Autoencoders & Variational Autoencoders

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

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

LMH
Lady Margaret Hall

July 25, 2025

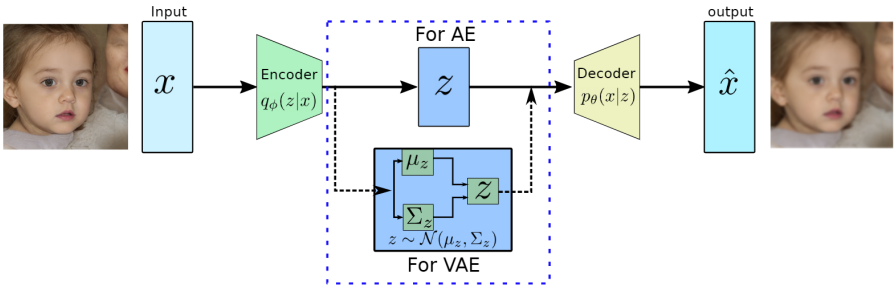

Input
x
Encoder
$q_\phi(z|x)$
For AE
z
μ_z
Σ_z
z
$z \sim \mathcal{N}(\mu_z, \Sigma_z)$
For VAE
Decoder
$p_\theta(x|z)$
output
$\hat{x}$

Table of Contents

Table of Contents (cont.)

Table of Contents (cont.)

Deep Unsupervised Learning (DUL): **Motivation**

Supervised learning is effective, but it has some limitations:

- Needs a lot of labeled data to work well.
- Getting labels can be difficult or costly.
- Depends on experts to label the data.
- Not practical for many real-world situations where labeling is hard or impossible.

The Bottleneck

- Humans are good at:
  - Learning patterns from experience
  - Discovering hidden structures
- But:
  - They are slow and limited
  - Not always available
  - Can't always explain what they observe

A New Possibility

- What if machines could learn to:
  - Find hidden patterns
  - Understand structures in data
  - Group and compare similar things

Without any human-provided labels?

Unsupervised Learning to the Rescue

- Learns **without labels**
- Discovers **structure within the data**
- Mimics how humans learn from **observation**
- Useful for:
  - Clustering
  - Dimensionality reduction
  - Representation learning

Why Deep?

- Simple models can't handle:
  - High variability in real-world data
  - Complex abstract features
- We need:
  - Neural networks
  - With many layers $\rightarrow$ **"Deep"**
  - To learn hierarchical patterns

Deep Unsupervised Learning — The Goal

- Discover hidden patterns
- Learn rich representations
- Group and distinguish data points
- Match similar structures

Deep Unsupervised Learning (DUL): **Introduction**

What is Deep Unsupervised Learning?

Deep Unsupervised Learning aims to discover **hidden patterns** and **structures** in raw data using deep neural networks, **without any human-provided labels**.

It focuses on **capturing rich patterns in raw data with deep networks in a label-free way**.

- **Learns from observation** — mimics how humans learn from experience
- **Discovers structure** — finds groups, similarities, and representations
- **Scales to complex data** — uses deep models to capture abstract features
- **Generative Models** — recreate raw data distribution
- **Self-supervised Learning** — "puzzle" tasks that require semantic understanding

Why do we care?

Geoffrey Hinton
(in his 2014 AMA on Reddit)

"The brain has about 10^{14} *synapses and we only live for about* 10^{9} *seconds. So we have a lot more parameters than data. This motivates the idea that we must do a lot of unsupervised learning since the perceptual input (including proprioception) is the only place we can get* 10^{5} *dimensions of constraint per second."*

Yann LeCun

Need tremendous amount of information to build machines that have common sense and generalize well.

[LeCun-20161205-NeurIPS-keynote]

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- **(Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)**

"Ideal Intelligence"

- **"Ideal Intelligence" is all about compression** (finding all patterns)
- Finding all patterns = short description of raw data (*low Kolmogorov Complexity*)
- Shortest code-length = optimal inference (*Solomonoff Induction*)
- Extensible to optimal action-making agents (*AIXI*)

Transfer Learning via Compression:

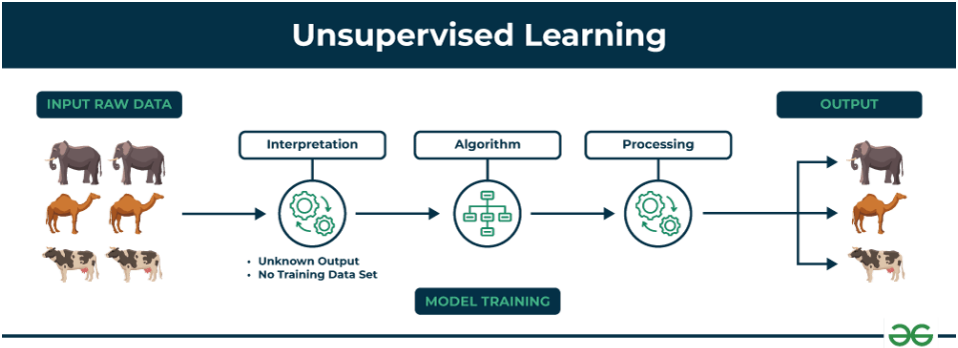
- Assume we pretrain unsupervised on Data Distribution D_1 and then finetune on Data Distribution D_2
- If D_1 and D_2 are related, compressing D_2 conditioned on D_1 should be more efficient than compressing D_2 outright
- **Hence:** pretraining on D_1 should aid faster learning of D_2

Deep Unsupervised Learning (DUL): **Components**

Components of Unsupervised Learning

- **Data**: Unlabeled data, e.g., images, text, or sensor readings.
- **Model**: A mathematical representation of the data, e.g., a mixture model or a neural network.
- **Objective function**: A measure of how well the model fits the data, e.g., likelihood or reconstruction error.
- **Optimization algorithm**: An algorithm to minimize the objective function, e.g., gradient descent or expectation-maximization.
- **Evaluation metrics**: Measures to assess the quality of the learned model, e.g., silhouette score or clustering accuracy.

Applications: Use cases for unsupervised learning, e.g., clustering, dimensionality reduction, or anomaly detection.



Source: geeksforgeeks.org

Supervised vs Unsupervised Learning

Aspect	Supervised Learning	Unsupervised Learning
Objective	Learn a function f from labeled input–output pairs.	Discover structure or representations in unlabeled data.
Evaluation	Accuracy, precision/recall on held-out labels.	Clustering validity indices (e.g. silhouette), reconstruction error.
Cost	Methods range from $\mathcal{O}(n)$ to $\mathcal{O}(n^3)$ per fit.	k-means $\mathcal{O}(nkd)$, hierarchical $\mathcal{O}(n^2)$, PCA $\mathcal{O}(nd^2)$.
Labels/Clusters	Fixed, known set of classes.	Number of clusters unknown; must be chosen or inferred.
Output	Classifier or regressor for new inputs.	Cluster assignments, embeddings, density models, or generative samples.

