老派模型解析：AE, VAE 与 Diffusion

笔记整理

2025 年 11 月 23 日

目录

1 Autoencoder (AE)

1.1 AE Loss 的由来

目标是最大化 $p_{\theta,\phi}(x)$，也即训练数据 x 通过 encoder 隐空间 z 流出 decoder 之后复原的概率：

$$L_{\text{AE}} = -\,\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] \tag{1}$$

我们这样解释这个 loss：极大似然拆分为

$$\max_\theta \log p_\theta(x) = \max_\theta \log \int p_\theta(x \mid z)\, p(z)\, dz \tag{2}$$

这里面只包含 decoder 的 θ 参数，然而隐空间的分布 $p(z)$ 不知道，所以引入 encoder 参数 ϕ，用 encoder 的条件分布 $q_\phi(z|x)$ 来近似隐空间分布。

基本假设 1： encoder 的条件输出 $q_\phi(z \mid x)$ 是条件高斯分布， ϕ 是 encoder 的参数；

$$q_\phi(z \mid x) = \mathcal{N}\big(z \mid \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x)^2)\big) \tag{3}$$

注意这不算个很强的假设，它其实是混合高斯模型 GMM 的连续化，而熟悉泛函分析的都知道这样最后得到的无条件分布 $q_\phi(z)$ 在概率测度空间里是稠密的。换言之，该假设对隐空间分布没有任何限制。

基本假设 2： decoder 的条件输出 $p_\theta(x \mid z)$ 是条件高斯分布，θ 是 decoder 的参数

$$p_\theta(x \mid z) = \mathcal{N}\big(\hat{x}_\theta(z),\, \sigma^2 I\big) \tag{4}$$

所以 AE 的 loss 引入 ϕ 对真正的极大似然又近似了一层，并不是完全的 MLE。现在可以推出具体表达式了：

$$L_{\text{AE}} = \sum_z \frac{1}{2\sigma^2} \parallel x - \hat{x}^\theta(z) \parallel^2 + \text{const.} \tag{5}$$

怎么对 z 采样呢？直接采样是不行的，因为直接对 $q_\phi(z \mid x)$ 相当于拿上一轮训练好的参数 ϕ 来计算，梯度无法反向传播到 encoder，可以做一个数学上等价的操作：

$$\varepsilon \sim \mathcal{N}(0, I) \qquad \text{and} \qquad z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon \tag{6}$$

这就叫**重参数化技巧**。

至此我们完全搞明白了 AE 的 loss：

$$L_{\text{AE}} = \sum_z \frac{1}{2\sigma^2} \Big\| x - \hat{x}^\theta\big(z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon\big) \Big\|^2 \tag{7}$$

2 Variational Autoencoder (VAE)

2.1 VAE Loss 的由来

我们说了 AE 的 loss 其实是对似然函数做了二次近似，而 VAE 就要对似然函数本身取极大。熟悉统计学习的都知道，此时应该用 ELBO（可参见李航的《统计学习方法》，那里面叫 F-函数）来算 loss：

$$L(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\big[\log p_\theta(x \mid z)\big] + D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p(z)\big) \tag{8}$$

第一项和 AE 一样，不说了。为了算第二项，我们需要比 AE 多做一个假设：隐变量分布 $p(z)$ 是高斯分布 $\mathcal{N}(0, I)$。这可以看成是 VAE 为了优化完整似然函数，对 AE 做出的让步。

这时候就能直接算出：

$$D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p(z)\big) = \tfrac{1}{2}\sum_i \big(\mu_i^2 + \sigma_i^2 - \log\sigma_i^2 - 1\big) \tag{9}$$

所以总的 loss 就是：

$$L_{\mathrm{VAE}} = \sum_{\varepsilon\sim\mathcal{N}(0,I)} \|x - \hat{x}\|^2 + \sum_i \big(\mu_i^2 + \sigma_i^2 - \log\sigma_i^2 - 1\big) \tag{10}$$

