Latent Variable Models: Variational Autoencoders

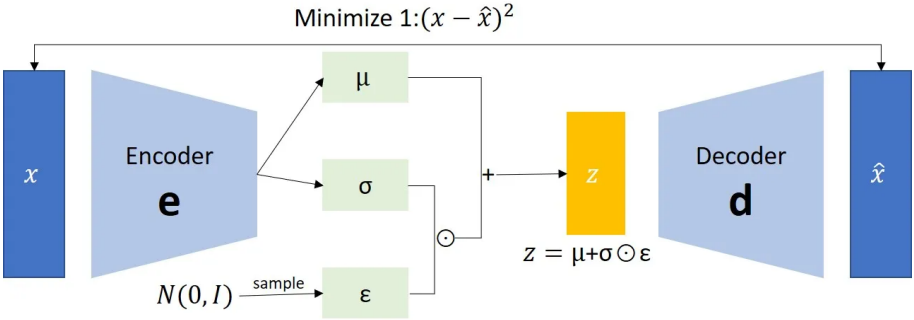
Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

June 23, 2025

Table of Contents



Minimize 2: $\frac{1}{2}\sum_{i=1}^{N}(\exp(\sigma_i) - (1+\sigma_i) + {\mu_i}^2)$

Why do we need VAEs?

- Traditional autoencoders are:
  - deterministic
  - lack generative capabilities
  - do not model uncertainty in latent space
- Need a compact, meaningful representation (latent space)
- We want to learn a probabilistic model of the data
- We want to generate new data samples similar to training data

Motivation (cont.)

Applications

- Image generation (e.g., generating new faces or handwritten digits).
- Data compression and denoising.
- Anomaly detection in various domains.

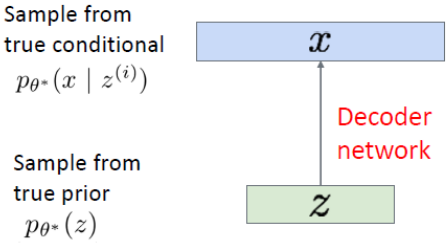
⇒ Learning Outcomes

By the end of this session, you will be able to:

- Explain what a VAE is and why we use it
- Describe how VAEs create and use hidden (latent) spaces
- Understand how VAEs learn from data
- Understand the ELBO and its role
- See how we train VAEs using special tricks
- Recognize different types of VAEs and some common challenges

VAE: **Introduction**

- We want a **generative model**, which given a prior **z** outputs a new sample from data.
- To make the latent space Z continuous, let's choose it to be Gaussian
- So now, we want to estimate the true parameters θ^* of this generative model.



- **Probabilistic Encoder:** Instead of encoding an input x to a fixed point z, the encoder learns a distribution over the latent variable z conditioned on the input:

$$q(z \mid x)$$

Typically, this is a multivariate Gaussian with parameters $\mu(x)$ and $\sigma(x)$ learned by a neural network.

- **Probabilistic Decoder:** Given a sample z from the latent distribution, the decoder reconstructs the input by modeling:

$$p(x \mid z)$$

This allows for generating new data by sampling from the latent space.

- **Latent Space:**
  - The model learns a smooth, continuous space for z.
  - Similar data points end up close together in this space.
  - This helps the model understand the main patterns in the data.

Objectives and Training: The training objective of a VAE is to maximize the **Evidence Lower Bound (ELBO)**:

$$\log p(x) \geq \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{\mathrm{KL}}(q(z|x)\|p(z))$$

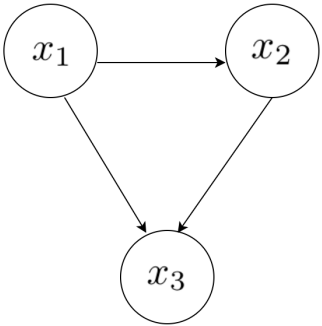
- The first term, $\mathbb{E}_{q(z|x)}[\log p(x|z)]$, encourages accurate reconstruction.
- The second term, $D_{\mathrm{KL}}(q(z|x)\|p(z))$, regularizes the latent space by making the approximate posterior $q(z|x)$ close to the prior $p(z)$, usually $\mathcal{N}(0, I)$.

Why Use VAEs?:

- Enable generative modeling: generate new samples by sampling from the latent distribution.
- Learn smooth, structured, and interpretable latent representations.
- Provide a principled probabilistic framework for inference and generation.

VAE: **Latent Variable Models**

- **Autoregressive models + Flows**: All random variables are observed.
- **Latent Variable Models (LVMs)**: Some random variables are hidden – we do not get to observe them.



