Sora的前世今生：从文生图到文生视频

✍️ 腾讯程序员 腾讯技术工程 *2024-02-22 17:28 广东*

原文地址：https://mp.weixin.qq.com/s/dIAUwc33IZMid9gVB5uTJg

作者：monychen

> 在2月16日凌晨，OpenAI首款文本生成视频模型Sora正式亮相，迅速在网络上引发广泛关注。对于Sora背后的技术原理，网络上已经充斥着各种分析和猜测，其中大多数分析都是从技术报告入手，对于普通读者来说难度相对较高。为了使技术原理更加通俗易懂，本文将从文本生成图像到文本生成视频的技术演进角度进行剖析，解读从AE、VAE、DDPM、LDM到DiT和Sora的技术发展路线，旨在为读者提供一条清晰简明的技术进化路径。

1.背景

最近AI圈内乃至整个科技圈最爆的新闻莫过于OpenAI的Sora了，感觉热度甚至远超之前ChatGPT发布时的热度。OpenAI也是放出了Sora的技术报告（有一定的信息量，也留下了大量的想象空间）。

技术报告传送门： https://openai.com/research/video-generation-models-as-world-simulators

今天就来尝试聊一下Sora的前世今生，欢迎交流讨论批评指正！

无论是文生图还是文生视频，很多这方面的工作其实都可以看成是自编码器的进阶版本，让我们从自编码器开始入手。

自编码器（Autoencoder）：压缩大于生成

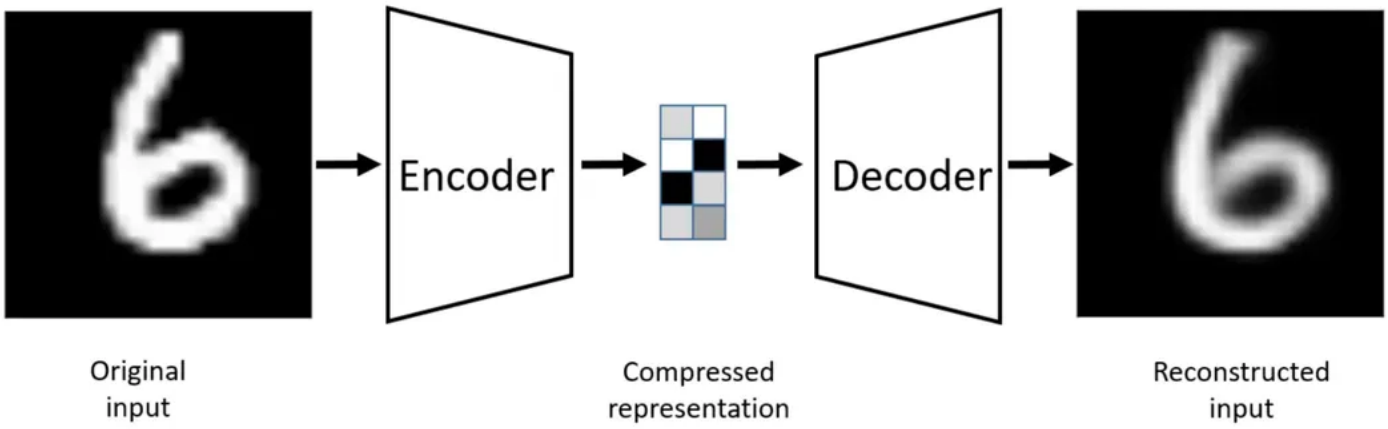
自编码器由编码器和解码器两个部分构成

编码器负责学习输入到编码的映射 $(\cdot)$ ，将高维输入（例如图片）转化为低维编码

$$z = e(x)$$

解码器则学习编码到输出的映射 $(\cdot)$ ，将这些低维编码还原为高维输出（例如重构的图片）

$$\hat{x} = d(z)$$



我们希望压缩前和还原后的向量尽可能相似（比如让它们的平均平方误差MSE尽可能小）

$$\text{loss} = ||x - \hat{x}||^2 = ||x - d(z)||^2 = ||x - d(e(x))||^2$$

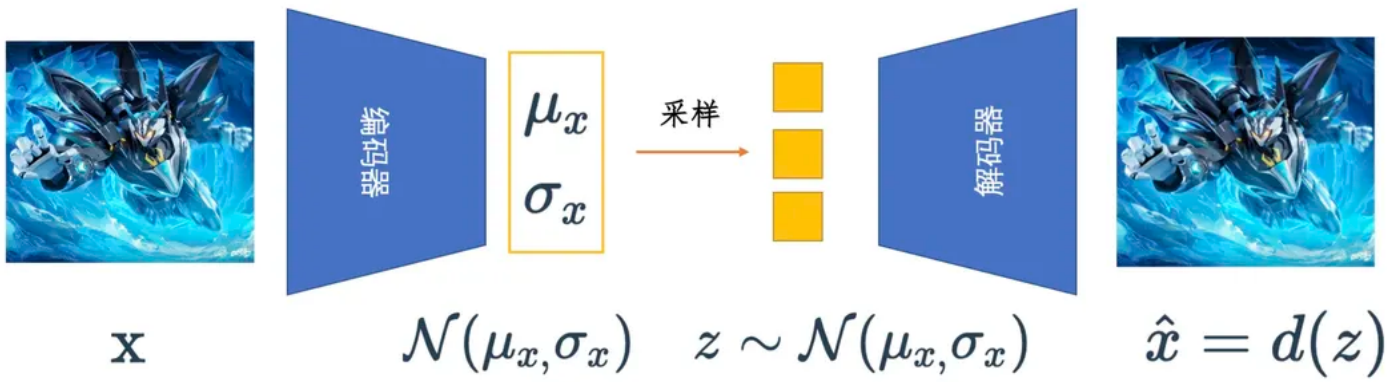
这样就能让神经网络学会使用低维编码表征原始的高维向量（例如图片）。

那有了自编码器能够做生成吗，答案是可以的，只要留下解码器，我随便喂入一个低维编码，不就能够得到一个生成的高维向量了（例如图片）。（当然，这里可能生成一起奇奇怪怪的东西，因为这里对低维向量没有做出约束，自编码器通过将图片转化为**数值编码**再还原回图片的过程，可能导致对训练数据过**拟合**。结果就是，对于未见过的低维编码，解码器重构的图片质量通常不佳。因此**自编码器更多用于数据压缩**。）

变分自编码器（Variational Autoencoders）：迈向更鲁棒地生成

自编码器不擅长图片生成是因为过拟合，那如果能够解决过拟合问题，不就能拿来生成图片了！

变分自编码器做的就是这么一件事：既然自编码器将图片编码为**确定性的数值编码**会导致过拟合，变分自编码器就将图片编码为一个**具有随机性的概率分布**，比如标准正态分布。 这样当模型训练好后，我们只要给解码器喂入**采样自标准正态分布的低维向量**，就能够生成较为“真实”的图片了。



因此，变分自编码器除了希望编码前和解码后的样本尽可能相似（MSE尽可能小），还希望用于解码的数据服从标准正态分布，也就是**低维编码的分布和标准正态分布的KL散度尽可能小**，损失函数加上这么一项约束。

$$\text{loss} = \|x - \hat{x}\|^2 + \text{KL}[\text{N}(\mu_x, \sigma_x), \text{N}(0,1)] = \|x - d(z)\|^2 + \text{KL}[\text{N}(\mu_x, \sigma_x), \text{N}(0,1$$

> 变分自编码器中涉及到了从标准正态分布中进行采样，因此作者引入了一种参数化的技巧（也是为了误差能够反向传播），感兴趣的可以进一步了解。
>
> 此外，VQ-VAE、VQ-GAN 也是一些值得学习的工作。由于不影响对后续的理解，这里不再进行赘述。

变分自编码器减轻了自编码器过拟合的问题，也确实能够用来做图片的生成了，但是大家会发现用**它生成图片通常会比较模糊**。可以这样想一下，变分自编码器的编码器和解码器都是一个神经网络，编码过程和解码过程都是**一步就到位了**，一步到位可能带来的问题就是建模概率分布的能力有限/或者说能够对图片生成过程施加的约束是有限的/或者说“可控性”是比较低的。