2.2 实际的 VAE 怎么做的

2.2.1 SDXL 的 VAE

Stable Diffusion XL 使用了和之前 Stable Diffusion 系列一样的 VAE 结构（KL-f8），但在训练中选择了更大的 Batch-Size（256 vs 9），并且对模型进行指数滑动平均操作（EMA，exponential moving average），EMA 对模型的参数做平均，从而提高性能并增加模型鲁棒性。

SD 2.x VAE 是基于 SD 1.x VAE 微调训练了 Decoder 部分，同时保持 Encoder 部分权重不变，使他们有相同的 Latent 特征分布，所以 SD 1.x 和 SD 2.x 的 VAE 模型是互相兼容的。而 SDXL VAE 是重新从头开始训练的，所以其 Latent 特征分布与之前的两者不同。

由于 Latent 特征分布产生了变化，SDXL VAE 的缩放系数也产生了变化。VAE 在将 Latent 特征送入 U-Net 之前，需要对 Latent 特征进行缩放让其标准差尽量为 1，之前的 Stable Diffusion 系列采用的缩放系数为 0.18215，由于 Stable Diffusion XL 的 VAE 进行了全面的重训练，所以缩放系数重新设置为 **0.13025**。

SDXL 的 VAE 使用的是一种组合损失：

$$L_{\mathrm{total}} = \lambda_{\mathrm{rec}} \|x - \hat{x}\|_1 + \lambda_{\mathrm{perc}} \sum_j w_j \|\phi_j(x) - \phi_j(\hat{x})\|_1 + \lambda_{\mathrm{KL}}\, D_{\mathrm{KL}}\big(q_\phi(z \mid x) \,\|\, p(z)\big) \tag{11}$$

其中：

- 第一项：L1 重构损失（局部细节）
- 第二项：感知损失（全局语义）

- 第三项：KL 正则（潜空间规整）

实际上，KL 项的权重通常非常小（$10^{-8} \sim 10^{-3}$），因为 SDXL 的 VAE 更偏向 **“感知压缩器（perceptual compressor）”**，也就是一个 Autoencoder，而不是 VAE。

> “Calling this model a ‘VAE’ is sort of a misnomer — it’s an encoder with some very slight KL regularization...”
>
> —— Gist

> “Trained on the same dataset as the original Stable Diffusion autoencoder.”
> “Fine-tuned for perceptual fidelity (L1 + LPIPS) on internal LAION-Aesthetics samples.”
>
> —— Hugging Face 官方模型卡

另外我有一个个人理解是这样的 loss 设计使得 SD 的 VAE 潜空间里离正态分布还有相当距离，是一个包含相当多图像信息的潜空间，不然如果真按照标准 VAE 把潜空间训练成高斯分布的话就没法再喂给 Diffusion 的 Encoder 了：接受的全是噪声。

2.2.2 SD3 的 VAE

之前 SD 系列中使用的 VAE 模型是将一个 512×512 的图像编码为 64×64 的 latent 特征，在 8 倍下采样的同时设置 **4 个通道（channel = 4）**。这种情况存在一定的压缩损失，直接影响是对 latent 特征重建时容易产生小物体畸变（例如人眼崩溃、文字扭曲等）。

因此，**SD 3 模型通过提升 latent 通道数**来增强 VAE 的重建能力，提高重建后的图像质量。SD 3 技术报告中对不同**通道数（4、8、16）**的对比实验显示：当设置为 **16 通道**时，VAE 模型的整体性能（FID 指标降低、Perceptual Similarity 指标降低、SSIM 指标提升、PSNR 指标提升）相比 4 通道时有显著提升。因此 SD 3 最终确定使用了 **16 通道的 VAE 模型**。

与此同时，随着 VAE 的通道数增加到 16，扩散模型部分（U-Net 或 DiT）的通道数也需要同步修改：

- 修改扩散模型与 VAE Encoder 衔接的第一层；
- 修改与 VAE Decoder 衔接的最后一层。

虽然这不会显著增加整体参数量，但会提升任务整体的训练难度。另外 83M 的容量不能说明训练简单，事实上仅仅从训练集规模上看，多模态模型的胃口非常之大，这个 83M 的 VAE 恐怕得吃下千万级乃至亿级张高质量图片，更不必提 VAE 具体的训练细节非常繁琐。