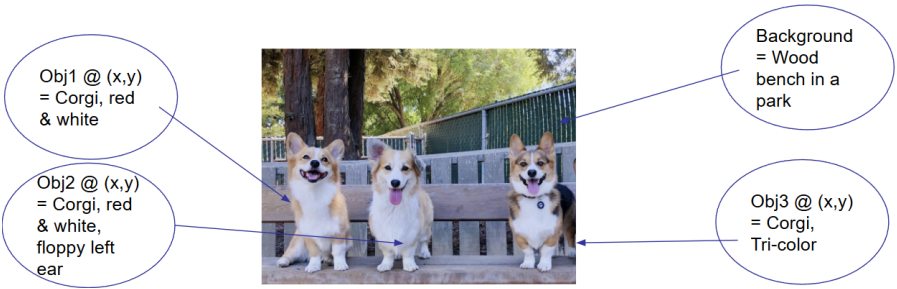
Latent Variable Models (cont.)

Latent Variable Models:

- **Definition**: Models that assume the data is generated from some unobserved (latent) variables.
- **Goal**: Learn a mapping from observed data x to latent variables z and vice versa.
- **Generative Process**:
  - Sample latent variable z from a prior distribution $p(z)$.
  - Generate data x from the latent variable using a likelihood function $p(x|z)$.
- **Inference Problem**: Given observed data x, infer the posterior distribution $p(z|x)$.
- **Challenge**: Direct computation of posterior $p(z|x)$ is often intractable, leading to the need for approximations.

Why Latent Variable Models?

- Simpler, lower-dimensional representations of data often possible
  - Latent variable models hold the promise of automatically identifying those hidden representations



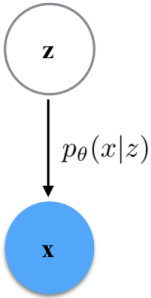
Why Latent Variable Models? (cont.)

- Autoregressive (AR) models are slow to sample because all pixels (observation dimensions) are assumed to be dependent on each other.
- We can make part of the observation space independent *conditioned on some latent variables*.
- Latent variable models *can* have faster sampling by exploiting statistical patterns.

Why Latent Variable Models? (cont.)

- Sometimes, it's possible to design a latent variable model with an understanding of the causal process that generates data
- In general, we don't know what are the latent variables and how they interact with observations
- Most popular models make little assumption about what are the latent variables
- Best way to specify latent variables is still an active area of research

- Sample
$$z \sim p_Z(z)$$
$$x \sim p_\theta(x \mid z)$$
- Evaluate likelihood $$p_\theta(x) = \sum_z p_Z(z) p_\theta(x \mid z)$$
- Train $$\max_\theta \sum_i \log p_\theta(x^{(i)}) = \sum_i \log \sum_z p_Z(z) p_\theta(x^{(i)} \mid z)$$
- Representation $$x \longrightarrow z$$



VAE: **Variational Inference**

Variational Inference

- Now, how to train this model?

Variational Inference

- Now, how to train this model?
- How about following the same strategy as in FVSBNs? Learn model parameters to maximize the likelihood of training data.

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Variational Inference

- Now, how to train this model?
- How about following the same strategy as in FVSBNs? Learn model parameters to maximize the likelihood of training data.

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

- But there is a problem here. It is Intractible to compute $p(x|z)$ for every z!

Variational Inference

- Now, how to train this model?
- How about following the same strategy as in FVSBNs? Learn model parameters to maximize the likelihood of training data.

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

- But there is a problem here. It is Intractible to compute $p(x|z)$ for every z!
- Intuitively, need to figure out which z corresponds to each x in the dataset, but such mapping is unknown.

Variational Inference

- Now, how to train this model?
- How about following the same strategy as in FVSBNs? Learn model parameters to maximize the likelihood of training data.

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

- But there is a problem here. It is Intractible to compute $p(x|z)$ for every z!
- Intuitively, need to figure out which z corresponds to each x in the dataset, but such mapping is unknown.
- This also makes posterior density $p(z|x)$ intractable because it depends on $p_\theta(x)$

$$p(z|x) = \frac{p_\theta(x|z) p_\theta(z)}{p_\theta(x)}$$

Solution

- Let's approximate $p(z|x)$ with another distribution by another distribution $q(z)$
- If $q(z)$ is a tractable distribution e.g. Gaussian distribution
- **Approach**: We can adjust parameters of $q(z)$ and make it as close to $p(z|x)$, i.e. $q(z) \approx p(z|x)$.
- **Goal**: Minimize the Kullback-Leibler (KL) divergence $KL(q||p)$.

Variational Inference (cont.)

$$
\begin{aligned}
KL(q(z)||p(z|x)) &= -\sum q(z) log \frac{p(z|x)}{q(z)} \\
&= -\sum q(z) log \frac{\frac{p(x,z)}{p(x)}}{q(z)} \\
&= -\sum q(z) log \left(\frac{p(x,z)}{q(z)} \frac{1}{p(x)} \right) \\
&= -\sum q(z) \left[log \frac{p(x,z)}{q(z)} + log \frac{1}{p(x)} \right] \\
&= -\sum q(z) \left[log \frac{p(x,z)}{q(z)} - log\ p(x) \right] \\
&= -\sum_z q(z) log \frac{p(x,z)}{q(z)} + log\ p(x) \sum_z q(z) \\
&= -\sum_z q(z) log \frac{p(x,z)}{q(z)} + log\ p(x) \quad \because \sum_z q(z) = 1
\end{aligned}
$$

Variational Inference (cont.)