去噪扩散概率模型（DDPM）：慢工出细活

既然变分自编码器一步到位的编解码方式可能导致生成图片的效果不太理想，DDPM就考虑拆成多步来做，它将编码过程和解码过程分解为多步：

编码过程

$$x = x_0 \to x_1 \to x_2 \to \cdots \to x_{T-1} \to x_T = z$$

解码过程

$$z = x_T \to x_{T-1} \to x_{T-2} \to \cdots \to x_1 \to x_0 = x$$

因此，所谓的扩散模型由两个阶段构成

- 前向扩散过程，比如给定一张照片，不断（也就是多步）往图片上添加噪声，直到最后这样图片看上去什么都不是（就是个纯噪声）
- 反向去噪过程，给定噪声，不断执行去噪的这一操作，最终得到一张“真实好看”的照片

下图中从右往左就是前向扩散的过程，从左往右就是反向去噪的过程

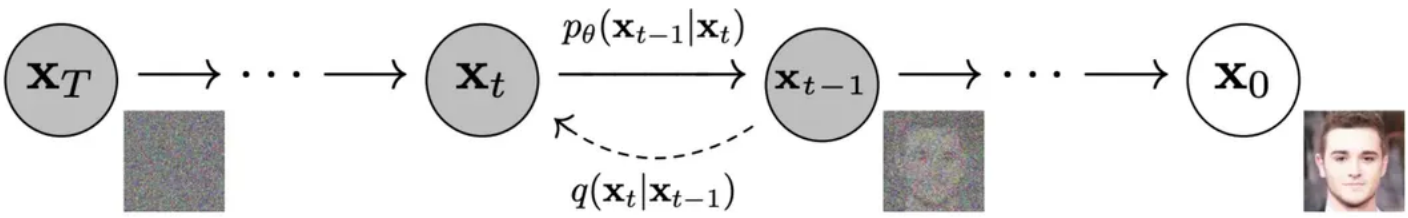


Figure 2: The directed graphical model considered in this work.

可以看到每一步的下标有一个t，t 的范围从0到T，$t=0$代表从数据集中采样出一张图片x_0，前向过程在每一步从高斯分布中采样出噪声叠加到$t-1$时刻的图像上，当T足够大时，最终会得到一个各向同性的高斯分布x_T（球形高斯分布，也就是各个方向方差都一样的多维高斯分布)。

> 引用一个苏剑林大佬的比喻帮助大家理解：我们可以将扩散过程想象为建楼，其中随机噪声是砖瓦水泥等原材料，样本数据是高楼大厦，所以生成模型就是一支用原材料建设高楼大厦的施工队。这个过程肯定很难的，但俗话说“破坏容易建设难”，建楼你不会，拆楼你总会了吧？我们考虑将高楼大厦一步步地拆为砖瓦水泥的过程，当我们有了“拆楼”的中间过程x_0 , x_1 ,... x_T 后，我们知道拆楼的每一步是如何完成的，那反过来不就是建楼的一步？如果我们能学会两者之间的变换关系，那么从X_T出发，反复地执行一步建楼的过程，最终不就能造出高楼大厦x_0出来？

DDPM通过多步迭代生成得到图片缓解了变分自编码器生成图片模糊的问题，但是由**于多步去噪过程需要对同一尺寸的图片数据进行操作**，也就导致了越大的图片需要的计算资源越多（原来只要处理一次，现在**有几步就要处理几次**）。

DDPM细节推导（选读）

下面从数学一点的角度对DDPM进行介绍，不感兴趣的同学可以直接跳过

假设原始数据的分布为$q(x_0)$，从数据集中拿出一张图片就类似于从原始数据分布中进行采样$x_0 \sim q(x_0)$，进一步可以定义第t步的前向扩散过程为

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

其中β_t是第t步的方差，$0 < \beta_1 < \beta_2 < \ldots < \beta_T < 1$且服从一个指定的变化规律（schedule），比如可以是线性变化、二次变换、或者按照正余弦的方式进行变化（有点类似于学习率变化的那种意思）。

有了x_{t-1}后，根据前向扩散过程的条件分布，我们可以通过从高斯分布中采样得到x_t

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon$$

其中$\epsilon \sim \mathcal{N}(0, I)$.

PS：为什么均值前面有个系数？其实是为了保证条件概率具有单位方差。

根据前向扩散这个条件分布，我们还可以将其一步一步展开来看x_t和x_0的关系，比如将

$$x_{t-1} = \sqrt{1-\beta_{t-1}}x_{t-2} + \sqrt{\beta_{t-1}}\epsilon$$

代入

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon$$

可以得到x_t和x_{t-2}的关系。进一步如果将每一步的扩散过程不过代入消元，最终可以得到

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

其中$\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_t, \alpha_t := 1 - \beta_t$，这一性质也被称为“nice property”。

有了上述前向扩散过程的定义和性质，给定数据x_0，就可以一步步叠加噪声得到

$x_1, \dots x_T$，其中x_T为纯噪声（也可以直接根据nice property得到任意时刻的x_t）；而反向去噪的过程则是从噪声中采样一个x_T出来，然后根据某个$p(x_{t-1}|x_t)$一步步去噪，最终得到x_0。回顾一下前面叠加噪声（或者说拆楼）的过程，我们可以得到许多的样本对(x_{t-1}, x_t)，那么“建楼”自然就是从这些数据对中学习一个从x_t到x_{t-1}的模型，也可以表示为概率分布$p(x_{t-1}|x_t)$，又因为$p(x_{t-1}|x_t)$这个模型是未知的，所以可以考虑采用神经网络来近似它！

神经网络可以被看作一个函数近似器，也就是输入为x_t的函数，记作$\mu(x_t)$，建楼的过程就是希望$\mu(x_t)$和x_{t-1}尽可能接近，比如说他们的欧式距离尽可能小

$$||x_{t-1} - \mu(x_t)||^2$$

不过直接让神经网络直接从x_t来预测x_{t-1}难度也还是比较大的，是不是可以考虑简化一下？回顾之前在“拆楼”的过程中其实已经具备从x_{t-1}到x_t的一个关系表示

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$$

那也就可以通过移项然后做个除法得到

$$x_{t-1} = g_1 x_t - g_2 \epsilon_t$$

这里g_1, g_2就是个系数，是什么先不管，但是可以认为这是从x_t到x_{t-1}的一个表达式，也能够启发算法的设计，如果把右边看作$\mu(x_t)$，由于x_t是确定的，那么$\mu(x_t)$可以表示为

$$\mu(x_t) = \epsilon_\theta(x_t, t) + C$$

其中C类似于一个确定项，是不是也意味着我们只需要去预测t时刻的噪声就可以了！ 接着把神经网络的预测代入希望最小化的损失函数中，化简可以得到

$$||x_{t-1} - \mu(x_t)||^2 = ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon, t)||^2$$

