Stony Brook University

Generative Models and Variational Auto-Encoders

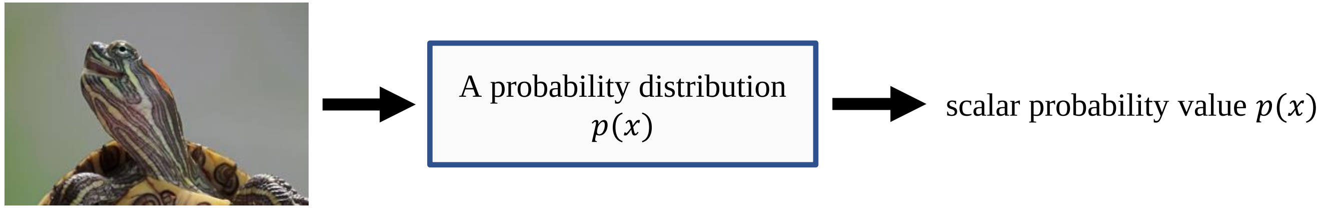
Wenhan Gao

Department of Applied Mathematics and Statistics

FAR BEYOND

Preliminary: Generative Models

A statistical generative model is a **probability distribution** $p(x)$.


A probability distribution $p(x)$
scalar probability value $p(x)$

It is generative because **sampling from $p(x)$ generates new data points**.

......

Preliminary: Generative Models

You often **need some form of control signal** (such as a latent variable z) for generation.

- **High-dimensional data**: Modeling the distribution $p(x)$ directly can be extremely difficult because the space of possible data points is vast and complex, requiring enormous amounts of data and may not generalize well.
- **Latent structure**: Data typically has underlying patterns and features. These latent factors are not directly observable in the data, but using a control signal allows the model to encode these underlying structures.

The data distribution $p(x)$ can be **factorized** through the control signal z. It's often useful to condition on rich information z.

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

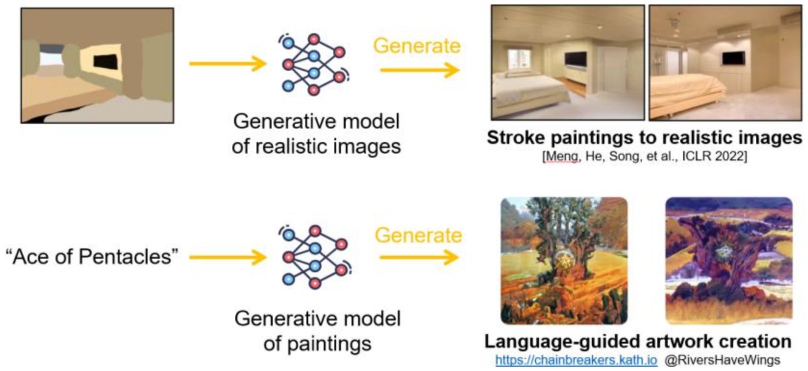
The model splits the task of generating data into two parts: 1) generating the latent variable z, and 2) generating data x conditioned on that z.
This **allows the model to disentangle factors of variation in the data** (e.g., shape, color, orientation) and represent them explicitly in the latent space.

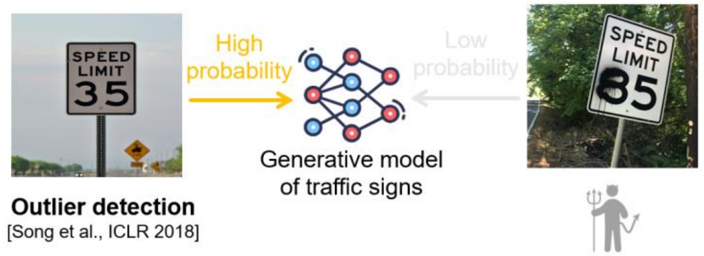
Stony Brook University

Preliminary: Generative Models

With probability distribution $p(x)$, we can

- **Generation**: Getting new samples from $p(x)$.
- **Density Estimation**: $p(x)$ should be high if x is in distribution.
- **Unsupervised Representation Learning**: Learn what images have in common; e.g. ears, hair colors, etc..