- $$KL(q(z)||p(z|x) = -\sum_z q(z) log \frac{p(x,z)}{q(z)} + log\ p(x)$$

- We can also write above equation as:

$$log\ p(x) = KL(q(z)||p(z|x) + \sum_z q(z) log \frac{p(x,z)}{q(z)}$$

Variational Inference (cont.)

- Given x , $log\ p(x)$ is a constant
- $KL(q(z)||p(z|x)$ is the quantity we wanted to minimize
- Assume $L = \sum_z q(z) log \frac{p(x,z)}{q(z)}$, then

$$\text{constant} = KL + L$$

$$L \leq log\ p(x) \quad \because kl \geq 0$$

- **Instead of minimizing KL we can maximise L**

VAE: **Evidence Lower Bound (ELBO)**

Evidence Lower Bound (ELBO)

What is ELBO?:

- Allows us to optimize the model.
- It provides a lower bound on the log-likelihood of the data, which we aim to maximize during training.
- The ELBO consists of two main components:
  - **Reconstruction term**: Measures how well the model can reconstruct the input data from the latent representation.
  - **Regularization term**: Encourages the learned latent distribution to be close to a prior distribution (usually Gaussian).

Evidence Lower Bound (ELBO) (cont.)

- Mathematically, the ELBO can be expressed as:

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

where:

  - $q_\phi(z|x)$ is the approximate posterior distribution (encoder).
  - $p_\theta(x|z)$ is the likelihood of the data given the latent variable (decoder).
  - D_{KL} is the Kullback-Leibler divergence between the approximate posterior and the prior distribution $p(z)$.

- The goal is to maximize the ELBO with respect to the model parameters θ and ϕ.
- By maximizing the ELBO, we ensure that the model learns a meaningful latent representation while also being able to generate new samples.

Evidence Lower Bound (ELBO) (cont.)

Looking at Lower bound L

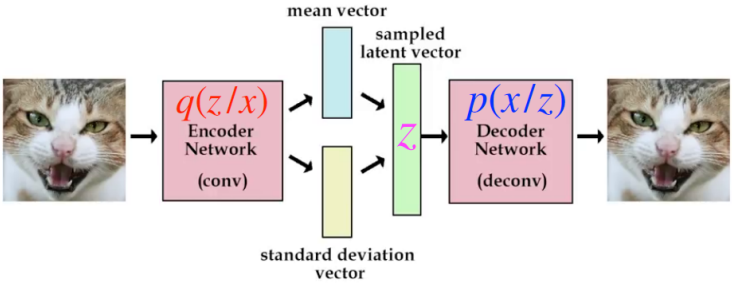
$$
\begin{aligned}
L &= \sum_z q(z) log \frac{p(x,z)}{q(z)} \\
&= \sum_z q(z) log \frac{p(x|z)p(z)}{q(z)} \\
&= \sum_z q(z) \left[log\ p(x|z) + log \frac{p(z)}{q(z)} \right] \\
&= \underbrace{\sum_z q(z) log\ p(x|z)}_{\text{Expectation } E_{q(z)}(log\ p(x|z))} + \underbrace{\sum_z q(z) log \frac{p(z)}{q(z)}}_{-KL(q(z)||p(z)}
\end{aligned}
$$

So,

$$
L = E_{q(z)}(log\ p(x|z)) - KL(q(z)||p(z))
$$

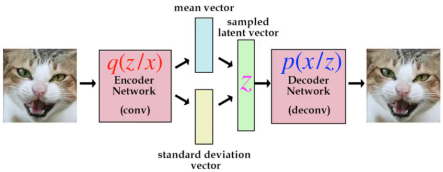
Evidence Lower Bound (ELBO) (cont.)

- $E_{q(z)}(log\ p(x|z))$ is conceptually reconstruction
- We can assume z to be Standard Normal Distribution



Variational Autoencoder Architecture

Evidence Lower Bound (ELBO) (cont.)