研究者正尝试提出新的 **scaling 变量**，例如：

$$L = f(\text{image tokens},\ \text{text tokens},\ \text{alignment entropy}) \tag{12}$$

或者从信息论角度定义：

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{13}$$

因此，**损失与跨模态互信息 $I(\text{image};\text{text})$ 呈幂律关系**。

一些工作（如 *OpenCLIP Scaling Analysis 2023*、*ALIGN 2021 Supplementary*）发现：

- 损失下降与“有效语义样本数”近似幂律；
- 但幂指数比语言模型低很多（约 0.1–0.15，而语言模型约为 0.3）。

结论一句话

- **LLM 的 scaling law 不适用于多模态。**
  因为多模态训练目标不同、信号密度差异大、模态间存在信息瓶颈。
- **对多模态模型而言，“语义覆盖度” 而非“token 数” 才是关键。**
  简单增加样本量不会按幂律提升性能，必须提升语义多样性与跨模态对齐质量。

当通道数从 4 增加到 16 时，SD 3 需要学习和拟合的内容量也随之增加了 4 倍，因此必须提升整体参数规模以增强模型容量（*model capacity*）。

SD 3 论文中的模型通道数与模型容量对比实验结果表明：

- 当模型参数量较小时，16 通道 VAE 的重建效果并不优于 4 通道 VAE；
- 随着模型参数量逐步增大，16 通道 VAE 的重建性能优势开始显现；
- 当模型深度（*depth*）增加到 **22** 时，16 通道的 VAE 性能明显优于 4 通道的 VAE。

2.2.3 FLUX.1 的 VAE

FLUX.1 系列中，FLUX.1 VAE 架构依然继承了 SD 3 VAE 的 8 倍下采样和输入通道数（16）。在 FLUX.1 VAE 输出 Latent 特征，并在 Latent 特征输入扩散模型前，还进行了 **Pack_Latents** 操作，一下子将 Latent 特征通道数提高到 64（$16 \to 64$），换句话说，FLUX.1 系列的扩散模型部分输入通道数为 64，是 SD 3 的四倍。

这也代表 FLUX.1 要学习拟合的内容比起 SD 3 也增加了 4 倍，所以官方大幅增加 FLUX.1 模型的参数量级来提升模型容量。

SD 3 和 FLUX.1 的 Patch 化方法的不同：

1. **SD 3（下采样卷积）：** 想象我们有一个大蛋糕，SD 3 的方法就像用一个方形模具，从蛋糕上切出一个的小方块。在这个过程中，我们提取了蛋糕的部分信息，但是由于进行了压缩，Patch 块的大小变小了，信息会有所丢失。
2. **FLUX.1（通道堆叠）：** FLUX.1 的方法更像是直接把蛋糕的块堆叠起来，不进行任何压缩或者切割。我们仍然保留了蛋糕的所有部分，但是它们不再分布在平面上，而是被一层层堆叠起来，像是三明治的层次。这样一来，蛋糕块的大小没有改变，只是它们的空间位置被重新组织了。

总的来说，相比 SD 3，FLUX.1 将特征 Patch 化操作应用于扩散模型之前。这也表明 FLUX.1 系列模型认可了 SD 3 做出的贡献，并进行了继承与优化。

目前发布的 FLUX.1-dev 和 FLUX.1-schnell 两个版本的 VAE 结构是完全一致的。同时与 SD 3 相比，FLUX.1 VAE 并不是直接沿用 SD 3 的 VAE，而是基于相同结构进行了重新训练，两者的参数权重是不一样的。

3 Diffusion

有了前面的铺垫，diffusion 的 loss 就很好讲了，仍然是 VAE 的 ELBO，但此时 encoder（逐步去噪）的参数是预先固定的，所以就只剩下 MSE，具体对什么对象做 MSE 视情况而定，常见的是 noise 和 v-prediction，这些在数学上都是等价的。