Generate
Generative model of realistic images
Stroke paintings to realistic images
[Meng, He, Song, et al., ICLR 2022]
"Ace of Pentacles"
Generate
Generative model of paintings
Language-guided artwork creation
https://chainbreakers.kath.io @RiversHaveWings



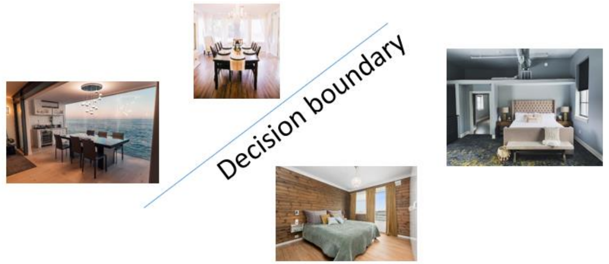
FAR BEYOND

Image: https://deepgenerativemodels.github.io/syllabus.html

Stony Brook University

Preliminary: Discriminative vs. Generative

Discriminative: classify bedroom (B) vs. dining room (D)



We are interested in learning the conditional probability:

$P(Y = Bedroom \mid X =$ $)$

Generative: generate X.

$Y = B, X =$

The input X is **not** given. Requires a model of the **joint distribution over both X and Y.**

We are interested in learning the joint probability:

$P(Y = Bedroom, X =$ $)$

FAR BEYOND

Image: https://deepgenerativemodels.github.io/syllabus.html

Preliminary: Discriminative vs. Generative

Discriminative: Y is simple; X is always given, so not need to model $P(X)$. Therefore, it cannot handle missing data or generate new data.

Generative: We generate new X. Therefore, we need to model the complex distribution $P(X)$.

The conditional probability and the joint probability are related by the Bayes' Rule.

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)} = \frac{P(X,Y)}{P(X)}$$

FAR BEYOND

Preliminary: Distribution of Data

The MNIST consists of grayscale images with pixel values between 0 and 255. We can normalize them to [0,1].

- Each MNIST image **can be represented as a 784-dimensional vector** (28×28 pixels).
- After normalizing, each pixel's value represents the probability that the pixel is on (1) or off (0). We model each pixel as an **independent Bernoulli random variable**.

Each pixel is modeled as an independent Bernoulli random variable, the joint distribution is the product of 784 Bernoulli distributions.

$$p(x) = p(x_1, x_2, \ldots, x_{784} = \prod_{i=1}^{784} p(x_i)$$

Note: Obviously, pixels are not independent, but we make this assumption.

Goal: Learn a probability density $p(x;\ \theta)$ that is close to $p(x)$ that is observed from data. In other words, if we sample from $p(x;\ \theta)$, the sample should look like a digit.

Preliminary: Entropy, Cross Entropy, and KL Divergence

- Entropy $H(p)$ is a measure of the uncertainty in the distribution.
$$H(p) = \mathbb{E}_{X \sim p}[-\log p(X)]$$
    - Non-negativity: $H(p) \geq 0$, with equality if and only if p is a degenerate distribution (all the probability mass is on one outcome).
- Cross-entropy $H(p, q)$ measures the expected number of bits needed to encode data from p using the distribution q.
$$H(p, q) = \mathbb{E}_{X \sim p}[-\log q(X)]$$
    - Non-negativity: Cross-entropy is always non-negative.
    - Asymmetric: Cross-entropy is not symmetric, i.e., $H(p, q) \neq H(q, p)$.
    - Lower Bound: The cross-entropy $H(p, q)$ is greater than or equal to the entropy $H(p)$, i.e., $H(p, q) \geq H(p)$.
    - Equality: $H(p, q) = H(p)$ if and only if $p = q$, i.e., when the distributions are the same.
- KL Divergence $D_{\mathrm{KL}}(p \| q)$: is a measure of how one probability distribution diverges from another.
$$D_{\mathrm{KL}}(p \| q) = \mathbb{E}_{X \sim p}\left[\log \frac{p(X)}{q(X)}\right]$$
    - Non-negativity: $D_{\mathrm{KL}}(p \| q) \geq 0$, with equality if and only if $p = q$. This is a consequence of Jensen's inequality.
    - Asymmetry: KL divergence is not symmetric, meaning $D_{\mathrm{KL}}(p \| q) \neq D_{\mathrm{KL}}(q \| p)$.
    - Relation to Cross-Entropy: The KL divergence can be expressed as the difference between the cross-entropy and the entropy:
$$D_{\mathrm{KL}}(p \| q) = H(p, q) - H(p)$$

Preliminary: Entropy, Cross Entropy, and KL Divergence

All the properties can be proven. Here is an example.