$$p(x|\hat{x}) = e^{-|x-\hat{x}|^2}$$

$$log\ e^{-|x-\hat{x}|^2} = -|x-\hat{x}|^2$$

$$L = E_{q(z)}(-|x-\hat{x}|^2) - KL(q(z)||p(z))$$

$$\min |x-\hat{x}| + KL(q(z|x)||\mathcal{N}(0,1))$$

$$\min |x-\hat{x}| - 0.5 * (1 + log\ \sigma^2 - \sigma^2 - \mu^2)$$

For full derivation of KL Loss, read here

VAE: **Reparameterization Trick (Pathwise Derivative Estimator)**

Reparameterization Trick

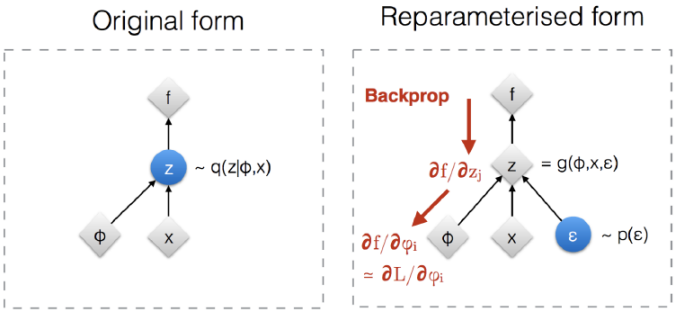
Problem: Cannot backpropagate through stochastic sampling.

Solution: Reparameterize z as:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

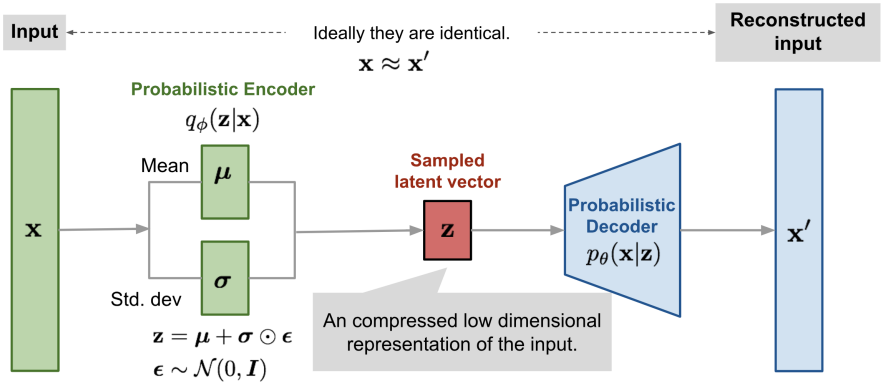
Benefit: Enables gradient-based optimization by making the sampling operation differentiable.

Reparameterization Trick (cont.)



Reparameterization trick to make back propagation possible

Reparameterization Trick (cont.)

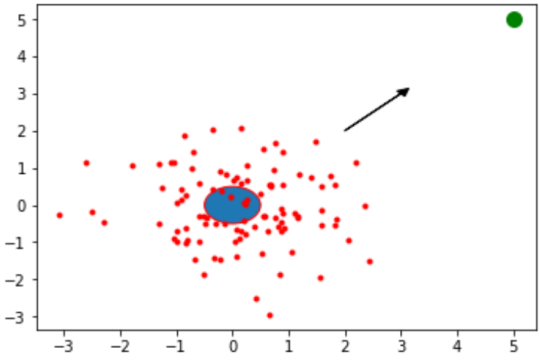


Variational Autoencoder with reparameterization trick

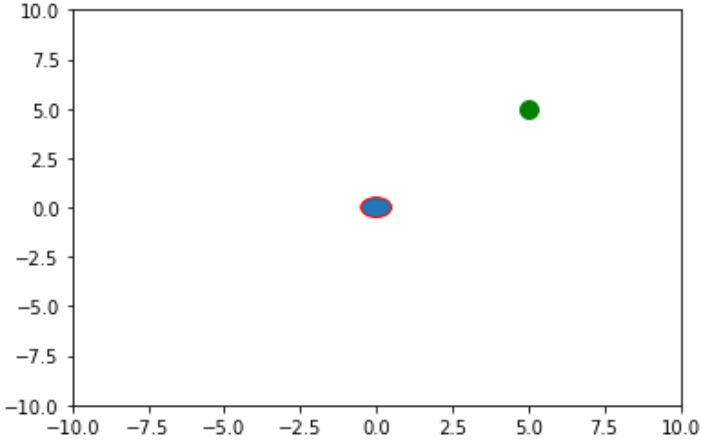
Pathwise Derivative - Toy Problem

Learn $\mu \in \mathbb{R}^2$ to minimize the objective below to reach the green point

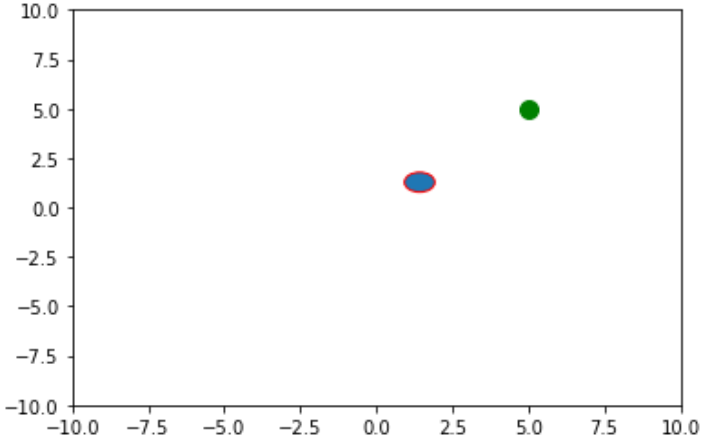
$$\mathcal{L} = \mathbb{E}_{x \sim N(\mu, I)} \left\| x - \begin{bmatrix} 5 \\ 5 \end{bmatrix} \right\|_2^2$$