综上，算法的训练流程可以表示为

Algorithm 1 Training

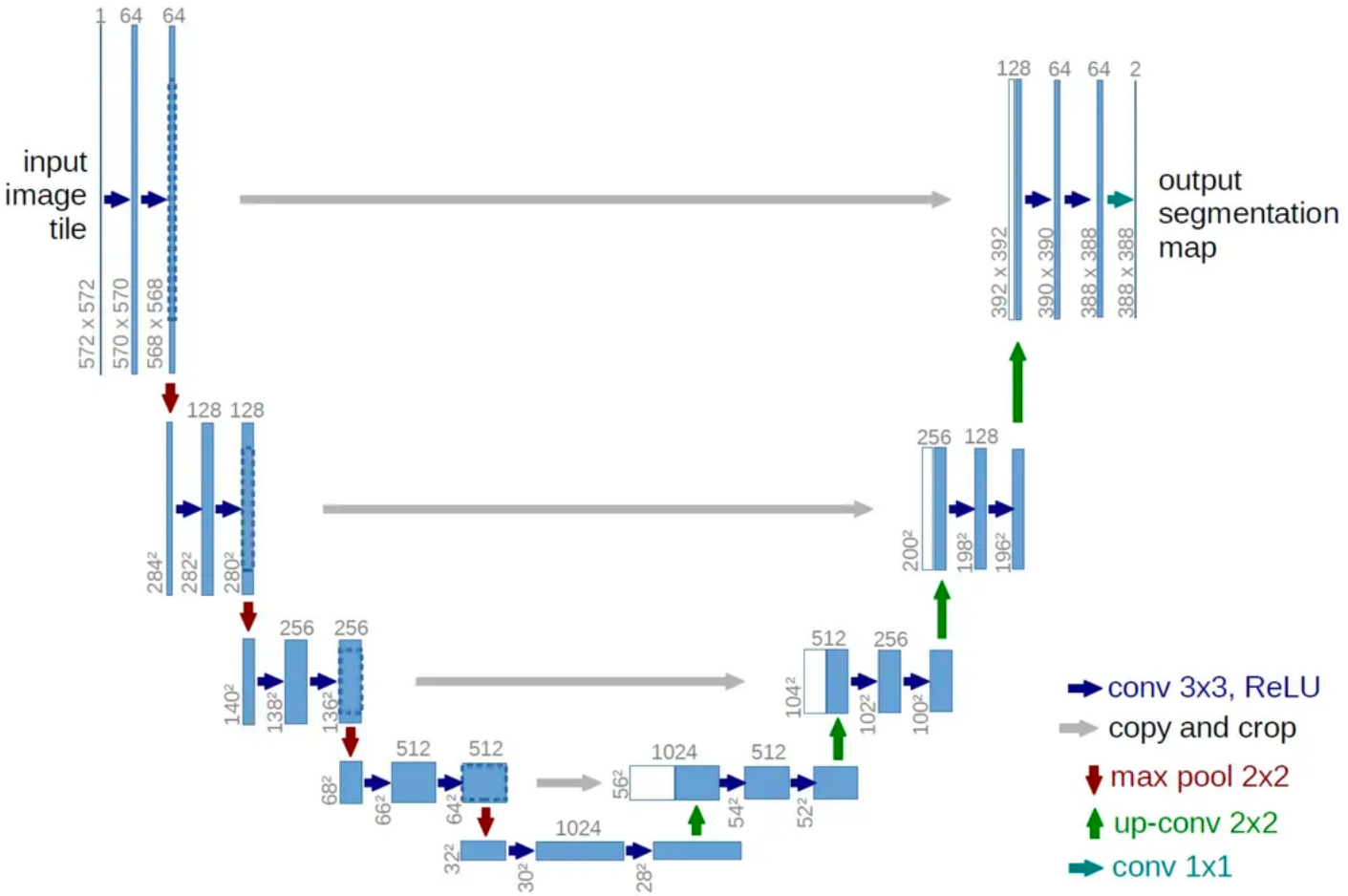
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

算法流程

- 随机从数据集中采样图片
- 随机采样一个步数
- 从高斯分布中采样一个噪声并通过nice property得到t步后含噪的图片x_t
- 神经网络根据x_t预测噪声，计算误差，反向传播

PS：在原论文中作者把方差给固定住了，仅学习均值部分来实现去噪的效果（当然，后续也有改进论文提出把方差也一起拿来学习）。

知道这个网络怎么训练了，我们再来思考一下网络的结构该如何设计。由于网络的输入是叠加噪声后的图片x_t（以及时间步长t），输出为噪声的预测$\hat{\epsilon}_t$，也就是输入和输出的维度其实是一样的，那么只要满足输入输出维度一样的神经网络结构都可以拿来一试（比如自编码器、U-net等），作者这边采用的则是U-net



> U-net提出的初衷是为了解决医学图像分割的问题，其实整体来看，这个也是一个Encoder-Decoder的结构。这个结构先对图片进行卷积和池化，在U-net论文中是池化4次，比方说一开始的图片是224x224的，那么就会变成112x112，56x56,28x28,14x14四个不同尺寸的特征。然后我们对14x14的特征图做上采样或者反卷积，得到28x28的特征图，这个28x28的特征图与之前的28x28的特征图进行通道上的拼接，然后再对拼接之后的特征图做卷积和上采样，得到56x56的特征图，再与之前的56x56的特征拼接，卷积，再上采样，经过四次上采样可以得到一个与输入图像尺寸相同的224x224的预测结果。

对于时间步长t，可以将其转化为time embedding（比如类似position embedding的做法），然后叠加到（U-net的各层）图片上。

模型结构也确定了，怎么训练也知道了，那么训练完后该如何生成图片呢？图片的生成过程也称为采样的过程，流程如下

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

反向去噪过程：从高斯分布中采样一个噪声，从T时刻开始，每次通过训练好的神经网络预测噪声，然后得到略微去噪后的图片，再代入神经网络预测噪声，不断迭代，最终得到第0步的图片。

潜在扩散模型（LDM）：借助VAE来降本增效

既然DDPM在原始空间（像素级别的图片）进行扩散模型的训练（hundreds of GPU days）和采样比较费资源的，那我们就可以考虑把它降维后在latent space进行操作。因此Latent Diffusion Models的做法就是安排上一个自编码器来对原始图片进行降维，比如原来的图片是512*512的，降成64*64后再来进行DDPM这一套流程，生成图片时则是通过扩散模型拿到64*64大小的图片，再通过自编码器的解码器还原为512*512的图片。