Key differences between Supervised and Unsupervised Learning

Deep Unsupervised Learning (DUL): **Applications**

Unsupervised Learning - Applications

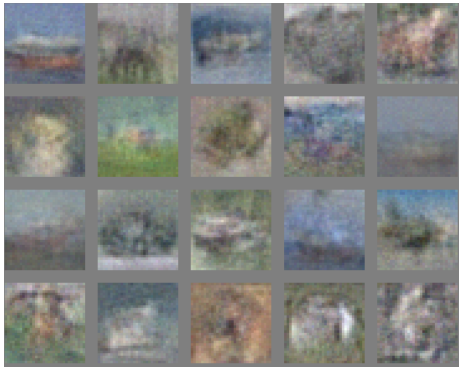
Deep Unsupervised Learning has many powerful applications:

- **Generate Novel Data**: Create new data samples similar to the training set.
- **Conditional Synthesis Technology**: Enable advanced applications such as WaveNet for audio generation and GAN-based models like pix2pix for image-to-image translation.
- **Compression**: Learn compact representations for efficient data storage and transmission.
- **Improve Downstream Tasks**: Enhance performance of supervised tasks through unsupervised or self-supervised pre-training.
- **Production-Level Impact**: Power real-world systems, e.g., Google Search improvements with BERT pre-training.
- **Flexible Building Blocks**: Provide reusable components for a wide range of machine learning pipelines.

Application: Generate Images

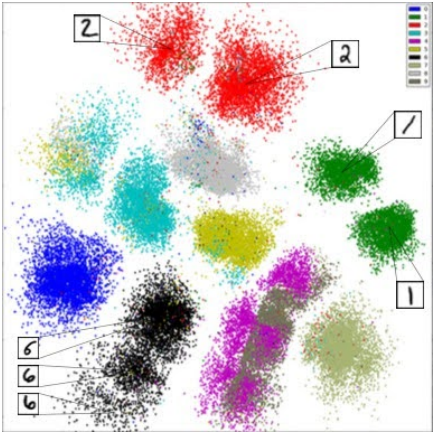
[Deep Belief Nets, Hinton, Osindero, Teh, 2006]

Application: Generate Images (cont.)



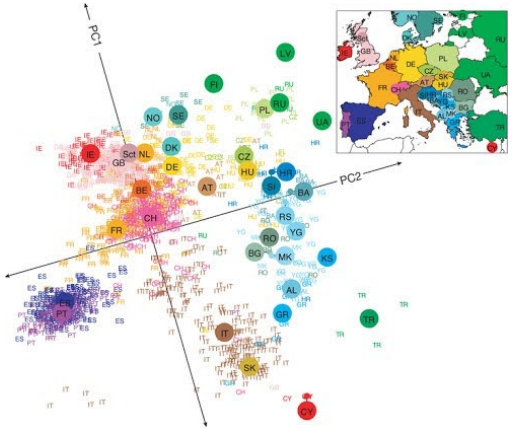
[GAN, Goodfellow et al. 2014]

Application: Discovering Structure in Digits



Unsupervised learning can discover structure in digits without any labels.

Application: DNA Analysis

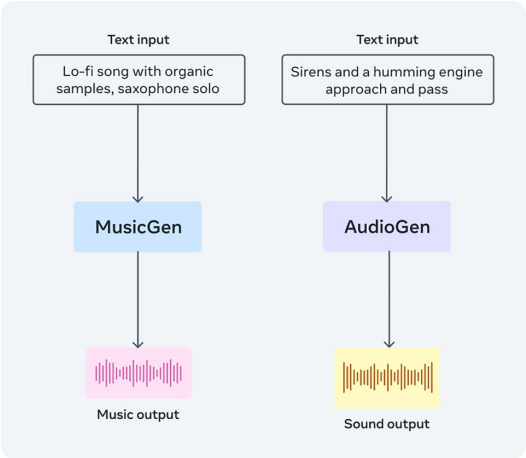


Dimensionality reduction applied to DNA reveal the geography of European countries.

Application: Text to Image Generation

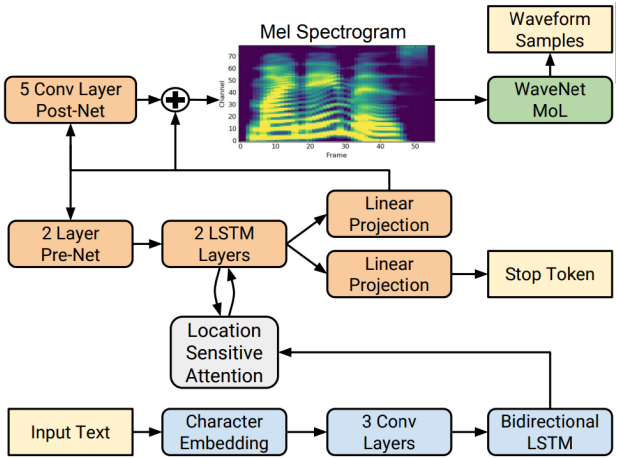
"a masterful oil painting a persian exotic cat discovering their astounding crypto losses while checking their phone"

Application: Text to Audio Generation



[AudioCraft, Copet et al., 2023]

Application: Text to Speech Generation



[Tacotron2 , Shen et al., 2018]

Application: Generate Math

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_*} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ??? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x''}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_S(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathcal{O}_{\mathrm{Spec}(k)}} \mathcal{O}_{S,s} - i_{\overline{X}}^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \bar{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

[Char-rnn, karpathy, 2015]

Application: Generate Text

Prompt [human written]: A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.
GPT-2: The incident occurred on the downtown train line, which runs from Covington and Ashland stations. In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief. 'The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,' said Tom Hicks, the U.S. Energy Secretary, in a statement. 'Our top priority is to secure the theft and ensure it doesn't happen again.' The stolen material was taken from the University of Cincinnati's Research Triangle Park nuclear research site, according to a news release from Department officials. The Nuclear Regulatory Commission did not immediately release any information. According to the release, the U.S. Department of Energy's Office of Nuclear Material Safety and Security is leading that team's investigation. 'The safety of people, the environment and the nation's nuclear stockpile is our highest priority,' Hicks said. 'We will get to the bottom of this and make no excuses.'

[OpenAI's GPT2 – Radford, Wu, Child, Luan, Amodei, Sutskever, 2019]

Deep Unsupervised Learning (DUL):
Challenges

Unsupervised Learning - Challenges

Why is unsupervised learning challenging?