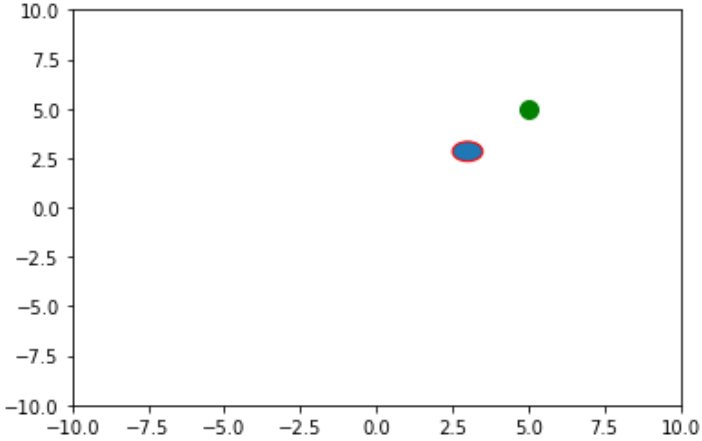
Pathwise Derivative - Toy Problem (cont.)



Pathwise Derivative - Toy Problem (cont.)



Pathwise Derivative - Toy Problem (cont.)



VAE: **Loss Function**

Loss Function Breakdown

Total Loss:

$$L = \text{Reconstruction Loss} + \text{KL Divergence}$$

Reconstruction Loss:

- Measures how well $\hat{x}$ matches x.
- Common choices: Mean Squared Error (MSE) or Binary Cross-Entropy.

KL Divergence:

- Measures how much $q(z \mid x)$ diverges from the prior $p(z)$.
- Encourages the latent space to follow a standard normal distribution.

VAE: **Results**

Results

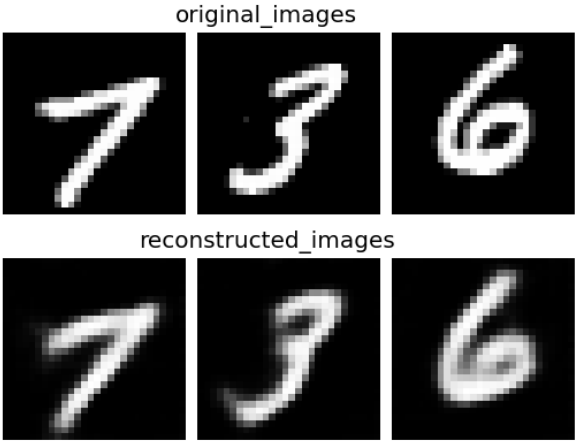


Image reconstruction with variational autoencoders on MNIST digits dataset

Results (cont.)

generated_images

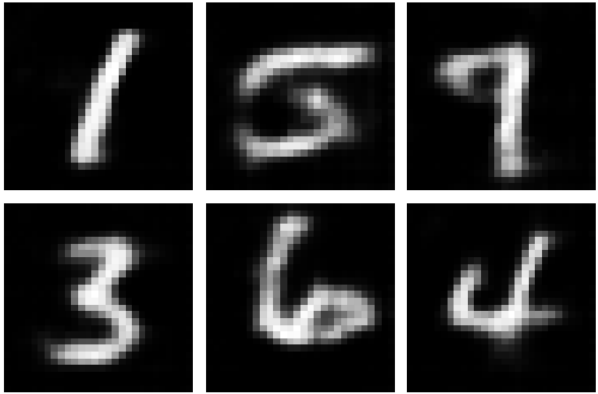
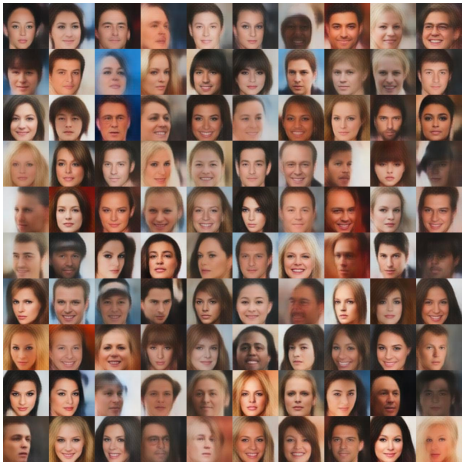


Image generation with variational autoencoders on MNIST digits dataset. Sample an encoding vector from $\mathcal{N}(0, 1)$ and passed it through decoder

Results (cont.)