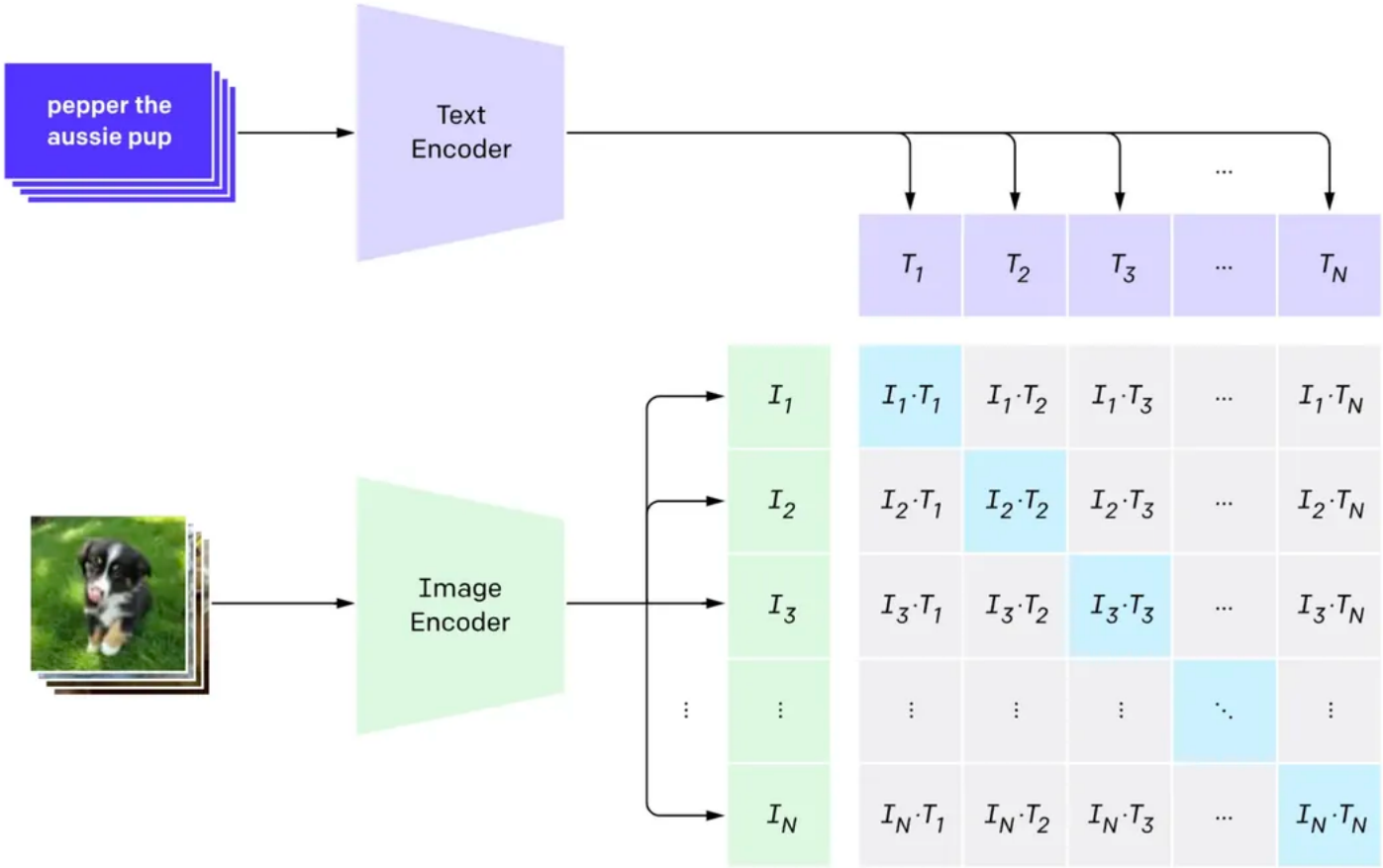
LDM文生图（选读）

下面关于LDM的一些细节介绍，不感兴趣的同学可以直接跳过。

理解了latent的含义后，接下来再考虑下如何根据文本来生成图片呢？ 既然要接收文本，那就需要给模型安排上文本编码器（text encoder），**把文本转化为模型能够理解的东西**。Stable Diffusion采用了CLIP的文本编码器，它的输入是一段文本，输出是77个token的embeddings向量，每个向量的维度为768（可以理解为一段话最多保留77个字（或词），每个字（或词）用768维的向量表示）。

> CLIP是在400million的图片和标题构成的数据集（通过搜索的方式，查询了50w个queries，且每个query保留最多2w条（image，text）数据）上进行训练得到的，作者采用了对比学习的方式，只需要预测文本的整体和图像的成对关系，而不需要预测出具体文本的每一个字。假设一个批次有N对（图像，文本）对，可以有 $N \times N$ 种组合方式，对比学习把原始数据集中的N个组合作为正样本（下图对角线），把其他的 $N \times N - N$ 种组合作为负样本（下图非对角线）有了正负样本后，就可以按照分类的思路来训练，比如每一行就是个N分类问题，其中label就是真实组合所在位置，比如第一行的label是0，第二行的label是1，以此类推。

1. Contrastive pre-training



得到文本的表示后，就需要考虑如何在原来的U-net里叠加上文本的信息呢？ 原来的U-net的输入由两部分组成

- 加噪后的图片
- 时间步长（t）

现在则是由三部分组成

- 加噪后的图片
- 时间步长（t）
- 文本的token embedding

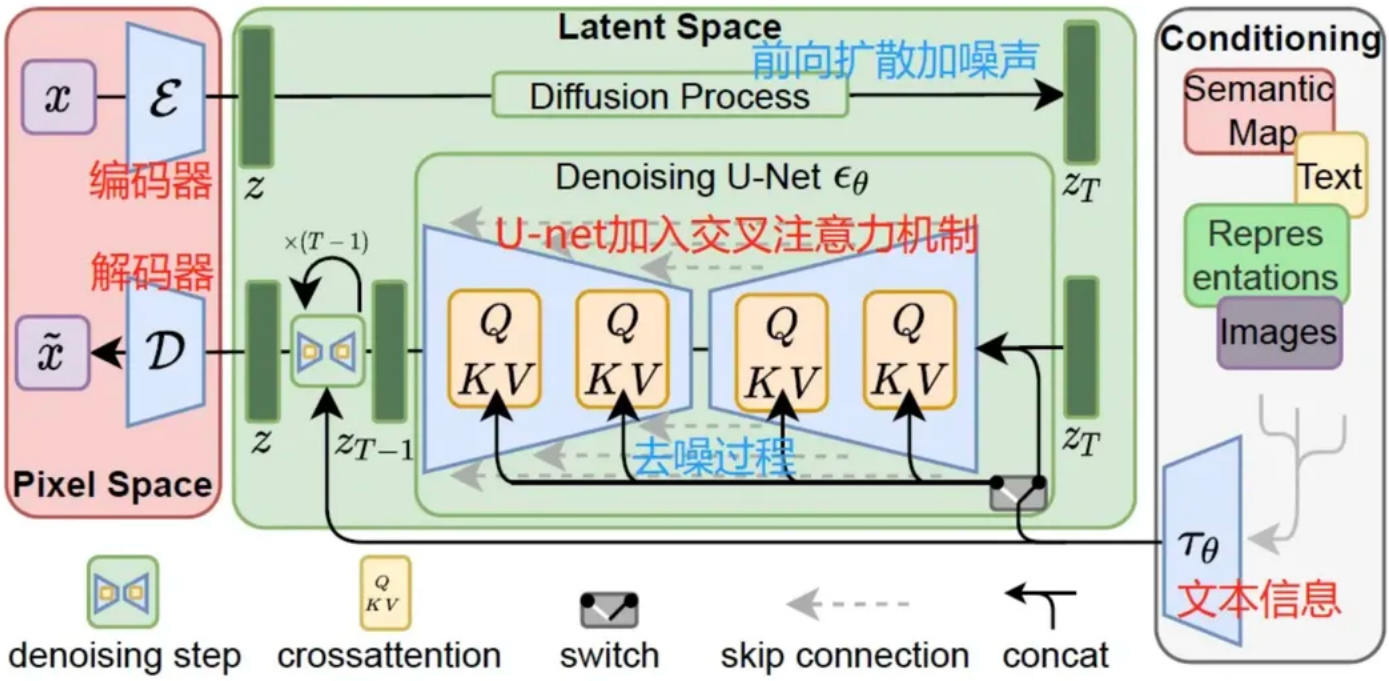
为了能够充分利用文本的信息，让模型按照文本的意思来绘画，Stable Diffusion提出了一种文本和图片之间的交叉注意力机制，具体的

$$Attention(Q,K,V)=softmax\left(QK^T/\sqrt{d}\right)\cdot V$$

其中，Q来自于图像侧，K和V来自于文本侧；可以理解为**计算图像和文本的相似度，然后根据这个相似度作为系数对文本进行加权。** 在前面U-net的基础上，每个resnet之后接上这样一个注意力机制就得到了注入文本信息的U-net的结构了。

然后就可以愉快地像之前的DDPM一样，**预测噪声，最小化MSE**，学习模型了！

最后再来一张整体的结构图总结一下



Diffusion Transformers（DiT）：当扩散模型遇到Transformer

LDM的扩散模型使用了U-net这一网络结构，但这个结构会是最佳的吗？

参考其他领域或者任务的经验，比如去年火了一整年的大语言模型、多模态大模型绝大部分用的都是Transformer结构，相比于U-net，**Transformer结构的Scaling能力**（模型参数量越大，性能越强）更受大家认可。因此，**DiT其实就是把LDM中的U-net替换成了Transformer**，并在Vision Transformer模块的基础上做了略微的修改使得在图片生成过程能够接受一些额外的信息，比如时间步t，标签y。