- **Lack of labeled data**:
  We don't have labels or answers to guide us—it's like solving a puzzle without a picture.
- **Difficult to assess performance**:
  It's hard to know if our results are good, since there's nothing to compare them to.
- **Sensitivity to noise**:
  The algorithms can get confused by noisy or unusual data points.
- **Curse of dimensionality**:
  When there are too many features, it's tough to find real patterns—the data gets too spread out.
- **Interpretability issues**:
  The results can be hard to understand or explain, making it tricky to use them effectively.

Autoencoders (AE)

Autoencoders: **Motivation**

Motivation

- Real-world data is often **large**, noisy, and difficult to interpret.
- We aim to **compress** data efficiently while preserving essential information.
- It is also valuable for machines to **generate** new, realistic data samples.
- **Autoencoders** are a type of **unsupervised learning** model that address both compression and generation tasks.
- They **learn** to represent data in a **lower-dimensional space**, capturing its most important features.

Autoencoders: **Learning Outcomes**

⇒ Learning Outcomes

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

- Explain the basic concept and purpose of autoencoders
- Identify and describe the main components: **encoder**, **bottleneck**, and **decoder**
- Understand how autoencoders can be used to generate new data
- Recognize practical applications and limitations of autoencoders

Autoencoders: **Introduction**

Autoencoders

Autoencoders are a type of artificial neural network used for learning efficient codings of input data in an unsupervised manner. The goal is to learn a compressed (encoded) representation of the input and then reconstruct the original input from this compressed version.

- **Developed originally in the 1980s**, autoencoders have gained renewed popularity due to advances in deep learning and hardware.
- Often used in **dimensionality reduction, denoising, anomaly detection, and generative modeling**.

Autoencoders (cont.)

- Autoencoders are a family of neural networks designed to learn efficient representations of data, typically for dimensionality reduction or feature learning.
- They work by compressing the input into a latent-space representation (encoding), and then reconstructing the original input from this representation (decoding).
- The main goal is to project the input into a lower-dimensional latent space and then reconstruct the input from that latent representation.

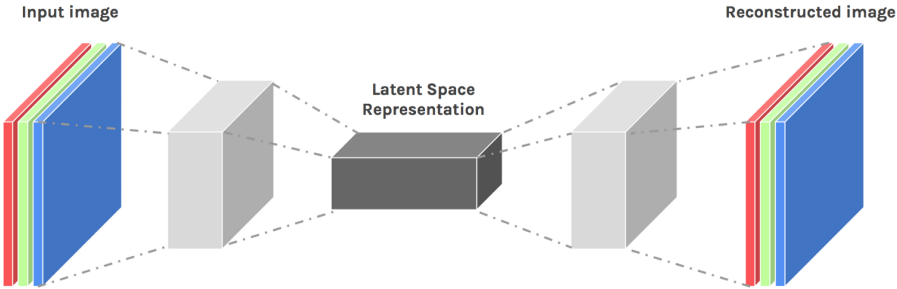
Autoencoders (cont.)

Autoencoders consist of two main components:

- **Encoder**: Maps the input data to a latent space Z, compressing the input into a lower-dimensional representation.
- **Decoder**: Reconstructs the input data from the encoded latent representation.

Variants of autoencoders may use altered or corrupted versions of the input as the target output, which can help the model learn more robust features (e.g., denoising autoencoders).

Autoencoders (cont.)



Autoencoder architecture

Autoencoders (cont.)

- **Bottleneck (Latent Space)**
  - The bottleneck is a small hidden layer in the middle of the network.
  - It forces the network to squeeze the input into a shorter, simpler code.
  - This helps the network find and keep only the most important information.
  - A small bottleneck stops the autoencoder from just memorizing the input.
- **Loss Function**
  - Tells us how close the output is to the original input.
  - The most common loss is **Mean Squared Error (MSE)**—it measures the average difference between input and output.
  - For special cases, we can use other losses, like binary cross-entropy for yes/no data, or KL divergence for special types of autoencoders.
  - The loss function guides the network to learn better and more useful features.

Autoencoders: **Architecture**

Design Considerations:

- **How deep?** More layers can learn more, but also make training harder and risk overfitting.
- **How wide?** More neurons per layer can help, but too many may just copy the input instead of learning.
- **Symmetry:** Encoder and decoder can look the same, but don't have to—sometimes different shapes work better.
- **Regularization:** Tricks like dropout or weight penalties help prevent the model from just memorizing.
- **Latent size:** The "bottleneck" size is key—too small loses info, too big doesn't compress.
- **Activation:** Functions like ReLU or sigmoid help the network learn complex patterns.

Autoencoders: Architecture (cont.)

Encoder:

- Takes input data (e.g., image, text, audio)
- Uses neural layers to reduce it to a smaller dimension
- Learns to keep only the important features
- *Example*: 784-pixel image to 32 numbers

Latent Space (Bottleneck):

- This is the core representation of the data
- It's where the model is forced to be efficient
- If too big: model memorizes; too small: model forgets
- Sweet spot = good generalization

Autoencoders: Architecture (cont.)

Decoder:

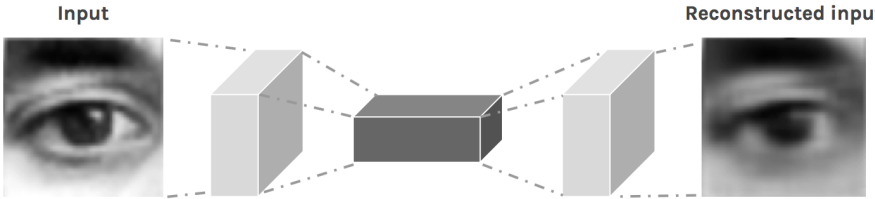
- Takes the bottleneck output
- Expands it back to the original input size
- Tries to make it as close as possible to the real input
- Learns to generate data from compressed features
- *Example*: Converts 32 numbers back to a 784-pixel image

Autoencoders: Architecture (cont.)

Component	Description
Encoder	Transforms the input data into a compressed latent representation using layers such as Dense, Convolutional, or Recurrent layers.
Latent Space (Bottleneck)	The central compact representation of the input. Forces the model to learn the most essential features.
Decoder	Reconstructs the input from the latent space. Typically mirrors the encoder's structure in reverse.
Loss Function	Measures the difference between input and output (e.g., Mean Squared Error, Binary Cross-Entropy). Drives learning during training.
Training	Involves minimizing reconstruction error using gradient-based optimization methods such as SGD or Adam.

Architecture Summary Table

Autoencoders: Architecture (cont.)



Sample Autoencoder

Autoencoders: Interactive Demo

https://douglasduhaime.com/posts/visualizing-latent-spaces.html

Autoencoders: **As Generative Models**

Autoencoders as Generative Models

- Autoencoders project data into a latent space Z.
- The latent space is not necessarily continuous or structured.

