error** (minimize NLL expected under our distribution  $q_\phi$ ).
- Second term: **regularization** (minimize KL-divergence).

# The Role of Regularization



Figure: Latent distributions. Left: no regularization. Right: regularization. Source: <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>

- Each  $x^i$  gets its own Gaussian distribution  $q_\phi(z | x^i)$ .
- On left: learned Gaussians are sharp and far apart from each other.
- Overfitting - Nonsense data generated from latent vectors between distributions.
- On right: broader distributions. Keeps the distributions close together.
- Latent vectors between distributions produce more realistic data.

# The Role of Regularization

---

Regularization is provided by the  $\mathcal{KL}(q_\phi(z \mid x^i) \parallel p(z))$  term.

- KL-divergence penalizes  $q_\phi(z \mid x)$  for getting too far from  $p(z) = \mathcal{N}(0, 1)$ .
- Without KL regularization, model learns Gaussians  $q_\phi(z \mid x)$  with very low variance
- Low variance Gaussian is just a “spike” at a particular point in latent space.
- This makes it practically deterministic, like an ordinary autoencoder.
- This regularization is how VAEs prevent this overfitting.

Loss function:

$$-\text{ELBO}_i(\theta, \phi) = -E_{z \sim q_\phi(z \mid x^i)}[\log(p_\theta(x^i \mid z))] + \mathcal{KL}(q_\phi(z \mid x^i) \parallel p(z))$$

How do we train? Remember our simplifying assumptions:

- $q_\phi(z \mid x^i) = \mathcal{N}(\mu(x^i, \phi), \sigma^2(x^i, \phi))$
- $p_\theta(x^i \mid z) = \mathcal{N}(f(z, \phi), \sigma^2)$
- $p(z) = \mathcal{N}(0, 1)$

Loss function:

$$-\text{ELBO}_i(\theta, \phi) = -E_{z \sim q_\phi(z | x^i)}[\log(p_\theta(x^i | z))] + \mathcal{KL}(q_\phi(z | x^i) || p(z))$$

Computing first term:

- Draw a few (or sometimes just one) samples

$$z \sim q_\phi(z | x^i)$$

- Approximate the expectation using Monte Carlo.

Loss function:

$$-\text{ELBO}_i(\theta, \phi) = -E_{z \sim q_\phi(z | x^i)}[\log(p_\theta(x^i | z))] + \mathcal{KL}(q_\phi(z | x^i) || p(z))$$

Computing second term:

- Just the KL-divergence between two Gaussians.
- Has a closed form:

$$\begin{aligned} \mathcal{KL}(\mathcal{N}(\mu_1, \Sigma_1) || \mathcal{N}(\mu_2, \Sigma_2)) \\ = \frac{1}{2} \left( \text{Tr}(\Sigma_2^{-1} \Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) - k + \log \left( \frac{|\Sigma_2|}{|\Sigma_1|} \right) \right) \end{aligned}$$



Loss function:

$$-\text{ELBO}_i(\theta, \phi) = -E_{z \sim q_\phi(z | x^i)}[\log(p_\theta(x^i | z))] + \mathcal{KL}(q_\phi(z | x^i) || p(z))$$

Problem:

- Drawing samples  $z \sim q_\phi(z | x^i)$  to estimate the expectation on RHS.
- This is **not a differentiable step**.
- **Can't backpropagate error** for SGD!