我们主要提一下 diffusion 的 SDE 视角，这也是通向现代生成模型（如 flow matching）之路。

3.1 SDE 看 Diffusion

3.1.1 正向 SDE（forward SDE）

一般形式（参考 Song et al., 2021）：

$$dx = f(x,t)\,dt \;+\; g(t)\,dW_t, \qquad t \in [0,T], \tag{14}$$

其中 W_t 为标准维纳过程，f 为漂移项，g 控制噪声强度。

常见实例：

- **VP（variance preserving）：**

$$dx = -\tfrac{1}{2}\beta(t)\,x\,dt \;+\; \sqrt{\beta(t)}\,dW_t. \tag{15}$$

- **VE（variance exploding）：**

$$dx = 0 \cdot dt \;+\; \sqrt{\tfrac{d\,\sigma^2(t)}{dt}}\,dW_t. \tag{16}$$

3.1.2 逆向 SDE（reverse-time SDE）

令 $p_t(x)$ 为正向过程在时刻 t 的边缘密度，$\nabla_x \log p_t(x)$ 为其 **score**。则反向时间（从 $T \to 0$）的 SDE 为：

$$dx = \big[\, f(x,t) \;-\; g(t)^2\,\nabla_x \log p_t(x) \,\big]\,dt \;+\; g(t)\,d\bar{W}_t, \tag{17}$$

其中 $d\bar{W}_t$ 是反向时间的维纳增量（与正向独立，同分布）。

在实践中用 $s_\theta(x,t) \approx \nabla_x \log p_t(x)$ 进行替换。此时的 MSE 损失正是噪声方差倒数加权的 BLEU 版 MSE，所以两种理解有内在统一性。

3.1.3 概率流 ODE（probability flow ODE）

由 Fokker-Planck 方程可以推出与上述随机过程等价的确定性常微分方程（产生相同的边缘分布族 $\{p_t\}$）：

$$\frac{dx}{dt} = f(x,t) \; - \; \tfrac{1}{2}\, g(t)^2\, \nabla_x \log p_t(x). \tag{18}$$

采样时用 $s_\theta(x,t)$ 近似 score，即

$$\frac{dx}{dt} \approx f(x,t) \; - \; \tfrac{1}{2}\, g(t)^2\, s_\theta(x,t). \tag{19}$$

推理时就用不同的 ODE 求解器解这个 ODE。

3.2 实际的 U-Net 架构

3.2.1 扩散模型中的 U-Net 结构（以 UNet2DConditionModel 为例）

扩散模型中的 **U-Net** 通常分为三个主要部分：

- **Encoder（下采样块）**
- **Middle Block（中间块）**
- **Decoder（上采样块）**

各部分结构说明 在这三个部分中：

- 每个 block 都包含若干 **ResBlock（残差卷积）**；
- 有的 block 还包含 **Cross-Attention 层**；
- 每个 block 都会接受 **时间步（timestep）embedding** 的调制。

这些模块协同作用，使得模型能够在多分辨率特征层面融合时序与语义信息。

3.2.2 为什么不在所有层都加 Cross-Attention？

原因如下：

1. **早期层（高分辨率）：** 主要负责低级纹理与边缘等细节特征；
2. **深层（低分辨率）：** 主要负责全局语义与结构特征；
3. **计算开销大：** Cross-Attention 消耗显著，不必在所有层都使用。

实践经验

只在中高语义层注入 Cross-Attention 的语义信息，既能保证生成质量，又能显著提高计算效率。

U-Net 在结构和参数量上都碾压 VAE，VAE 根本没有时间嵌入和交叉注意力，只有 CNN 和自注意力，就是个很简单的无条件图像生成器。

SDXL Refiner 模型的训练逻辑与 SDXL Base 一样，不过 Refiner 模型只在前 200 个 Timesteps 上训练（设置的 noise level 较低）。