Lecture 24: Generative Models (VAEs) and Post-Training Compute

Contents

1 Generative Models Revisited

1.1 Problem Statement

The core goal of generative modeling is to understand and sample from an unknown distribution of data, $P(\vec{x})$. Unlike classification or regression, which map inputs to labels, generative models aim to produce new examples (sampling) .

While unconditional generation samples purely from the learned distribution ($z \to x$), practically useful settings often involve **conditional generation** (e.g., prompts in LLMs or text-to-image models), where we sample from $P(x|c)$ given some condition c .

1.2 Approaches that Fail

Before introducing Variational Autoencoders (VAEs), it is instructive to analyze why naive approaches fail.

1.2.1 1. Using a Classifier

One might attempt to generate a "cat" by taking a random noise image and optimizing it to maximize the "cat score" of a pre-trained classifier .

- **Method:** Sample random noise $\vec{x} \sim \mathcal{U}$, then perform gradient ascent on the class score.
- **Result:** This produces noise-like images (adversarial examples) that the classifier confidently labels as "cat", but do not look like natural images .
- **Reason:** The manifold of natural images is extremely low-dimensional within the pixel space. The optimization moves the noise into a region of "adversarial inputs" rather than the natural image manifold .

1.2.2 2. Naive Autoencoders

An autoencoder compresses input $\vec{x}$ into a latent code $\vec{z}$ via an encoder, and reconstructs it via a decoder .

- **Method:** Train an autoencoder. Discard the encoder. Sample $\vec{z} \sim$ Random Distribution and pass it through the decoder .
- **Result:** The output is usually noise or "blurry junk" .
- **Reason:** The encoder maps training data to a very specific, often sparse, low-dimensional subset of the latent space. The decoder has never seen random noise $\vec{z}$ during training. Thus, sampling from a generic distribution (e.g., Gaussian) queries the decoder on undefined regions .

VAE 的出现正是为了解决朴素自编码器的问题：**强制**潜在空间 z 遵循一个**已知**的、**连续**的分布（例如**标准正态分布** $\mathcal{N}(0, I)$），这样我们就能从中随机采样了。

2 Variational Autoencoders (VAEs)

To fix the autoencoder approach, we must force the latent space $\vec{z}$ to follow a known distribution (e.g., a Gaussian) that we can sample from.

2.1 Core Ingredients

VAEs introduce three key modifications to the standard autoencoder :

1. **Stochasticity during Training:** Make $\vec{z}$ random during the training phase, not just inference.
2. **Distribution Regularization:** Add a loss term to force the distribution of $\vec{z}$ produced by the encoder to match a desired prior distribution $P(z)$.
3. **Differentiable Sampling:** Use the "Reparameterization Trick" to allow Stochastic Gradient Descent (SGD) to optimize through the random sampling step .

2.2 The VAE Loss Function

The training objective consists of two competing loss terms :

$$\mathcal{L} = \mathcal{L}_{\text{reconstruction}} + \lambda \cdot \mathcal{L}_{\text{distribution}} \tag{1}$$

为了强制编码器输出的这个 z 分布（记为 $Q(z)$）符合我们想要的**先验分布**（记为 $P(z)$，通常是 $\mathcal{N}(0, I)$），VAE 在损失函数中增加了一个新项，称为 **KL 散度 (KL Divergence)**。

2.2.1 Distribution Loss: KL Divergence

We desire the latent distribution $Q(z)$ to match a prior $P(z)$, typically the standard normal distribution $\mathcal{N}(0, I_k)$. We measure the difference using the **Kullback-Leibler (KL) Divergence** :

$$KL(Q||P) = \int Q(z) \ln \frac{Q(z)}{P(z)} dz \tag{2}$$

We place the desired distribution in the P spot (denominator) to penalize Q heavily if it generates latents that are unlikely under the prior .

Closed Form for Gaussians: If we assume the encoder predicts a mean $\vec{\mu}_q$ and a covariance Σ_q, and our prior is $\mathcal{N}(0, I)$, the KL divergence has a closed-form analytical solution :

$$KL(\mathcal{N}(\vec{\mu}_q, \Sigma_q)||\mathcal{N}(0, I)) = \frac{1}{2}\left[\mathrm{Tr}(\Sigma_q) + \vec{\mu}_q^T \vec{\mu}_q - k - \ln\det(\Sigma_q)\right] \tag{3}$$

where k is the dimensionality of the latent space.