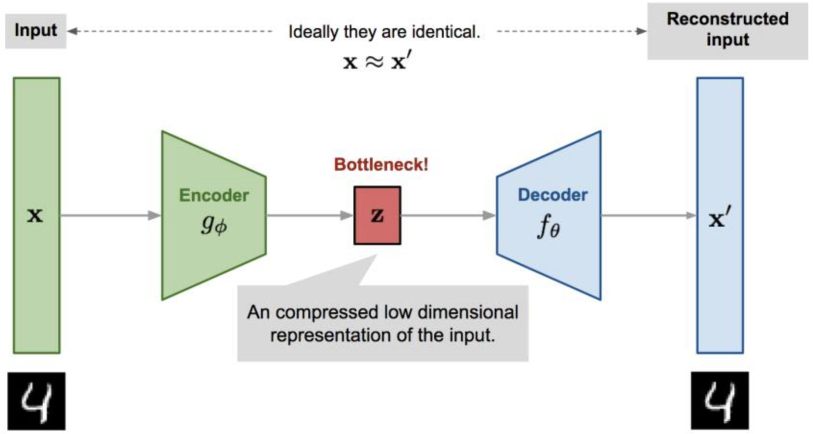
Proposition: $H(p,q) = H(p) + D_{\mathrm{KL}}(p\|q)$.

Proof:

$$\begin{aligned} H(p,q) &= \mathbb{E}_{X\sim p}[-\log q(X)] \\ &= \mathbb{E}_{X\sim p}[-\log p(X) + \log p(X) - \log q(X)] \\ &= \mathbb{E}_{X\sim p}[-\log p(X)] + \mathbb{E}_{X\sim p}[\log p(X) - \log q(X)] \\ &= \mathbb{E}_{X\sim p}[-\log p(X)] + \mathbb{E}_{X\sim p}\left[\log \frac{p(X)}{q(X)}\right] \\ &= H(p) + D_{\mathrm{KL}}(p\|q) \end{aligned}$$

Stony Brook University

Variational Autoencoder



Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

High-level idea:

- Auto-encoder: we do not know how to generate new data.
- Variational AE: Enforce the latent variable to follow some distribution so that **you can generate new samples conditioned on random samples from that distribution**. The loss consists of two parts:
  1) reconstruction loss.
  2) KL divergence between the distribution of your latent distribution and the assumption distribution.

FAR BEYOND

Image: https://lilianweng.github.io/posts/2018-08-12-vae/

VAE Formulation

We are now interested in learning a **probability distribution** $p_\theta(x)$, parameterized by θ. I sometimes write $p(x;\theta)$ instead. I'll drop θ for simplicity.

Defining VAE:

- We aim to model the true data distribution with $p_\theta(x)$ by introducing latent variables z, such that the joint distribution $p_\theta(x, z)$ is:

  $$p_\theta(x, z) = p_\theta(x \mid z) p_\theta(z)$$

  Here, $p_\theta(z)$ is a prior over the latent variables, typically chosen as a simple Gaussian, $p_\theta(x \mid z)$ is the likelihood of the data x given the latent variables z.

  - Given the distributions, we can generate new data in two steps:

    1. First, sample a $z^{(i)}$ from a prior distribution $p_\theta(z)$.
    2. Then a value $x^{(i)}$ is generated from a conditional distribution $p_\theta\left(x \mid z = z^{(i)}\right)$.

Stony Brook University

VAE Formulation

Maximum Marginal Likelihood Learning:

- Note that

$$p_\theta(x) = \int p_\theta(x, z)\, dz = \int p_\theta(x \mid z) p(z)\, dz$$

- The optimal parameter θ^* is the one that maximizes the probability of generating real data samples:

$$\theta^* = \arg\max_\theta \prod_{i=1}^n p_\theta\left(x^{(i)}\right) = \arg\max_\theta \sum_{i=1}^n \log p_\theta\left(x^{(i)}\right),$$

  where

$$p_\theta\left(x^{(i)}\right) = \int p_\theta\left(x^{(i)}, z\right)\, dz = \int p_\theta\left(x^{(i)} \mid z\right) p_\theta(z)\, dz.$$

  This integral is often intractable due to the complex forms of $p_\theta(x \mid z)$ as we cannot check all the possible values of z and sum up. Therefore, we turn to variational inference.

FAR BEYOND

VAE Formulation

Variational Inference:

- Instead of directly marginalizing z, we approximate the true posterior $p_\theta(z \mid x)$ with a variational distribution $q_\phi(z \mid x)$. This distribution is parameterized by a neural network (encoder) and typically takes the form of a Gaussian:
$$q_\phi(z \mid x) = \mathcal{N}(z; \mu(x), \Sigma(x))$$
Here, $\mu(x)$ and $\Sigma(x)$ are outputs of a neural network, i.e., the encoder network. This introduces a variational family of distributions.
- Now the structure looks a lot like an autoencoder:
    - The approximation function $q_\phi(z \mid x)$ is the probabilistic encoder.
    - The conditional probability $p_\theta(x \mid z)$ defines a generative model, similar to the decoder. $p_\theta(x \mid z)$ is also known as probabilistic decoder.