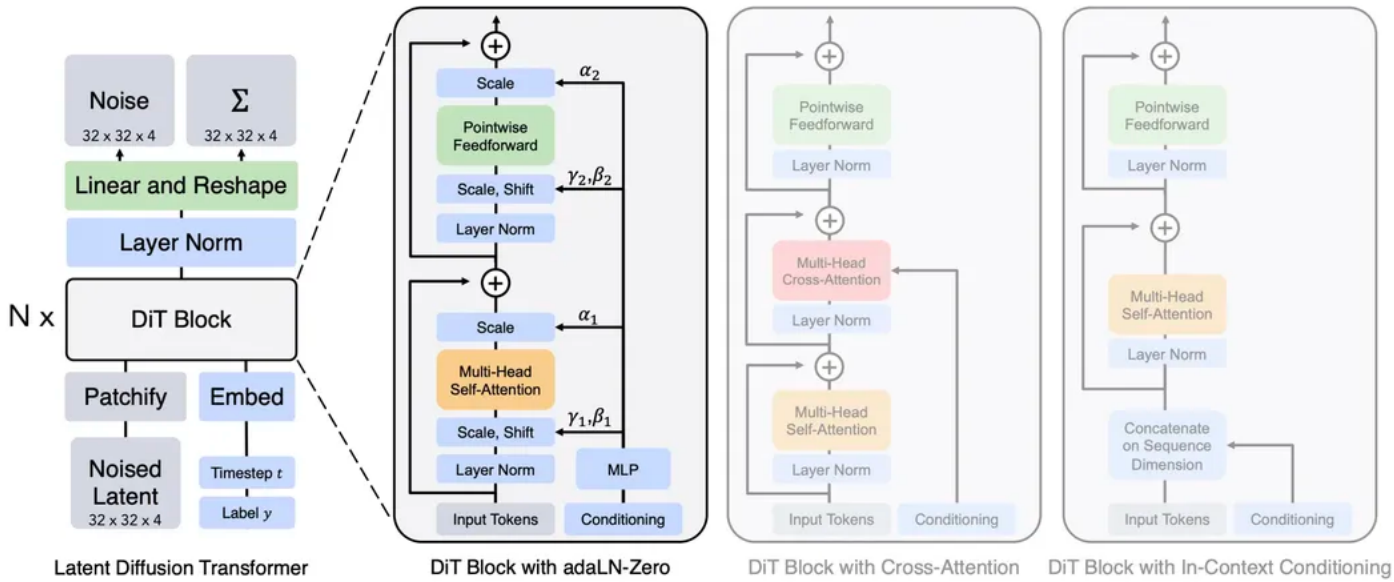
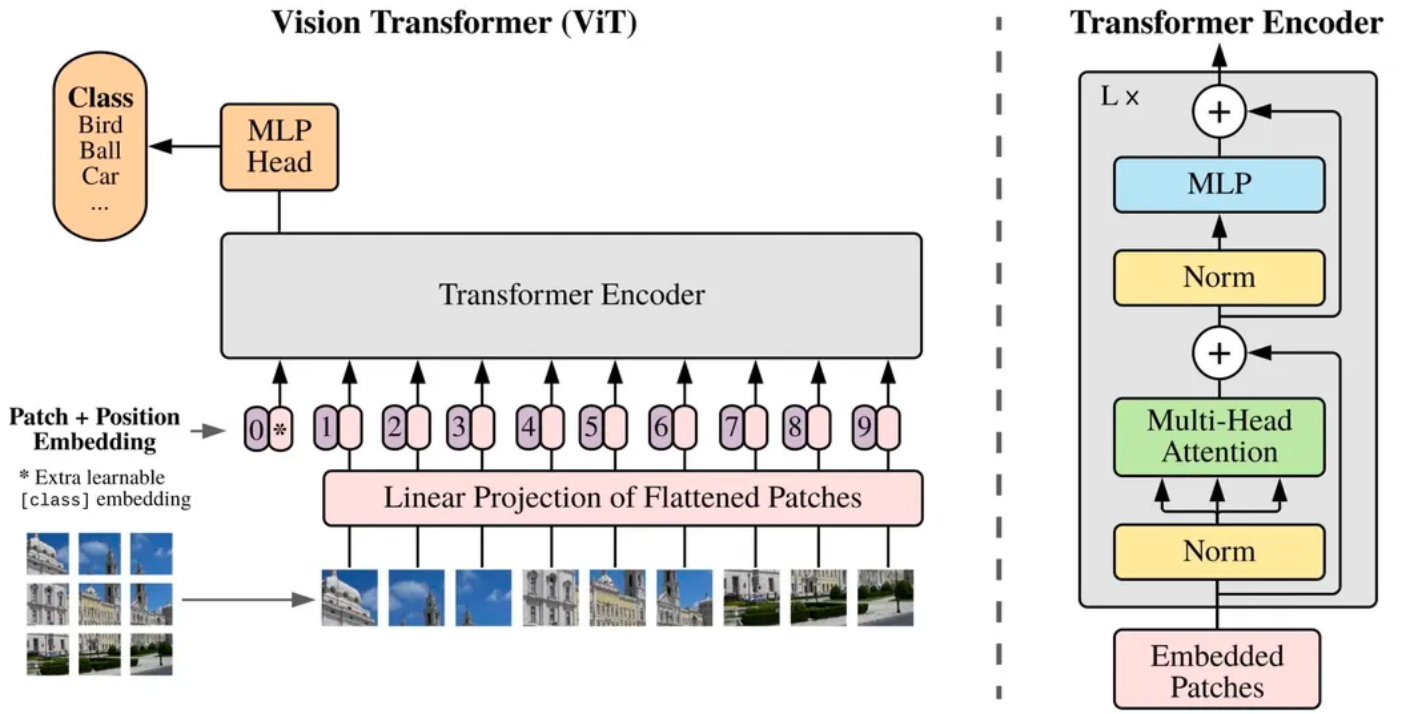


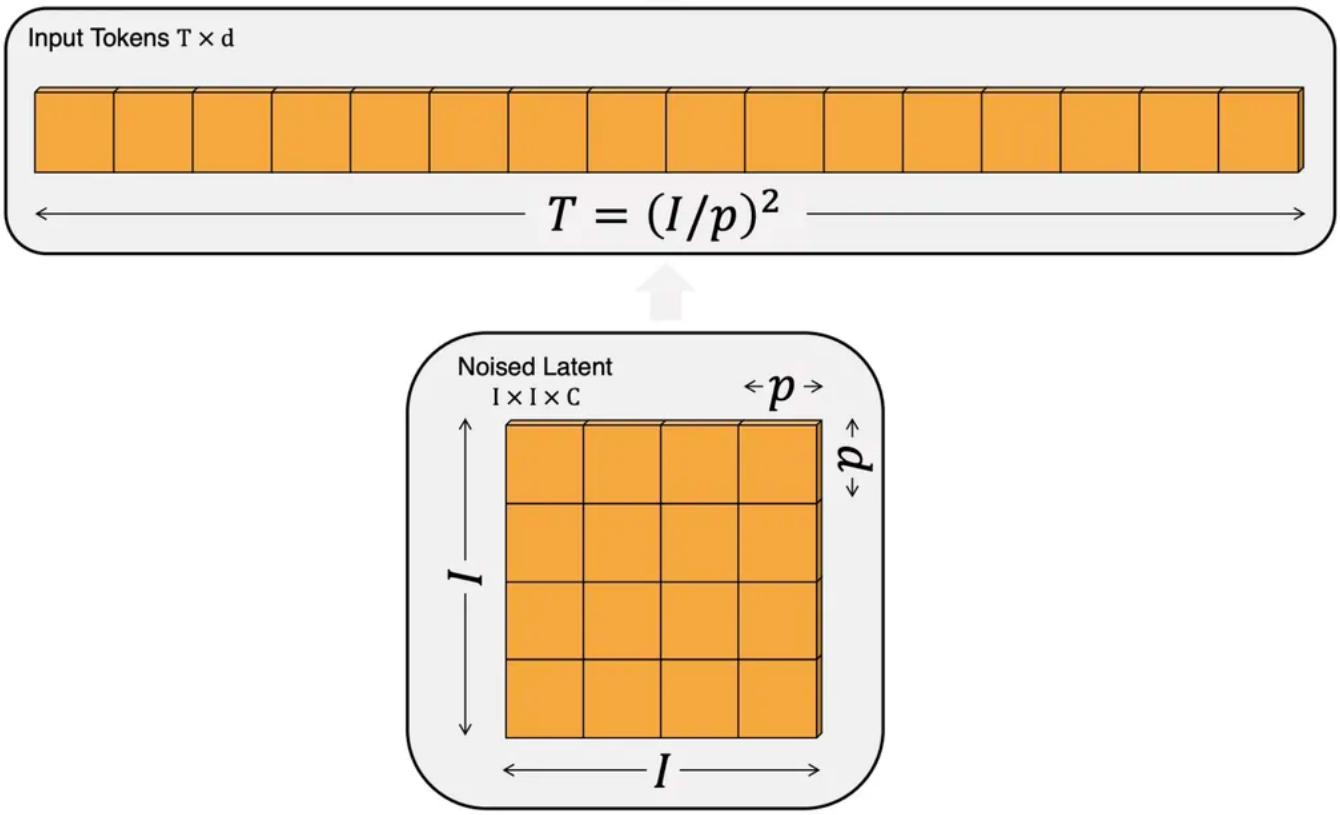
Figure 3. **The Diffusion Transformer (DiT) architecture.** *Left:* We train conditional latent DiT models. The input latent is decomposed into patches and processed by several DiT blocks. *Right:* Details of our DiT blocks. We experiment with variants of standard transformer blocks that incorporate conditioning via adaptive layer norm, cross-attention and extra input tokens. Adaptive layer norm works best.