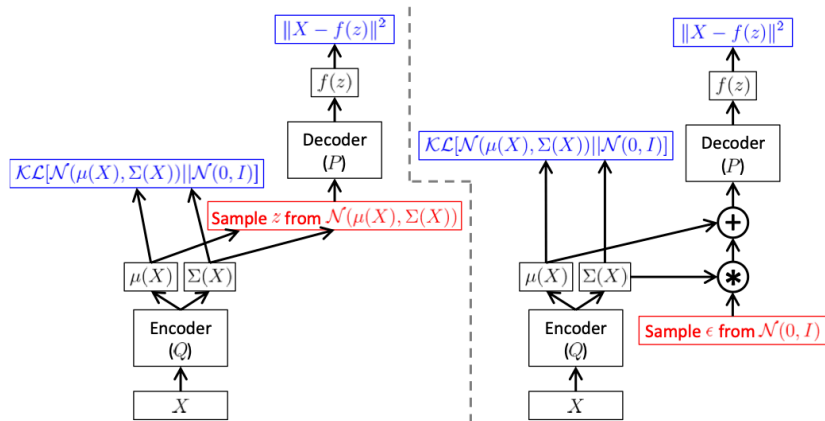
# Reparametrization Trick

---

**Solution:** Use the **reparametrization trick**:

- Draw a sample  $\varepsilon \sim \mathcal{N}(0, 1)$ .
- $z = A\varepsilon + \mu$  will be distributed as  $\mathcal{N}(\mu, A^T A)$ .
- Use Cholesky decomposition  $\Sigma = A^T A$  to get the desired  $A$ .
- This is a **differentiable step**! All the randomness comes from  $\varepsilon$ .
- $A(x^i, \phi)$  and  $\mu(x^i, \phi)$  depend on network parameters.

# Reparametrization Trick



**Figure:** Left: Without reparametrization trick. Right: with reparametrization trick. Source: <https://arxiv.org/pdf/1606.05908.pdf>

# Inference: Generating New Samples

---

- Latent space can be used to **generate new samples**.
- Goal is to generate data that looks like training, but is different.
- Simply sample a random  $z$  and push it through the decoder.
- Hopefully look like new samples thanks regularization term  $\mathcal{KL}(q_\phi(z | x) || p(z))$ .

## Inference: Generating New MNIST Samples

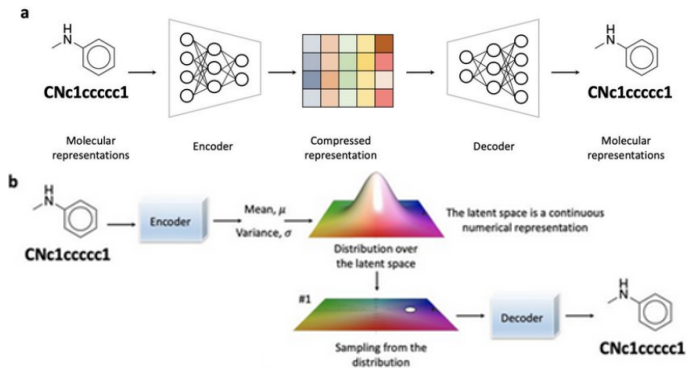
---



Figure: MNIST digits created with a VAE. Source: <https://arxiv.org/pdf/1312.6114.pdf>

- Look like **slightly blurry digits**. This is a drawback of VAEs.
- The noise from sampling  $z$  tends to produce blurry images compared to GANs.

# Inference: Generating New Chemical Samples



**Figure:** Creating new molecules similar to a given molecule. a) normal autoencoder. b) variational autoencoder. Source: [https://www.researchgate.net/figure/The-autoencoder-and-the-variational-autoencoder-a-An-autoencoder-encodes-input\\_fig3\\_43786548](https://www.researchgate.net/figure/The-autoencoder-and-the-variational-autoencoder-a-An-autoencoder-encodes-input_fig3_43786548)

---

## retina-VAE: Variationally Decoding the Spectrum of Macular Disease

---

Stephen G. Odaibo\*

(1) Department of Machine Learning Research  
RETINA-AI Health, Inc.

(2) Department of Head & Neck Surgery  
Ophthalmology Section  
MD Anderson Cancer Center  
stephen.odaibo@retina-ai.com

- Uses VAE to gain insights into macular disease.
- Generated 3,000 “patient profile vector” samples  $x^i$  using simulated clinical data based on known statistics.
- Trained a VAE with 3-dimensional latent space.
- Found 14 well-defined clusters when plotting the latent vectors.
- Classified clusters using  $k$ -means.
- Different clusters might respond better to different treatments.