Stony Brook University

VAE Formulation

Evidence Lower Bound (ELBO):

- The marginal likelihood $p_\theta(x)$ can be decomposed using the following identity:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z \mid x)}\right] + \mathrm{KL}(q_\phi(z \mid x) \| p_\theta(z \mid x))$$

  Since the KL divergence $\mathrm{KL}(q_\phi(z \mid x)\|p_\theta(z \mid x)) \geq 0$, this gives a lower bound on $\log p_\theta(x)$:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z \mid x)]$$

  This is the Evidence Lower Bound (ELBO). Expanding it gives:

$$\mathrm{ELBO} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - \mathrm{KL}(q_\phi(z \mid x)\|p_\theta(z))$$

- Note that this lower bound becomes an equality iff $\mathrm{KL}(q_\phi(z \mid x)\|p_\theta(z \mid x)) = 0$

Reparameterization Trick:

- Optimizing the ELBO is intractable as the expectation $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)]$ is intractable to differentiate. To address this, the reparameterization trick is used.
- Instead of directly sampling z from $q_\phi(z \mid x)$, we reparameterize z as a deterministic function of $\mu(x), \Sigma(x)$, and some random noise ϵ drawn from a simple distribution (e.g., a standard normal distribution):

$$z = \mu(x) + \Sigma(x)^{1/2} \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

- This allows backpropagation through the sampling process, making the VAE model differentiable and enabling efficient optimization of the ELBO using gradient-based methods.

FAR BEYOND

Stony Brook University

Recap: The Reparameterization Trick

- The issue arises because the sampled value z is treated as a "**constant**" once sampled, and the gradient of a constant with respect to the distribution's parameters θ is zero.
- When you sample z from a probability distribution $p(z \mid \theta)$, like $z \sim \mathcal{N}\left(\mu, \sigma^2\right)$, the random variable z is drawn based on the parameters $\theta = (\mu, \sigma)$ (mean and standard deviation). The sampling process itself is a discrete event, meaning once the sample z is obtained, it becomes a fixed value.
  - Suppose $\mu = 2, \sigma = 1$, and you sample $z = 2.5$.
  - Once $z = 2.5$ is drawn, it's just a fixed number. There's no obvious relationship left between this fixed value 2.5 and the parameters μ and σ anymore.
  - **The gradient of a constant with respect to any parameter is always zero**:

$$\nabla_\mu z = 0 \quad \text{and} \quad \nabla_\sigma z = 0$$

- The reparameterization trick allows us to **express the random variable z as a deterministic function of the parameters** θ and a separate random variable ϵ.
  - $z = \mu + \sigma \cdot \epsilon, \epsilon \sim \mathcal{N}(0, 1)$

$$\nabla_\mu z = \nabla_\mu(\mu + \sigma \cdot \epsilon) = 1$$
$$\nabla_\sigma z = \nabla_\sigma(\mu + \sigma \cdot \epsilon) = \epsilon$$

- The reparameterization trick effectively decouples the randomness from the parameters, allowing for a gradient-based optimization approach that is both efficient and effective.

FAR BEYOND

VAE Formulation

Final Objective:

- The final objective to maximize is the ELBO, which consists of two parts:
  1. The reconstruction term: $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)]$, typically implemented as a Gaussian loglikelihood or Bernoulli likelihood depending on the data.
  2. The KL divergence term: $\mathrm{KL}(q_\phi(z \mid x) \| p_\theta(z))$, which regularizes the latent space to be close to the prior.
- Thus, the loss function becomes:

$$\mathcal{L} = -\mathrm{ELBO} = \mathbb{E}_{q_\phi(z|x)}[-\log p_\theta(x \mid z)] + \mathrm{KL}(q_\phi(z \mid x) \| p_\theta(z))$$

  This is the loss function that is minimized during VAE training.