Autoencoders as Generative Models

- Autoencoders project data into a latent space Z.
- The latent space is not necessarily continuous or structured.
- *What if we sample a new embedding vector from Z and then have the decoder reconstruct the image from it?*

Autoencoders as Generative Models

- Autoencoders project data into a latent space Z.
- The latent space is not necessarily continuous or structured.
- *What if we sample a new embedding vector from Z and then have the decoder reconstruct the image from it?*
- **Does not work**. Autoencoders just learn a function that maps input to output. The learned latent space is too discontinous to work as a generative model.

Autoencoders as Generative Models

- Autoencoders project data into a latent space Z.
- The latent space is not necessarily continuous or structured.
- *What if we sample a new embedding vector from Z and then have the decoder reconstruct the image from it?*
- **Does not work**. Autoencoders just learn a function that maps input to output. The learned latent space is too discontinous to work as a generative model.
- Sampling randomly from the latent space may result in invalid outputs.
- They do not maximize the likelihood of the data.

Autoencoders as Generative Models (cont.)

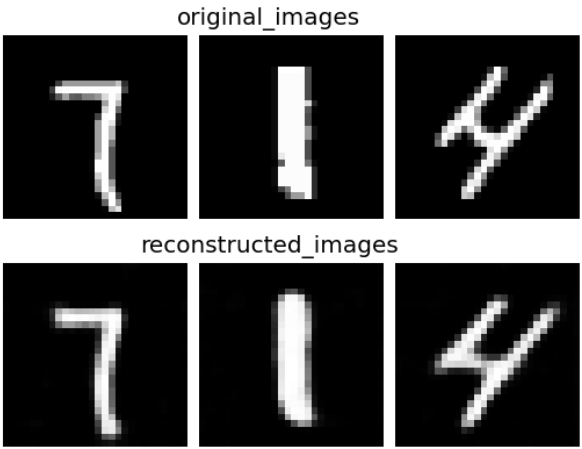


Image reconstruction with autoencoder trained on MNIST digits

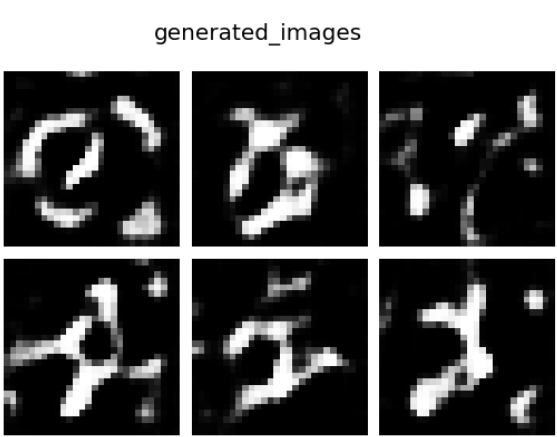


Image generation with autoencoder trained on MNIST digits. Encoding vector sampled from latent space Z and the passed to decoder.

Autoencoders: **Applications**

Applications of Autoencoders

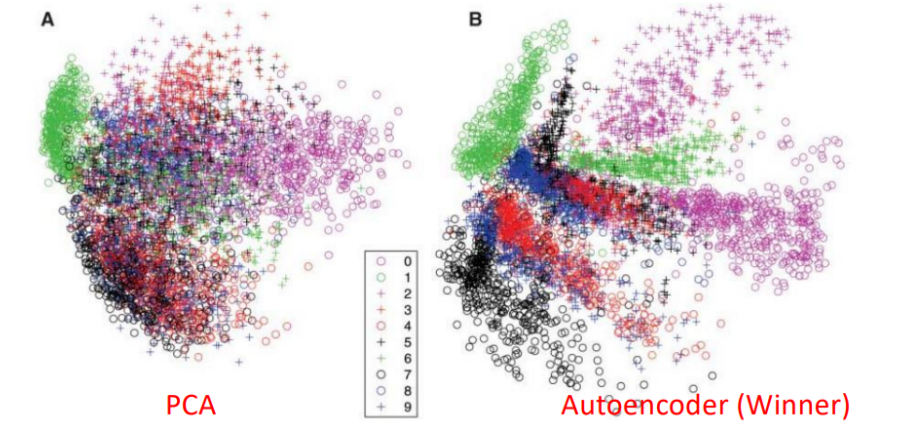
- Dimensionality Reduction
- Super-Resolution
- Colorization
- Anomaly Detection
- Denoising

Autoencoders help us reduce the number of features in our data, similar to PCA, but in a more flexible way.

- The encoder takes big, complex data and squeezes it into a smaller, simpler form.
- This smaller version keeps the important information.
- Great for making it easier to see patterns or to prepare data for other machine learning tasks.

Example: We can use autoencoders to turn images (like MNIST digits) into just 2 or 3 numbers, so we can plot and explore them easily.

Dimensionality Reduction (cont.)



t-SNE visualization on MNIST digits dataset. PCA vs. Autoencoders. The image vector is projected into $\mathbb{R}^2$.

Super-Resolution

Autoencoders can be used to reconstruct high-resolution images from their low-resolution counterparts.

- Input: Low-resolution image.
- Output: High-resolution image.
- Often implemented with convolutional layers for spatial pattern learning.

Use Case: Enhancing medical images, satellite images, or upscaling low-resolution photos.

Super-Resolution (cont.)

Image super-resolution using Autoencoders

Image Colorization

Autoencoders can learn to predict color information for grayscale images.

- Input: Grayscale image (1 channel).
- Output: Colorized image (3 channels — RGB).
- Requires learning semantic and contextual relationships to apply realistic colors.

Use Case: Restoring historical black-and-white photos.

Image Colorization (cont.)

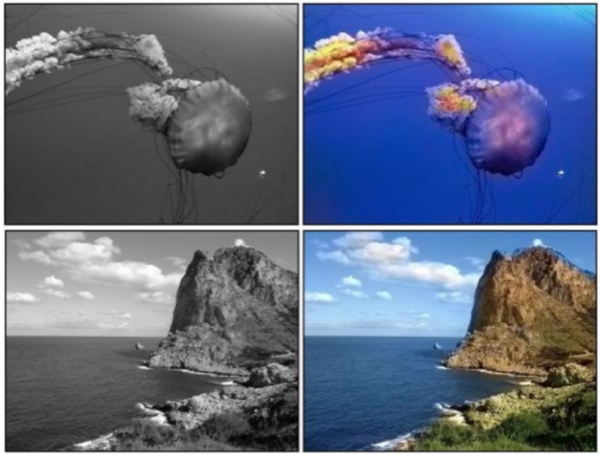


Figure 2: Image colorization using Autoencoders

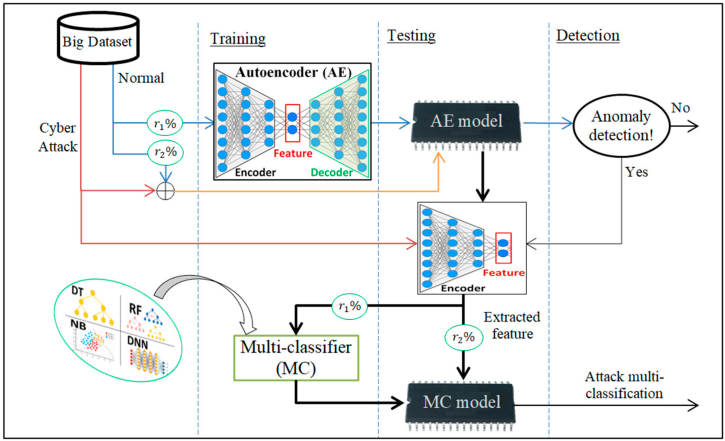
Anomaly Detection

- Autoencoders are trained to reconstruct input data.
- During training, the model learns to compress and decompress data that is similar to the training set (normal data).
- When presented with anomalous (outlier) data, the reconstruction error increases, as the autoencoder cannot effectively reconstruct unseen patterns.