Vision Transformer 基础（选读）

既然要用Transformer来替换U-net，我们需要了解一下Transformer如何处理图片数据的呢？经典之作当然是Vision Transformer (ViT)了，它的主要思想就是**将图片分割为固定大小的图像块（image patch/token），对于每个图像块进行线性变换并添加位置信息，然后将得到的向量序列送入一个标准的Transformer编码器。**



至于图片的切割分块可以参考DiT中的这张图，假如patch的大小为$p \times p$，那么一张$I \times I \times C$的图片会切成一个$T = (I/p)^2$的图片块序列，然后当作文本序列来看待就好了。



Sora：视频生成的新纪元

先抛出我的观点：**Sora就是改进的DiT**。

而DiT本质上是 VAE编码器 + ViT + DDPM + VAE解码器；从OpenAI的技术报告体现出来的创新点我认为主要有两个方面：

- 改进VAE -> 时空编码器
- 改进DiT -> 不限制分辨率和时长

至于图像分块、Scaling transformers、视频re-captioning、视频编辑（SDEdit）这些其实都是已知的一些做法了。

实现一个最low的视频生成模型

假如我们具备前面的知识了，或者说给你一个DiT（能够进行图片生成），如何能够实现视频的生成呢？

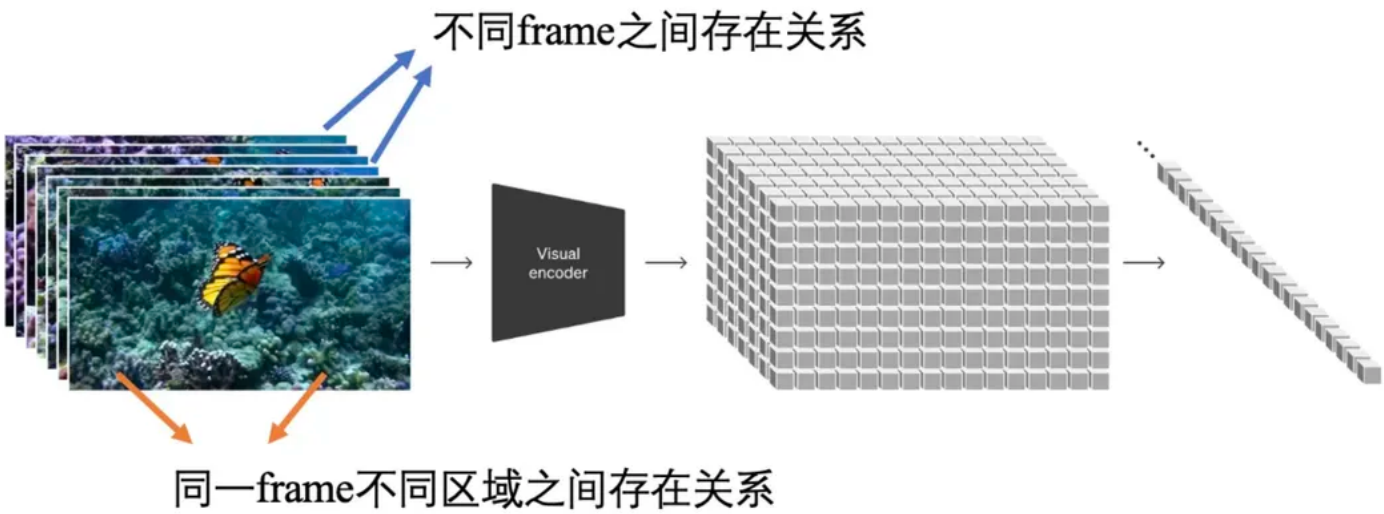
> 视频的每一帧（frame）本质上就是一张图片。在视频播放时，这些连续的图片以一定的速率（帧率，通常以每秒帧数FPS表示）快速播放，由于人眼的视觉暂留效应，这些连续的静态图片在观众眼中形成了动态效果，从而产生了视频的流畅运动感。

那要实现视频生成是不是可以**看作是多帧图片的生成，因此最low的做法就是把视频生成看作独立的图片生成，使用DiT生成多帧图片然后串起来就是视频了。**

当然，这样做的问题显然很大，因为没有考虑视频不同帧图片之间的关联，可能会导致生成的多帧图像很不连贯，串起来看就不像是视频了。

改进VAE：融入时间关联

为了使得视频的生成连贯，那在VAE编解码的过程自然需要去考虑视频不同帧的关系，原来对图片进行处理相当于考虑的是图片空间上的关系，现在换到视频就是多了时间上的关系，也就是经典的时空联合建模问题。



> 时空联合建模的方法其实非常多了，比如使用3D CNN、比如时间空间单独处理再融合、比如设计fancy的注意力机制等等。

对应的就是Sora技术报告中的Video compression network

> We train a network that reduces the dimensionality of visual data.20 This network takes raw video as input and outputs a latent representation that is compressed both temporally and spatially.

技术报告中没有提及具体的做法，但是可以看出这里显然是在VAE编码器上考虑了时空建模，对于解码器的设计没有相关介绍（可能也考虑了时空建模，也可能不做改动）。

这里再分享另外一个工作VideoLDM，他们的做法是在解码器上插入额外的temporal layers来考虑视频帧之间的关系，而编码器是保持不变的。

> 这里额外推荐几篇视频生成的相关论文，Make-A-Video、Emu Video、VideoLDM、AnimateDiff、VideoPoet、Lumiere，感兴趣的同学可以进一步了解，这里不再赘述。