## Application: Retina Disease

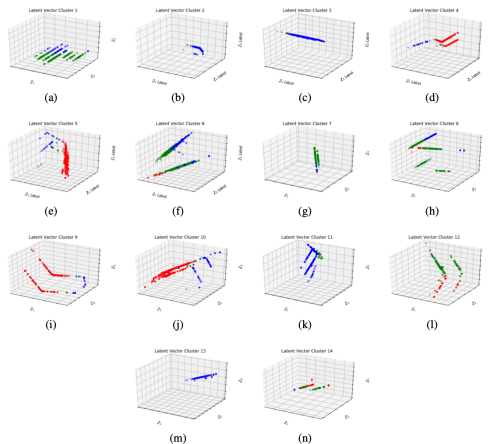
---



**Figure:** Left: normal retina. Right: macular degeneration. Source: <https://arxiv.org/pdf/1907.05195.pdf>



# Application: Retina Disease



**Figure:** 14 clusters of the 3d latent vectors sampled from  $q_{\phi}(z | x)$ . Source: <https://arxiv.org/pdf/1907.05195.pdf>

# Bayesian Variational Autoencoders

---

- BVAEs place a prior over the network parameters  $\theta$ .
- Estimate the marginal likelihood using

$$p(\tilde{x} | x) \propto \int \int p_{\theta}(\tilde{x} | z)p(z)p(\theta | x)dzd\theta$$

- There are various tricks for implementing BVAEs. See e.g. <https://arxiv.org/pdf/1912.05651.pdf>.
- Can perform better on out-of-distribution detection tasks.

## Modification: $\beta$ -VAEs

---

- $\beta$ -VAEs are a generalized version of VAEs.
- Hyperparameter  $\beta$  controls regularization strength:

$$\text{Loss} = -E_{z \sim q_\phi(z | x^i)}[\log(p_\theta(x^i | z))] + \beta \mathcal{KL}(q_\phi(z | x^i) || p(z))$$

- $\beta$ -VAEs allow for training with a special emphasis on *disentanglement*.

# Disentanglement

- Phenomenon where latent vector components are highly interpretable.
- Changing one component of a latent vector only affects one part of output data.
- Well-chosen  $\beta$  values (typically  $\beta > 1$ ) lead to **increased disentanglement**.

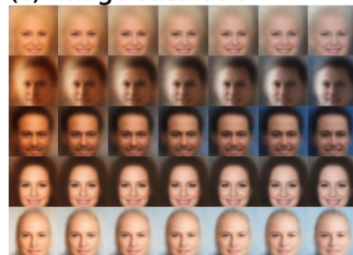
(a) Skin colour



(b) Age/gender



(c) Image saturation



**Figure:** Changes to one component of the latent vector  $z$  produce interpretable changes in the output data.

# Supervised Variational Autoencoders



(a) Row collision.



(b) Untraversable obstacle.



(c) Traversable obstacle.

**Figure:** Supervised VAE used in a self-driving toy car trained to classify terrain types. Source: <https://arxiv.org/pdf/2012.08637.pdf>

# Supervised Variational Autoencoders

---

- In supervised learning, the decoder is not used.
- Given new data  $\hat{x}$ , we **encode it as a latent distribution**  $q_\phi(z \mid \hat{x})$ .
- We can then use an output layer of our choosing.
- For example, could use a categorical estimator in final layer

$$p(y \mid x) = \text{Categorical}(y \mid \pi_c(\mu(\hat{x}, \phi), \Sigma(\hat{x}, \phi)))$$

where  $\pi_c$  is a probability vector.

- VAEs are **deep generative models**.
- Same architecture are autoencoders, but uses **latent distribution**.
- Regularization from **KL-divergence** allows for more realistic sampling.
- Use the **reparametrization** trick so that all steps are differentiable.
- With some effort, can achieve interpretability through **disentanglement**.
- Downside: the random nature of the latent vectors **tends to lead to “blurrier” samples** compared to GANs.