Use Cases:

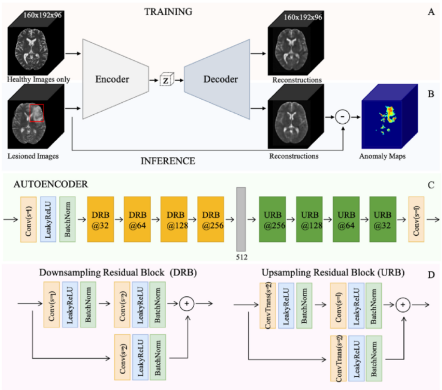
- Fraud detection in financial transactions
- Fault detection in industrial systems
- Intrusion detection in cybersecurity
- Medical anomaly detection (e.g., rare diseases in imaging)

Anomaly Detection (cont.)



Structure of the deep autoencoder (AE) for anomaly detection and feature extraction for multi-classification of cyber-attacks.

Anomaly Detection (cont.)



Autoencoder-based anomaly detection framework: A) Training the autoencoder for modeling the distribution of healthy brain anatomy. B) In the inference phase, anomaly maps reveal lesions by subtracting their reconstruction from the input image. C and D) Details of the proposed autoencoder.

Key Idea:

- Train the autoencoder to reconstruct the original, clean input from a corrupted (noisy) version.
- The encoder learns robust features that capture the underlying structure of the data, ignoring noise.

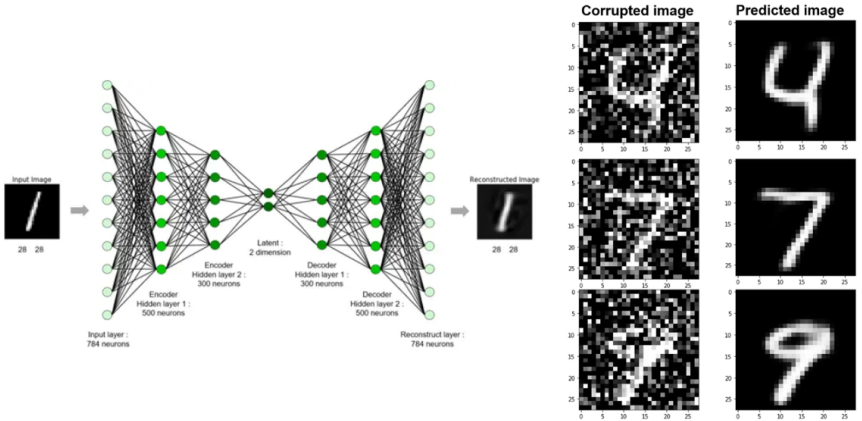
Training Process:

1. Add noise to the input data (e.g., Gaussian noise, salt-and-pepper noise).
2. Feed the noisy input to the autoencoder.
3. Compute the loss between the output and the original clean input.
4. Update the network weights to minimize this loss.

Use Cases:

- Image denoising: Removing noise from photographs or medical images.
- Speech enhancement: Improving audio quality by filtering out background noise.
- Preprocessing step: Providing cleaner data for downstream tasks such as classification or segmentation.

Denoising (cont.)



Example of denoising using an autoencoder.

Autoencoders: **Challenges**

Challenges

- May just copy input without real learning
- Needs careful tuning of bottleneck size
- Sometimes blurry outputs (especially basic autoencoders)
- Doesn't always generate creative results — use VAEs/GANs

Variational Autoencoders (VAE)

Variational Autoencoders: **Motivation**

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

Applications

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

Variational Autoencoders: **Learning Outcomes**

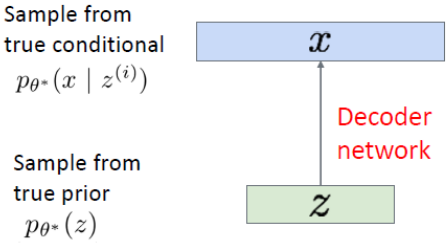
⇒ Learning Outcomes

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)**:

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

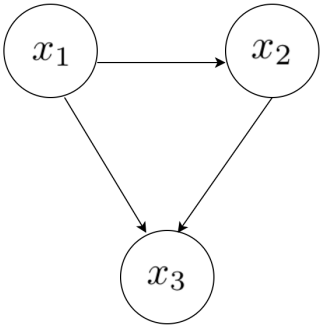
- The first term, $\mathbb{E}_{q(z|x)}[\log p(x|z)]$, encourages accurate reconstruction.
- The second term, $D_{\mathrm{KL}}(q(z|x)\|p(z))$, regularizes the latent space by making the approximate posterior $q(z|x)$ close to the prior $p(z)$, usually $\mathcal{N}(0, I)$.

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.

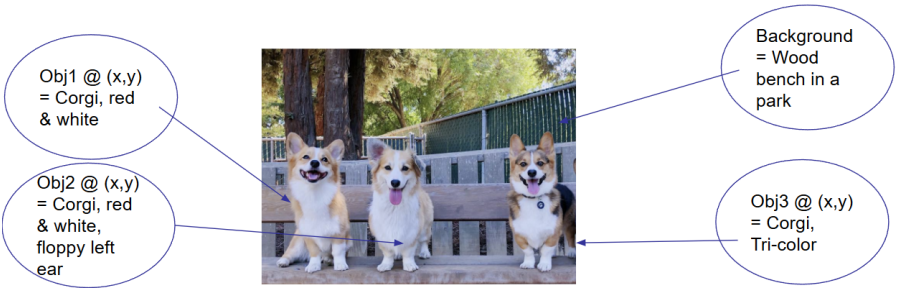


Latent Variable Models:

- **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.)

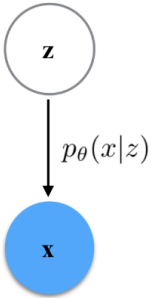
- Autoregressive (AR) models are slow to sample because all pixels (observation dimensions) are assumed to be dependent on each other.
- We can make part of the observation space independent *conditioned on some latent variables*.
- Latent variable models *can* have faster sampling by exploiting statistical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

Latent Variable Model

- **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?