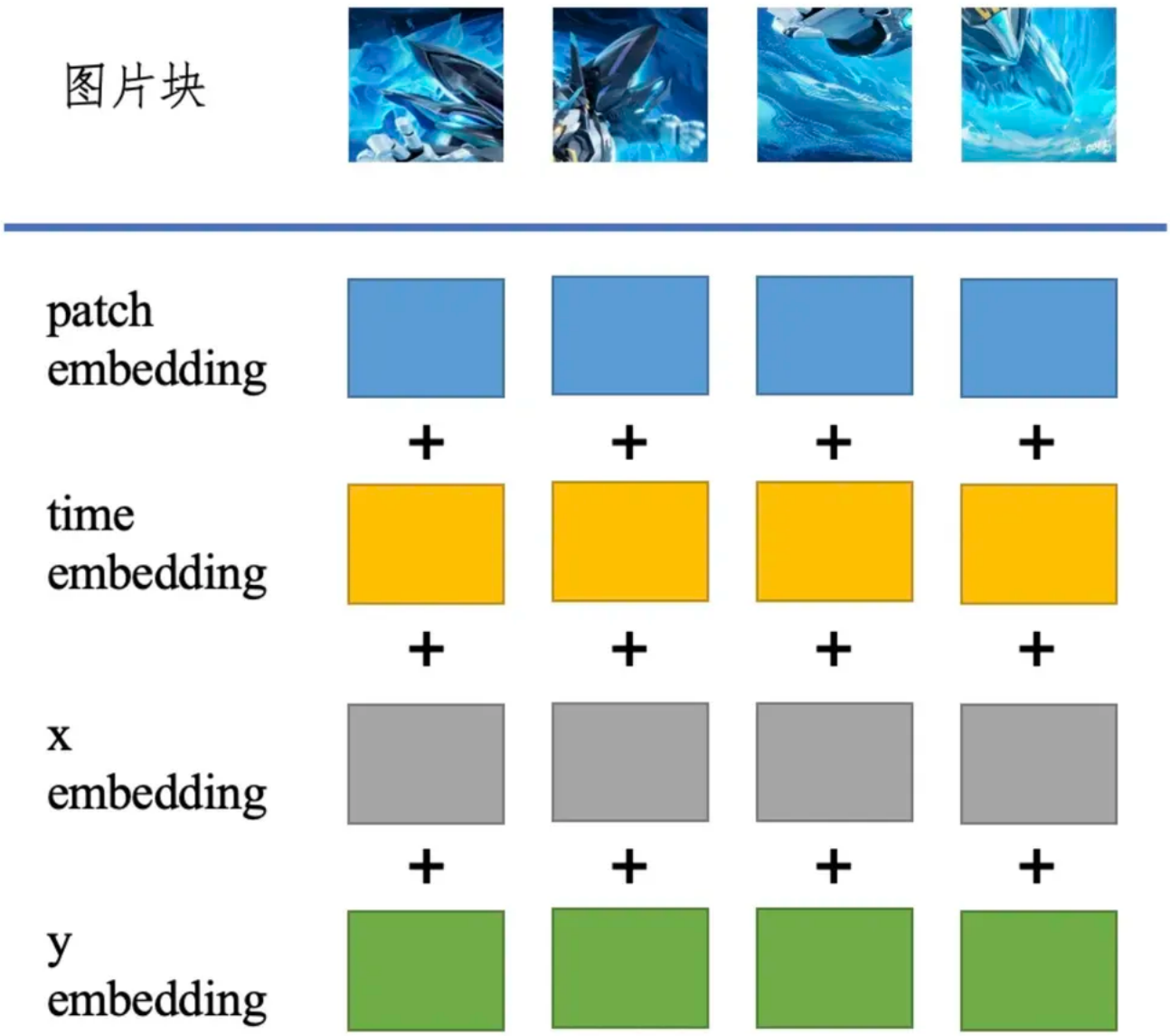
改进DiT：适配任意分辨率和时长

网上的很多分享都在传Sora能适配任意分辨率和时长是参考了NaViT这篇文章的做法，其实并非如此，Vision Transformer (ViT)本身就能够处理任意分辨率（不同分辨率就是相当于不同长度的图片块序列，不就类似你给大语言模型提供不同长度的输入一个意思）！NaViT只是提供了一种高效训练的方法。

接下来我们来思考下DiT如何处理不同分辨率和时长的视频数据呢？假如我们有一个$T \times X \times Y \times C$的视频会切成一个$T * (X/p_x) * (Y/p_y)$的图片块序列（当然这里图片块的排序也可以有一些讲究，比如按空间维度排序，或者按时间维度排序）。

因为T 、 X 、 Y都是可变的，所以关键的问题应该是改进DiT使它能够更好地识别不同图片块是属于原始视频中的哪个区域。

一种可行的做法就是从**位置编码**的角度入手，比如对于输入Transformer的图片块，我们可以在**patch embedding上叠加上时间、空间上的位置信息。**



> 类似的，其实在ViT的工作中考虑过使用2D-aware position embeddings来表示不同图片块的位置信息，只是没有什么明显的效果。

数据仍然还是王道

在数据层面上，OpenAI也给出了两点很有参考价值的方法

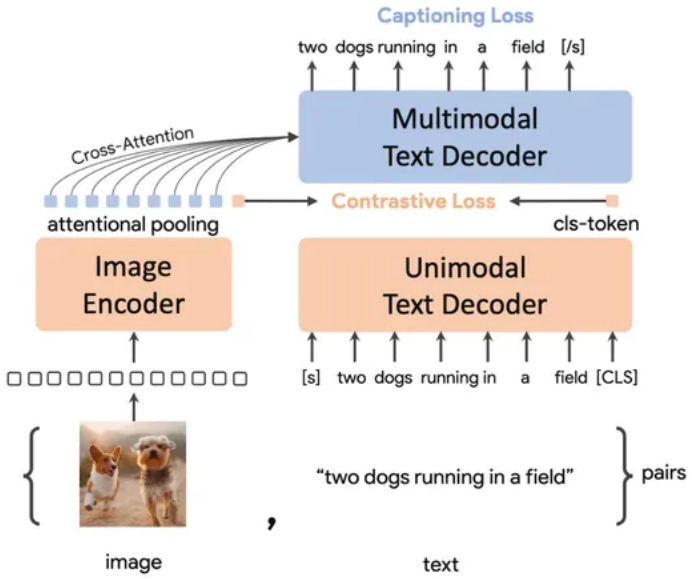
- 不对视频/图片进行裁剪等预处理，**使用native size**
- 数据的**质量很重要**，比如（文本、视频）这样的成对数据，原始的文本可能并不能很好的描述视频，可以通过re-captioning的方式来优化文本描述，这一点在DALL·E 3的报告中也已经强调了，这里只是进一步从图片re-captioning扩展到视频re-captioning。

那么DALL·E 3是如何做re-captioning的呢？（选读）

既然是生成文本，自然就是类GPT的语言模型，做的就是下一个词预测；考虑到生成的文本是图像的描述，因此需要基于图像这个条件，也就是条件生成，要把 image 放进去

$$L(t) = \sum_j \log P(t_j | t_{j-k}, \ldots, t_{j-1}; image; \Theta)$$

那么图像以什么形式放进去呢？简单点，就用 CLIP 来编码图像，然后把图像embedding放进去就ok了。那这样一个图像打标模型怎么训练呢？用的就是 CoCa 的方法，也就是同时考虑对比损失和LM损失



Algorithm 1 Pseudocode of Contrastive Captioners architecture.

```
# image, text.ids, text.labels, text.mask: paired {image, text} data
# con_query: 1 query token for contrastive embedding
# cap_query: N query tokens for captioning embedding
# cls_token_id: a special cls_token_id in vocabulary

def attentional_pooling(features, query):
  out = multihead_attention(features, query)
  return layer_norm(out)

img_feature = vit_encoder(image) # [batch, seq_len, dim]
con_feature = attentional_pooling(img_feature, con_query) # [batch, 1, dim]
cap_feature = attentional_pooling(img_feature, cap_query) # [batch, N, dim]

ids = concat(text.ids, cls_token_id)
mask = concat(text.mask, zeros_like(cls_token_id)) # unpad cls_token_id
txt_embs = embedding_lookup(ids)
unimodal_out = lm_transformers(txt_embs, mask, cross_attn=None)
multimodal_out = lm_transformers(
    unimodal_out[:, :-1, :], mask, cross_attn=cap_feature)
cls_token_feature = layer_norm(unimodal_out)[:, -1:, :] # [batch, 1, dim]

con_loss = contrastive_loss(con_feature, cls_token_feature)
cap_loss = softmax_cross_entropy_loss(
    multimodal_out, labels=text.labels, mask=text.mask)
```

vit_encoder: vision transformer based encoder; lm_transformer: language-model transformers.

Figure 2: Detailed illustration of CoCa architecture and training objectives.