Note: The general formula for the KL divergence between two multivariate Gaussians is

$$\mathrm{KL}(q(z) \| p(z)) = \frac{1}{2}\left[\mathrm{Tr}\left(\Sigma_p^{-1}\Sigma_q\right) + (\mu_p - \mu_q)^T \Sigma_p^{-1}(\mu_p - \mu_q) - k + \log\left(\frac{\det(\Sigma_p)}{\det(\Sigma_q)}\right)\right].$$

Assume the prior is $p(z) = \mathcal{N}(0, I)$ and the posterior $q(z \mid x)$ is diagonal Gaussian,

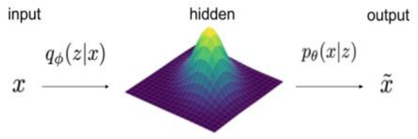
$$\mathrm{KL}(q(z \mid x) \| p(z)) = \frac{1}{2}\sum_{i=1}^{d}\left(\log\left(\sigma_q^2\right) + \frac{1}{\sigma_q^2} + \mu_q^2 - 1\right).$$

Stony Brook University

VAE Formulation

Diagram Overview:

- Encoder: Input $x \rightarrow$ Mean μ and variance $\sigma \rightarrow$ Latent variable z sampled from $q_\phi(z \mid x)$ with reparametrization trick.
- Decoder: Latent variable $z \rightarrow$ Reconstructed output x' from $p_\theta(x \mid z)$.



Latent distribution

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

- The first term, $\mathbb{E}_{q\phi(z|x)}[-\log p_\theta(x \mid z)]$, measures how well the model can reconstruct the input data from the latent space.
  - The VAE takes an input x, encodes it into a latent variable z through the encoder $q_\phi(z \mid x)$, and then reconstructs the input through the decoder $p_\theta(x \mid z)$. The reconstruction loss quantifies the difference between the original input x and the reconstructed version $p_\theta(x \mid z)$.
- The second term $\mathrm{KL}(q_\phi(z \mid x) | p_\theta(z))$, represents the Kullback-Leibler (KL) divergence between the learned latent distribution $q_\phi(z \mid x)$ and the prior distribution $p_\theta(z)$, which is typically a standard normal distribution.
  - This term regularizes the latent space by pushing the learned latent distribution closer to the prior distribution. This allows for better generalization, smoother interpolations, and the ability to sample new data points from the latent distribution.

FAR BEYOND

Image: https://deepgenerativemodels.github.io/syllabus.html

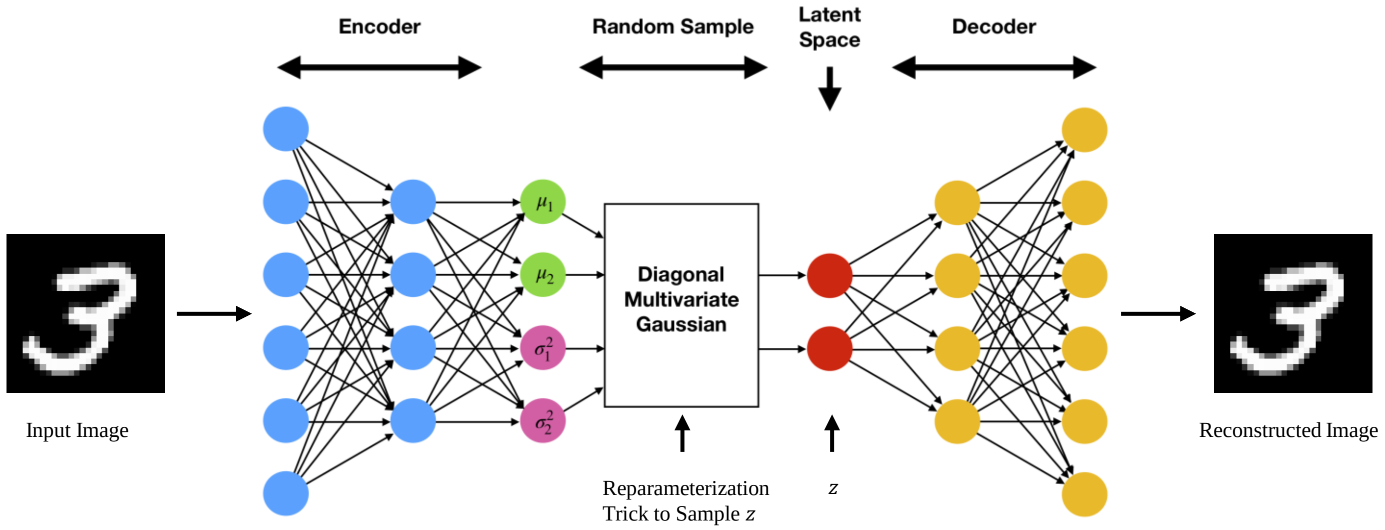


Image made based on: https://deepgenerativemodels.github.io/syllabus.html; https://www.compthree.com/blog/autoencoder/

Stony Brook University

FAR
BEYOND