- 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.

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

- 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.

- 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

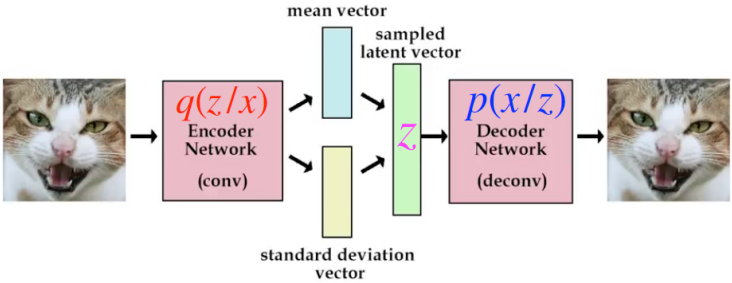
$$
\begin{aligned}
L &= \sum_z q(z) log \frac{p(x,z)}{q(z)} \\
&= \sum_z q(z) log \frac{p(x|z)p(z)}{q(z)} \\
&= \sum_z q(z) \left[log\ p(x|z) + log \frac{p(z)}{q(z)} \right] \\
&= \underbrace{\sum_z q(z) log\ p(x|z)}_{\text{Expectation } E_{q(z)}(log\ p(x|z))} + \underbrace{\sum_z q(z) log \frac{p(z)}{q(z)}}_{-KL(q(z)||p(z)}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

So,

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

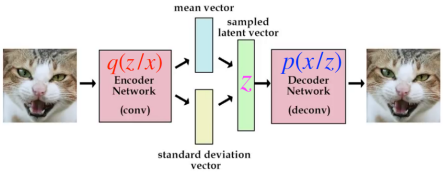
Evidence Lower Bound (ELBO) (cont.)

- $E_{q(z)}(log\ p(x|z))$ 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)**

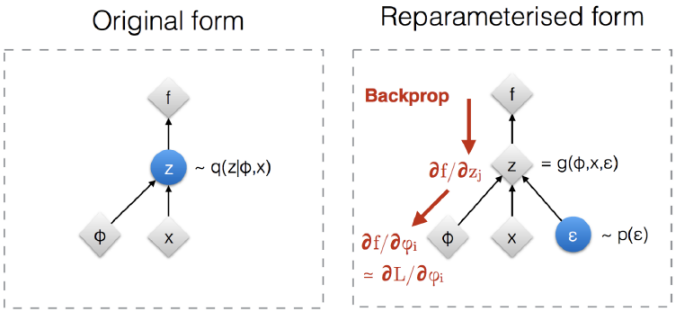
Problem: Cannot backpropagate through stochastic sampling.

Solution: Reparameterize z as:

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

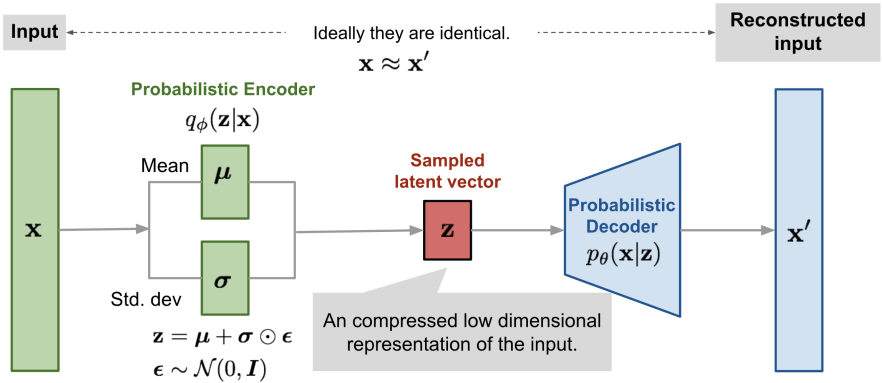
Benefit: Enables gradient-based optimization by making the sampling operation differentiable.

Reparameterization Trick (cont.)



Reparameterization trick to make back propagation possible

Reparameterization Trick (cont.)

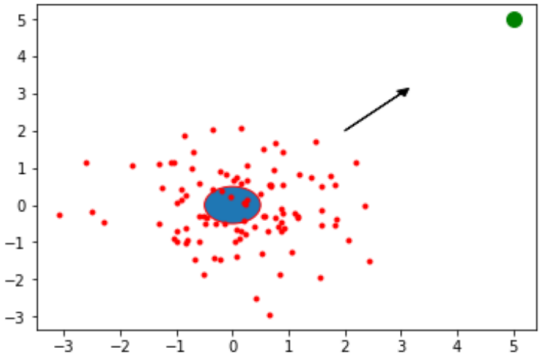


Variational Autoencoder with reparameterization trick

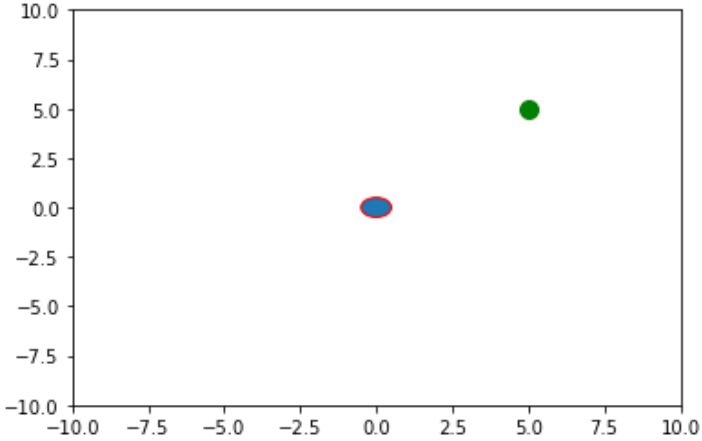
Pathwise Derivative - Toy Problem

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

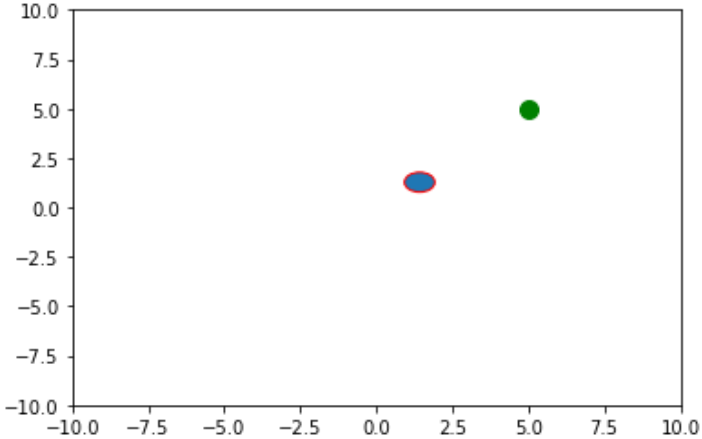
$$\mathcal{L} = \mathbb{E}_{x \sim N(\mu, I)} \left\| x - \begin{bmatrix} 5 \\ 5 \end{bmatrix} \right\|_2^2$$