模型推理策略

官方展示Sora的应用有很多，比如文生视频、图生视频、视频反推、视频编辑、视频融合等。 这里就会涉及一些有意思的做法，比如可以这么做（以下做法并不唯一）

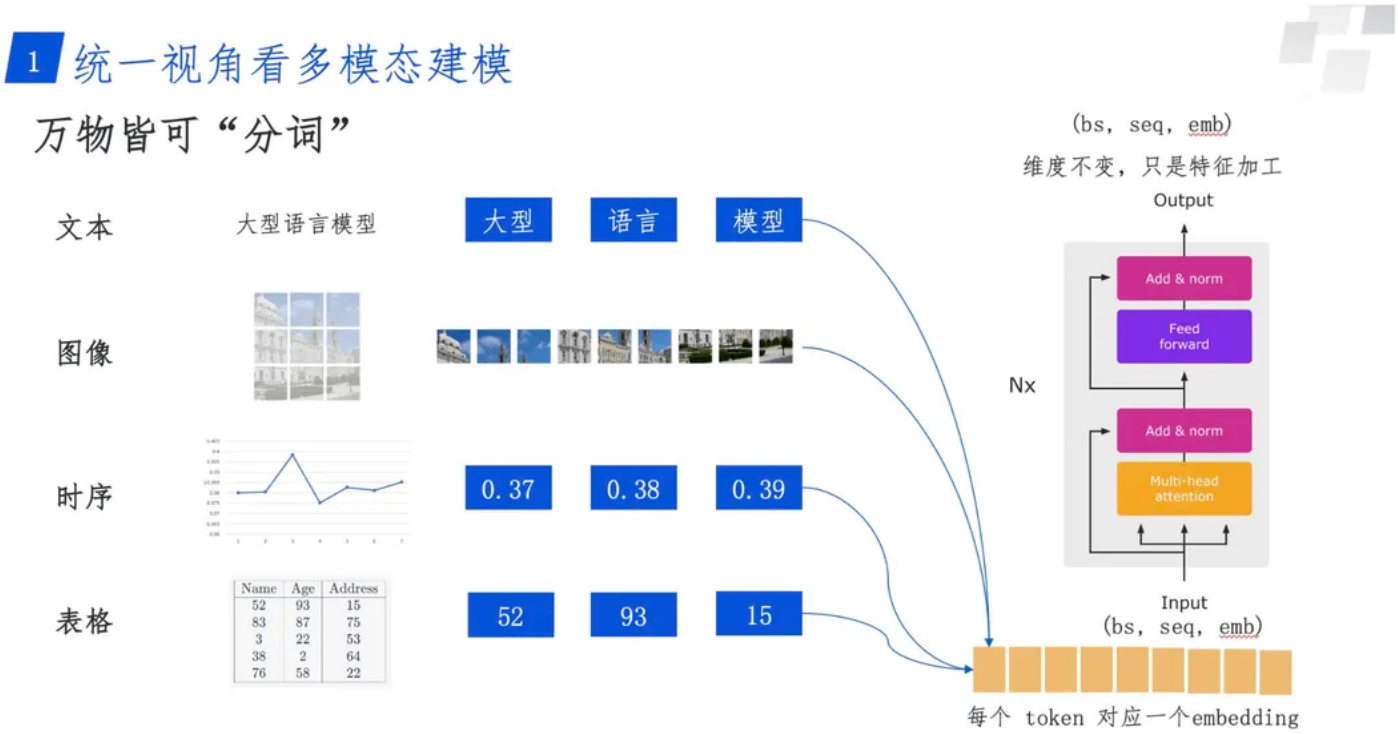
1.文生视频：喂入DiT的就是文本embedding+全噪声patch

2.视频编辑：类似SDEdit的做法，在视频上加点噪声（不要搞成全是噪声），然后拿去逐步去噪

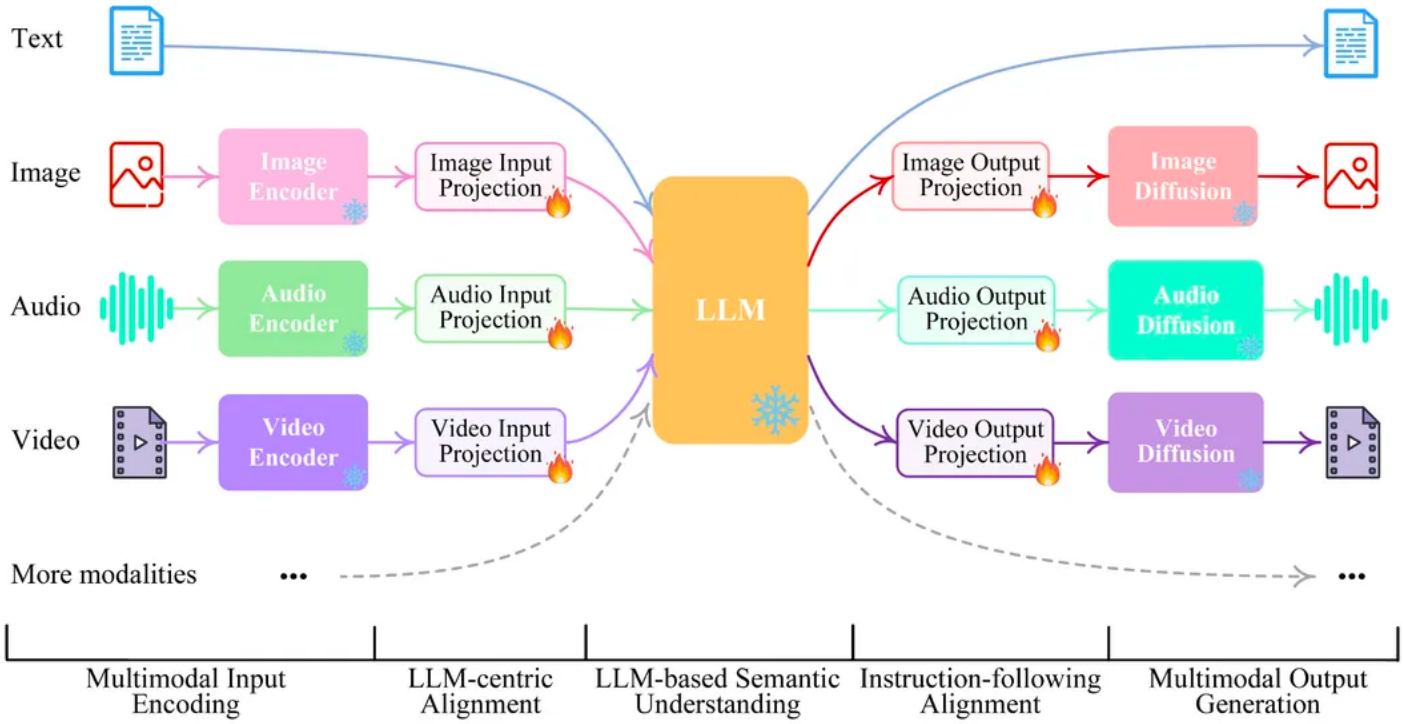
3.图生视频、视频反推、视频融合：喂入DiT的就是文本embedding（可选）+特定帧用给定图片的embedding+其他帧用全噪声patch

未来会是什么？

不知道大家有没有注意到，Sora还提到了它除了文生视频，也支持文生图，这里其实透露出了一种统一的味道。我相信未来的发展肯定会出现更加强大的多模态统一，这里引用一张去年我在一场内部分享中的slides来结束本次的文章，**万物皆可“分词”，选择合适的编码器 + Transformer结构或其他 + 合适的解码器，就可能实现各种不同模态之前的互相转换、互相生成！**



比如可以参考NExT-GPT的一个结构图



相信2024年同样会是AI领域突飞猛进的一年，期待更多精彩的工作。

【参考资料】

Autoencoders, https://arxiv.org/abs/2003.05991Auto-Encoding Variational Bayes, https://arxiv.org/abs/1312.6114Denoising Diffusion Probabilistic Models, https://arxiv.org/abs/2006.11239High-Resolution Image Synthesis with Latent Diffusion Models, https://arxiv.org/pdf/2112.10752.pdfScalable Diffusion Models with Transformers, https://arxiv.org/pdf/2212.09748.pdfAn Image is Worth 16x16 Words: Transformers for Image

Recognition at Scale, https://arxiv.org/abs/2010.11929Sora, https://openai.com/research/video-generation-models-as-world-simulatorsVideo LDM, https://arxiv.org/pdf/2304.08818.pdfNaViT, https://arxiv.org/abs/2307.06304CoCa, https://arxiv.org/pdf/2205.01917.pdfNExT-GPT, https://next-gpt.github.io/

相关推荐文章：**一文带你了解OpenAI Sora**