Image generation with variational autoencoders on CIFAR-10 32x32 dataset

Results (cont.)



https://github.com/houxianxu/DFC-VAE

VAE: **Variants**

Variants and Extensions

β-VAE:

- Introduces a hyperparameter β to control the trade-off between reconstruction and regularization.
- Encourages disentangled representations.

Conditional VAE (CVAE):

- Incorporates additional information y (e.g., class labels) into the encoder and decoder.
- Enables generation of data conditioned on y.

Discrete VAE:

- Utilizes discrete latent variables.
- Techniques like Gumbel-Softmax are used for differentiable sampling.

Disentangled Variational Autoencoders (β-VAEs)

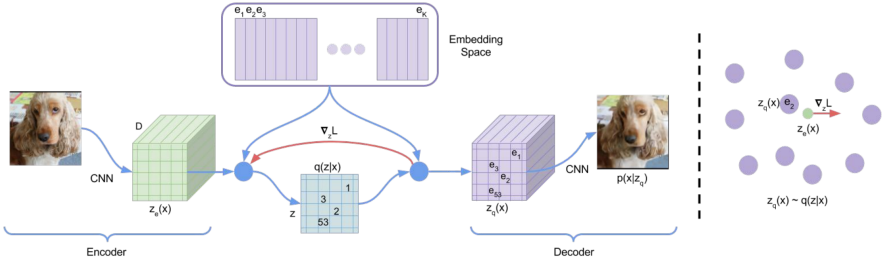
- **Basic Idea**: Different neurons in latent space should be uncorrelated, i.e. they all try to learn something different about input data.
- **Implementation**:

$$\mathcal{L}(\theta, \phi; x, z, \beta) = E_{q_\phi(z|x)}(log\ p_\theta(x|z)) - \beta\, KL(q_\phi(z|x)||p(z))$$

- Increasing the β is forcing variational autoencoder to encode the information in only few latent variables

Figure 2: Azimuthal rotation in β-VAEs and simple VAEs. β-VAEs produce more disentangled rotation, whereas some other features also change in simple VAEs.

Vector Quantized - Variational Autoencoders



- A latent embedding space $e \in \mathbb{R}^{K \times D}$ where K is the size of the discrete latent space (K-way categorical distribution)
- Encoder output $z_e(x)$ is mapped to discrete latent code z by a nearest neighbour look-up using the shared embedding space e
- The posterior categorical distribution $q(z|x)$ probabilities are defined as one-hot as:

$$q(z = k|x) = \begin{cases} 1 & \text{for } k = argmin_j ||z_e(x) - e_j||_2 \\ 0 & \text{otherwise} \end{cases}$$

- This model can be seen as a VAE in which the proposal distribution $q(z = k|x)$ is deterministic.
- The representation $z_e(x)$ is passed through the discretization bottleneck followed by mapping onto the nearest element of embedding e

$$z_q(x) = e_k, \qquad \text{where} \qquad k = argmin_j ||z_e(x) - e_j||_2$$

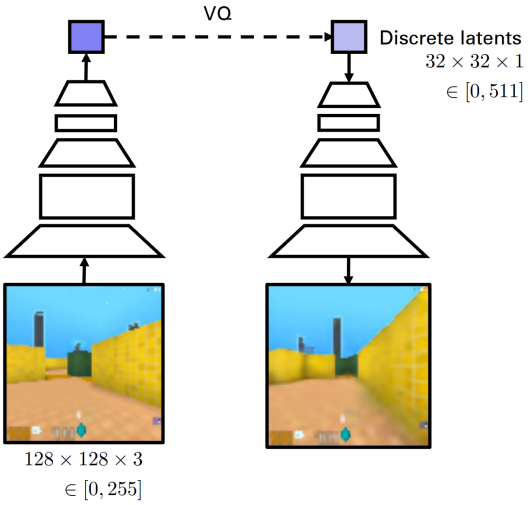
$$L = log\ p(x|z_q(x)) + ||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2$$

sg stands for the **stopgradient** operator that is defined as identity at forward computation time and has zero partial derivatives, thus effectively constraining its operand to be a non-updated constant.