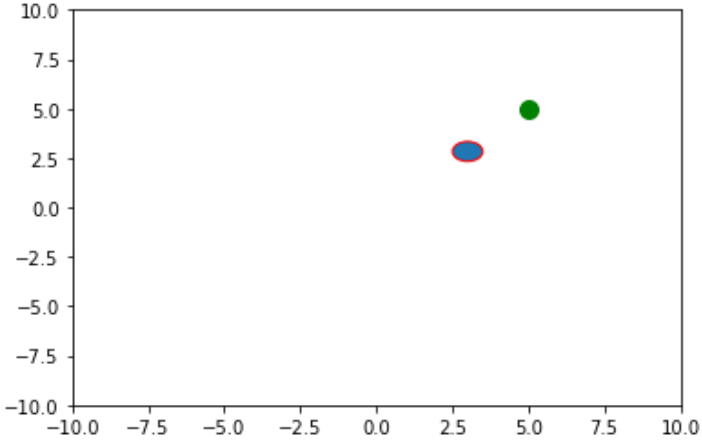
Pathwise Derivative - Toy Problem (cont.)



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

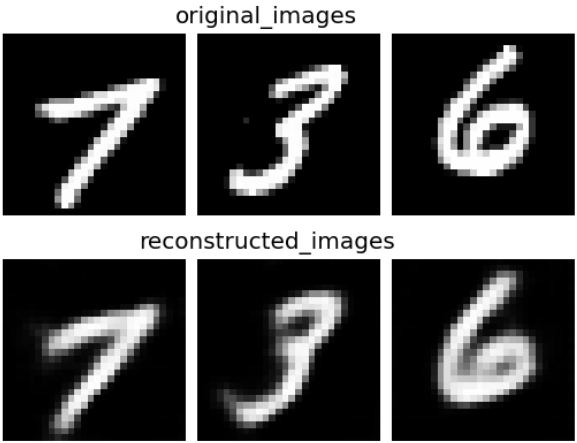


Image reconstruction with variational autoencoders on MNIST digits dataset

Results (cont.)

generated_images

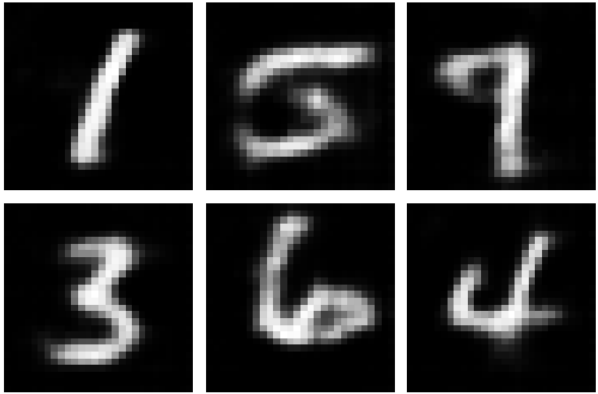
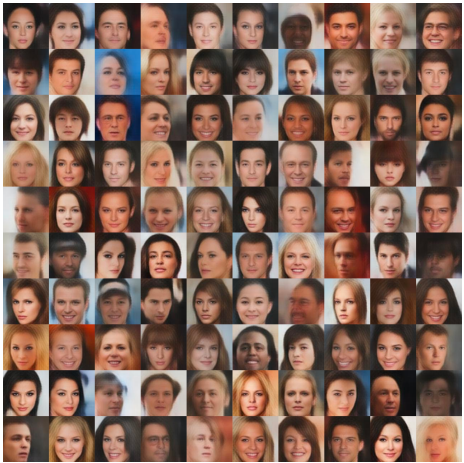


Image generation with variational autoencoders on MNIST digits dataset. Sample an encoding vector from $\mathcal{N}(0, 1)$ and passed it through decoder

Results (cont.)

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

Results (cont.)



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

VAE: **Variants**

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)

LMH
Lady Margaret Hall

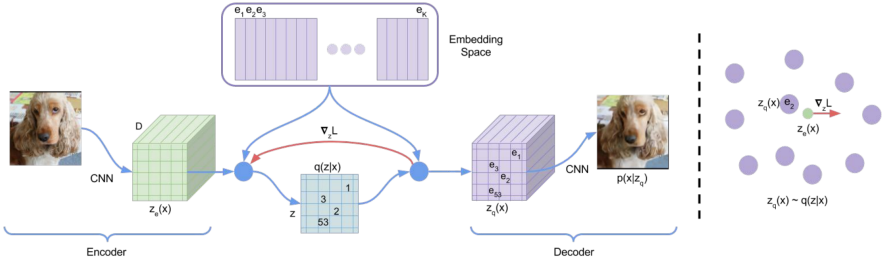
- **Basic Idea**: Different neurons in latent space should be uncorrelated, i.e. they all try to learn something different about input data.
- **Implementation**:

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

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

Figure 3: 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:

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

Vector Quantized - Variational Autoencoders

- 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

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

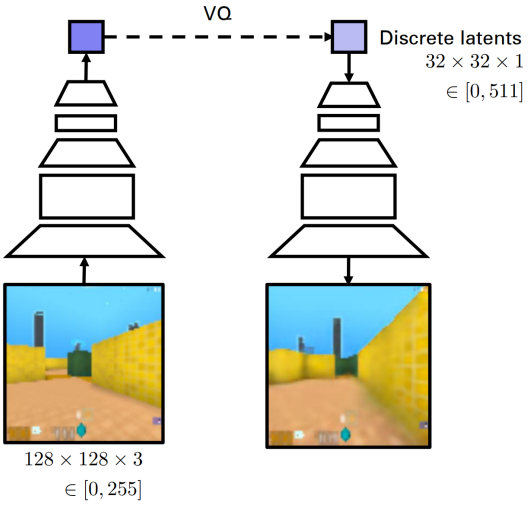
$$L = log\ p(x|z_q(x)) + ||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2$$