2.2.2 Constraints on Covariance

The encoder needs to output a covariance matrix Σ_q. However, a covariance matrix must be Positive Semi-Definite (PSD) . To ensure this structurally within the network, the encoder outputs a matrix A (often denoted as $\Sigma^{1/2}$), and we define:

$$\Sigma_q = \Sigma_q^{1/2}(\Sigma_q^{1/2})^T \tag{4}$$

This guarantees that Σ_q is PSD .

2.3 The Reparameterization Trick

A naive sampling operation $\vec{z} \sim \mathcal{N}(\vec{\mu}_q, \Sigma_q)$ breaks the gradient flow because we cannot backpropagate through a random node.

To fix this, we reparameterize $\vec{z}$ as a deterministic transformation of a fixed noise source :

1. Sample noise $\vec{v} \sim \mathcal{N}(0, I)$ (no parameters here).
2. Compute $\vec{z}$ as:

$$\vec{z} = \vec{\mu}_q + \Sigma_q^{1/2}\vec{v} \tag{5}$$

Gradient Flow:

- The gradient passes through the addition to $\vec{\mu}_q$.
- The gradient passes through the multiplication to $\Sigma_q^{1/2}$.
- The stochasticity is isolated in $\vec{v}$, which is treated as a fixed input for that specific step of SGD .

2.4 Interpretation: The Information Bottleneck

The VAE loss creates a tradeoff :

- **Reconstruction Loss:** Wants $\vec{z}$ to carry as much specific information about $\vec{x}$ as possible (low noise, specific means).
- **KL Loss:** Wants $\vec{z}$ to carry *no* information about $\vec{x}$ (collapse to $\mathcal{N}(0, I)$).

The KL term enforces an information bottleneck . It forces the "manifold" of latents to have thickness (due to noise injection), ensuring that the decoder becomes robust to small variations in $\vec{z}$.

3 Generalizing Noise Injection

The "trick" of treating stochastic operations as differentiable by passing gradients through the deterministic parameters is broadly applicable beyond VAEs .

3.1 Quantization-Aware Training (QAT)

A major application is compressing models for deployment (e.g., int8 or int4) .

- **Problem:** Quantization (rounding) has zero gradient almost everywhere or is non-differentiable.
- **Solution:** During training, keep high-precision weights but simulate quantization in the forward pass . Treat the quantization error as additive noise .
- **Gradient:** Pass gradients through the operation as if it were an identity function (Straight-Through Estimator) or specifically through the noise-injection formulation.

4 Post-Training and Test-Time Compute

The lecture transitions to modern Large Language Model (LLM) post-training techniques, particularly those focusing on reasoning.

4.1 Supervised Fine-Tuning (SFT)

Standard instruction following uses SFT with a masked cross-entropy loss. The model learns to predict response tokens given prompt tokens .

- **Chain of Thought (CoT):** It was discovered that models solve problems better if they "show their work." This prompted the inclusion of reasoning steps in training data .

4.2 Improving Reasoning: Two Paths

To make models better at solving complex problems, there are two complementary approaches :

1. **Test-Time Compute:** Spend more computational resources during inference (answering) .
2. **Training:** Train the model to intrinsically be better (e.g., RLVR - Reinforcement Learning with Verifiable Rewards) .

4.3 Test-Time Compute Strategies

1. **Pure Prompting:** Asking the model to "Think step by step" or "Be careful" .
2. **Repeated Generation (Best-of-N):**
   - Generate N responses.
   - Take the majority vote (for classification) or the highest likelihood sequence .
3. **Advanced Sampling (Tilting):** Recent research (Karan & Du, Oct 2025) suggests that base models are "smarter than you think" .
   - **Observation:** Models trained with RLVR tend to output sequences that were already high-probability under the base model .
   - **Technique:** Instead of sampling from $P(x)$, sample from a "tilted" distribution proportional to $(P(x))^{\alpha}$ (where $\alpha > 1$) .
   - **Effect:** This amplifies high-likelihood sequences, effectively performing a soft Beam Search, and can recover significant reasoning performance without additional training .