- The embedding space is learned via Vector Quantization (VQ), whose objective uses the l_2 error to move the embedding vectors e_i towards the encoder outputs $z_e(x)$

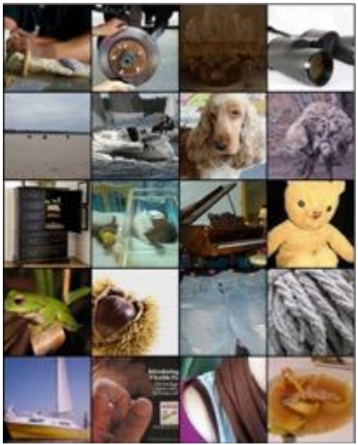
VQ-VAE

- For speech, image and videos one can respectively extract a 1D, 2D and 3D latent feature spaces
- 32 x 32 latents for ImageNet, or 8 x 8 x 10 for CIFAR10
- 512-dimensional latents for VCTK and LibriSpeech

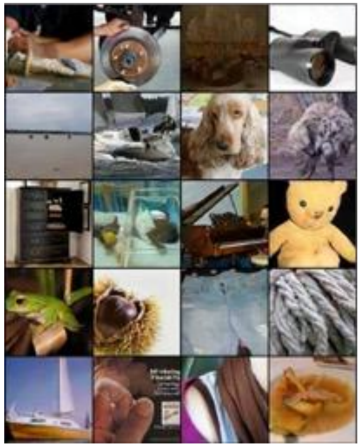


VQ-VAE ImageNet Reconstruction

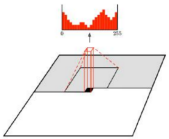
Original 128 x 128 images



Reconstructions



VQ-VAE Sampling



PixelCNN
Prior

Sampled
Discrete latents

$32 \times 32 \times 1$
$\in [0, 511]$

Decoder

VQ-VAE ImageNet Samples

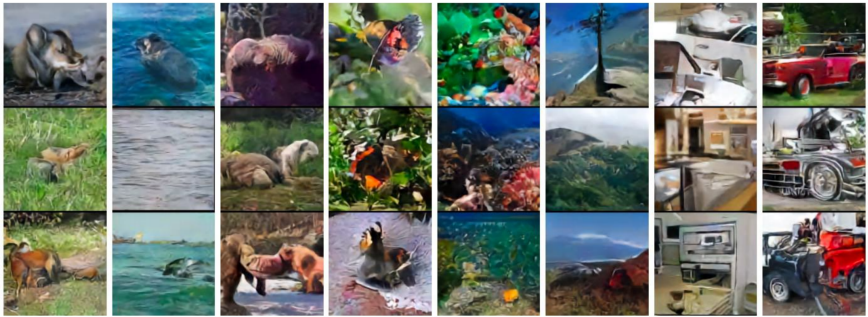
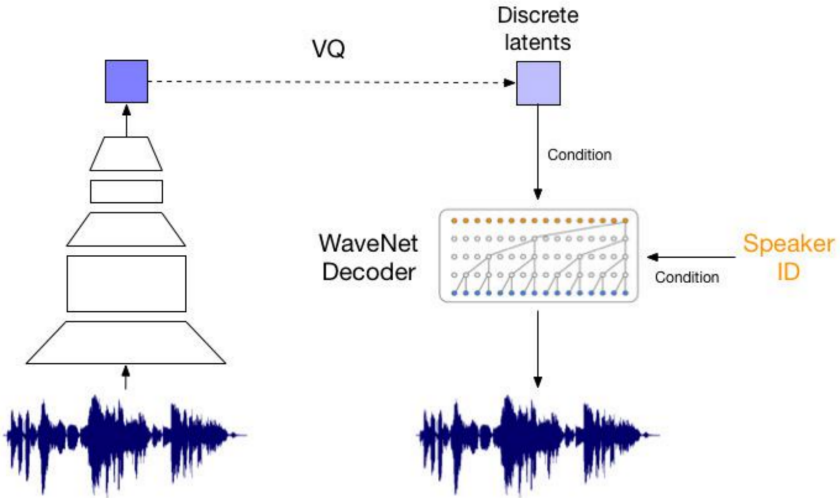
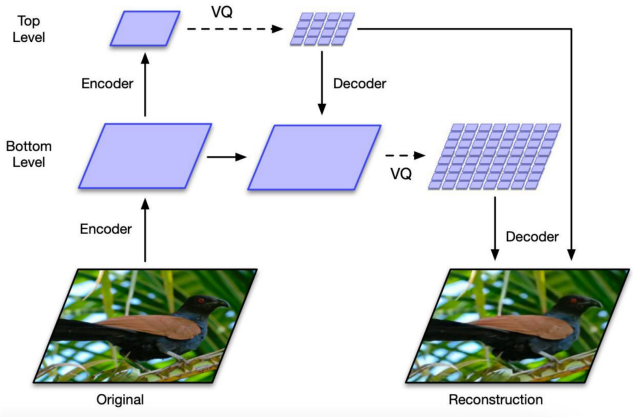


Figure 3: Samples (128x128) from a VQ-VAE with a PixelCNN prior trained on ImageNet images. From left to right: kit fox, gray whale, brown bear, admiral (butterfly), coral reef, alp, microwave, pickup.