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 $z_e(x)$

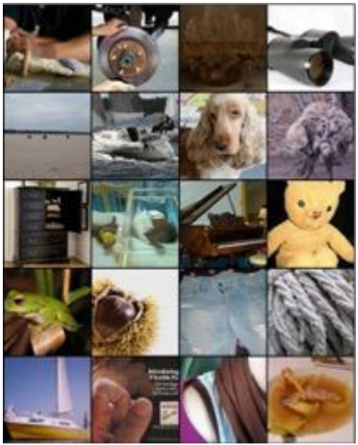
VQ-VAE

- 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

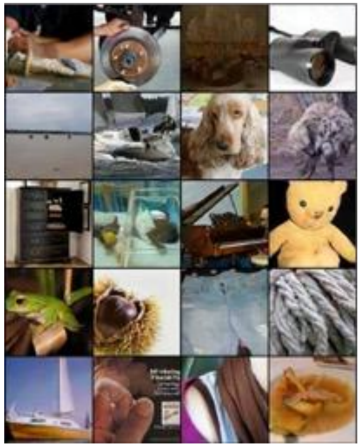


VQ-VAE ImageNet Reconstruction

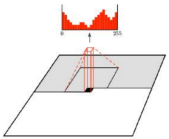
Original 128 x 128 images



Reconstructions



VQ-VAE Sampling



PixelCNN
Prior

Sampled
Discrete latents

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

Decoder

VQ-VAE ImageNet Samples

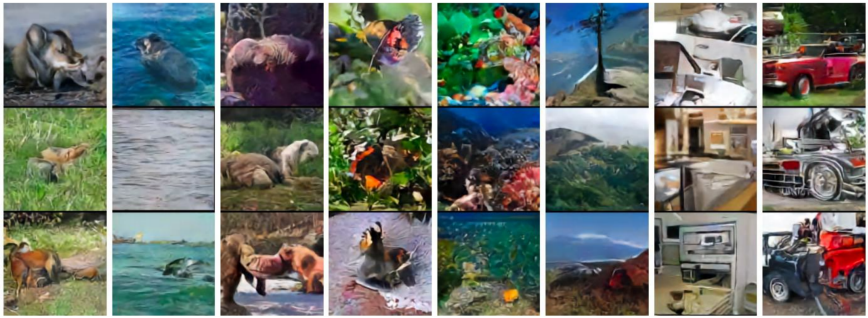
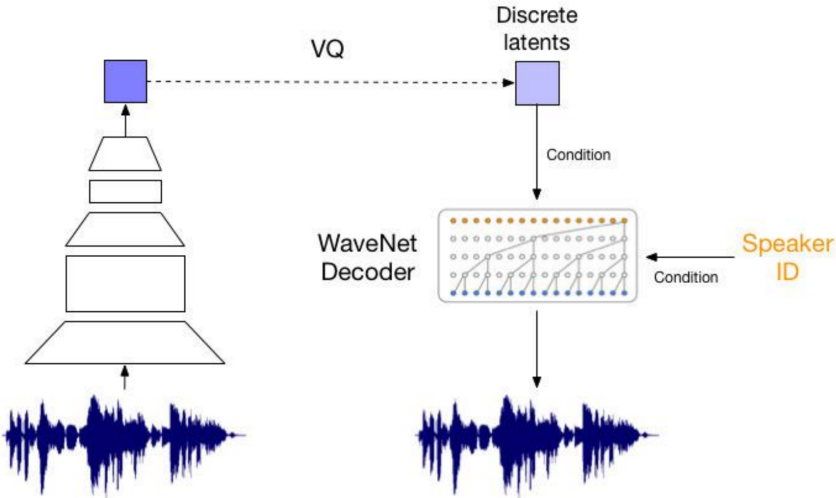


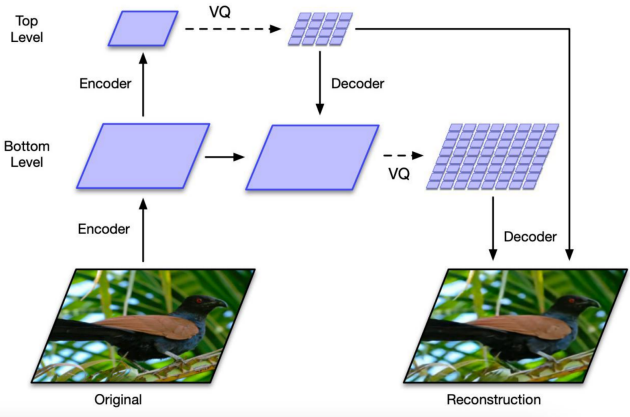
Figure 3: Samples (128x128) from a VQ-VAE with a PixelCNN prior trained on ImageNet images. From left to right: kit fox, gray whale, brown bear, admiral (butterfly), coral reef, alp, microwave, pickup.

VQ-VAE Voice Style Transfer

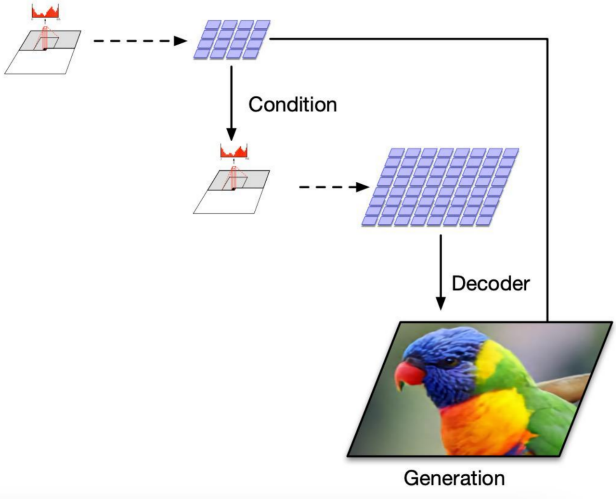


VQ-VAE 2

- 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.

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.

Autoencoders: **References**

References

[1] Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Larochelle, Hugo and Murray, Iain. "Neural autoregressive distribution estimation." In *AISTATS*, 2011, pp. 29–37.

[3] Uria, Benigno, Murray, Iain, and Larochelle, Hugo. "Neural Autoregressive Distribution Estimation." *JMLR*, 17(205):1–37, 2016.

[4] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096-1103).

[5] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

References (cont.)

[6] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

[7] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798-1828.

[8] Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning* (pp. 37-49).

[9] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

uria2013rnade Uria, Benigno, Murray, Iain, and Larochelle, Hugo. "RNADE: The real-valued neural autoregressive density-estimator." In *NeurIPS*, 2013, pp. 2175–2183.

van den Oord, Aaron, Kalchbrenner, Nal, and Kavukcuoglu, Koray. "Pixel Recurrent Neural Networks." In *ICML*, 2016, pp. 1747–1756.

van den Oord, Aaron, Kalchbrenner, Nal, and Kavukcuoglu, Koray. "Conditional Image Generation with PixelCNN Decoders." In *NeurIPS*, 2016, pp. 4790–4798.

van den Oord, Aaron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray. "WaveNet: A Generative Model for Raw Audio." *arXiv preprint arXiv:1609.03499*, 2016.

Salimans, Tim, Karpathy, Andrej, Chen, Xi, and Kingma, Diederik P. "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture

Likelihood and Other Modifications." *arXiv preprint arXiv:1701.05517*, 2017.

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.