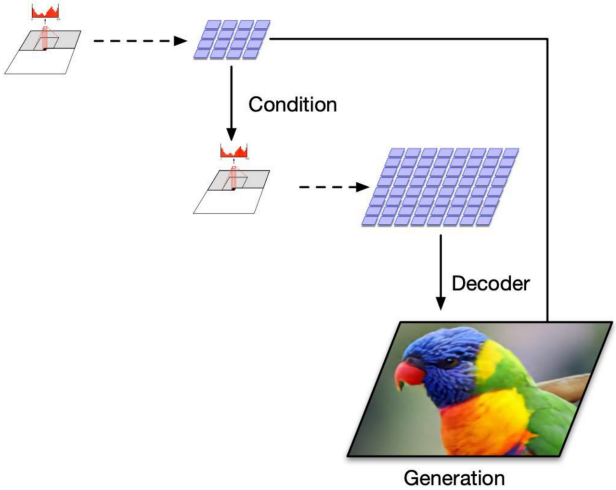
VQ-VAE Voice Style Transfer



- The input 256×256 image is compressed to quantized latent maps of size 64×64 and 32×32
- The decoder reconstructs the image from the two latent maps.



VQ-VAE Sampling



VQ-VAE ImageNet Samples

VAE: **Limitations and Challenges**

Limitations and Challenges

Limitations:

- **Gaussian Assumption:** The assumption that the latent variables follow a Gaussian distribution may not hold for all datasets.
- **Over-smoothing:** The model may produce overly smooth reconstructions, losing fine details in the data.
- **Sensitivity to Hyperparameters:** The performance of VAEs can be sensitive to the choice of hyperparameters, such as the weight of the KL divergence term.
- **Computational Complexity:** Training VAEs can be computationally expensive, especially for large datasets or complex models.
- **Evaluation Metrics:** Evaluating the quality of generated samples can be subjective and challenging, as traditional metrics may not capture the nuances of the data.

Limitations and Challenges (cont.)

Challenges:

- **Training Instability:** VAEs can be difficult to train, especially with complex datasets.
- **Mode Collapse:** The model may generate samples from only a subset of the latent space.
- **Balancing Reconstruction and Regularization:** Finding the right balance between reconstruction loss and KL divergence can be tricky.
- **Posterior Collapse:** In some cases, the model may ignore the latent variables, leading to poor representations.
- **Limited Expressiveness:** The Gaussian assumption for the latent space may not capture complex data distributions.
- **Disentanglement:** Achieving disentangled representations can be challenging, especially in high-dimensional spaces.

Limitations and Challenges (cont.)

Future Directions:

- **Improved Training Techniques:** Developing better optimization methods to stabilize training.
- **Advanced Architectures:** Exploring more complex latent variable models, such as Normalizing Flows or Hierarchical VAEs.
- **Better Regularization Techniques:** Investigating alternative regularization methods to improve disentanglement and representation quality.
- **Hybrid Models:** Combining VAEs with other generative models (e.g., GANs) to leverage their strengths.
- **Application-Specific Variants:** Tailoring VAEs for specific applications, such as text or video generation.

VAE: **References**

References

Reference Slides

- Fei-Fei Li "Generative Deep Learning" CS231
- Hao Dong "Deep Generative Models"
- Hung-Yi Lee "Machine Learning"
- Murtaza Taj "Deep Learning" CS437
- Aykut Erdem, COMP547: Deep Unsupervised Learning, Koc University

References (cont.)

[1] D. P. Kingma and M. Welling,

"Auto-Encoding Variational Bayes,"

arXiv preprint arXiv:1312.6114, 2013.

[2] D. J. Rezende, S. Mohamed, and D. Wierstra,

"Stochastic Backpropagation and Approximate Inference in Deep Generative Models,"

Proceedings of the 31st International Conference on Machine Learning (ICML), 2014.

[3] I. Higgins, L. Matthey, A. Pal, et al.,

"beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework,"

International Conference on Learning Representations (ICLR), 2017.

References (cont.)

[4] C. P. Burgess, I. Higgins, A. Pal, et al.,
"Understanding Disentangling in β-VAE,"
arXiv preprint arXiv:1804.03599, 2018.

[5] D. P. Kingma, T. Salimans, R. Jozefowicz, et al.,
"Improved Variational Inference with Inverse Autoregressive Flow,"
Advances in Neural Information Processing Systems (NeurIPS), 2016.

[6] A. van den Oord, O. Vinyals, and K. Kavukcuoglu,
"Neural Discrete Representation Learning,"
Advances in Neural Information Processing Systems (NeurIPS), 2017.

References (cont.)

[7] I. Higgins, D. Amos, D. Pfau, et al.,
"Towards a Definition of Disentangled Representations,"
arXiv preprint arXiv:1812.02230, 2018.

[8] C. Doersch,
"Tutorial on Variational Autoencoders,"
arXiv preprint arXiv:1606.05908, 2016.

[9] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe,
"Variational Inference: A Review for Statisticians,"
Journal of the American Statistical Association, 2017.

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.