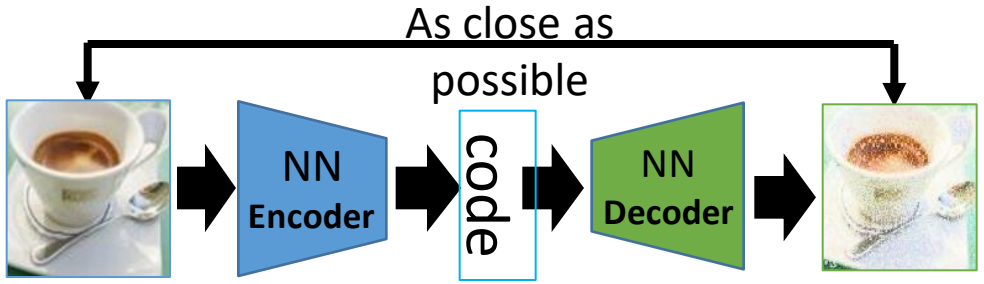
《计算机模拟》

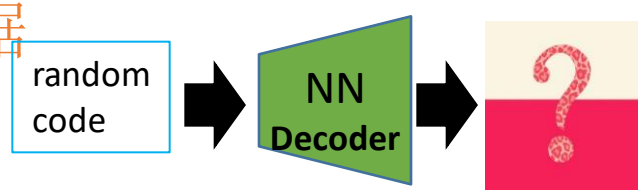
第15讲 – 扩散生成模型

胡贤良

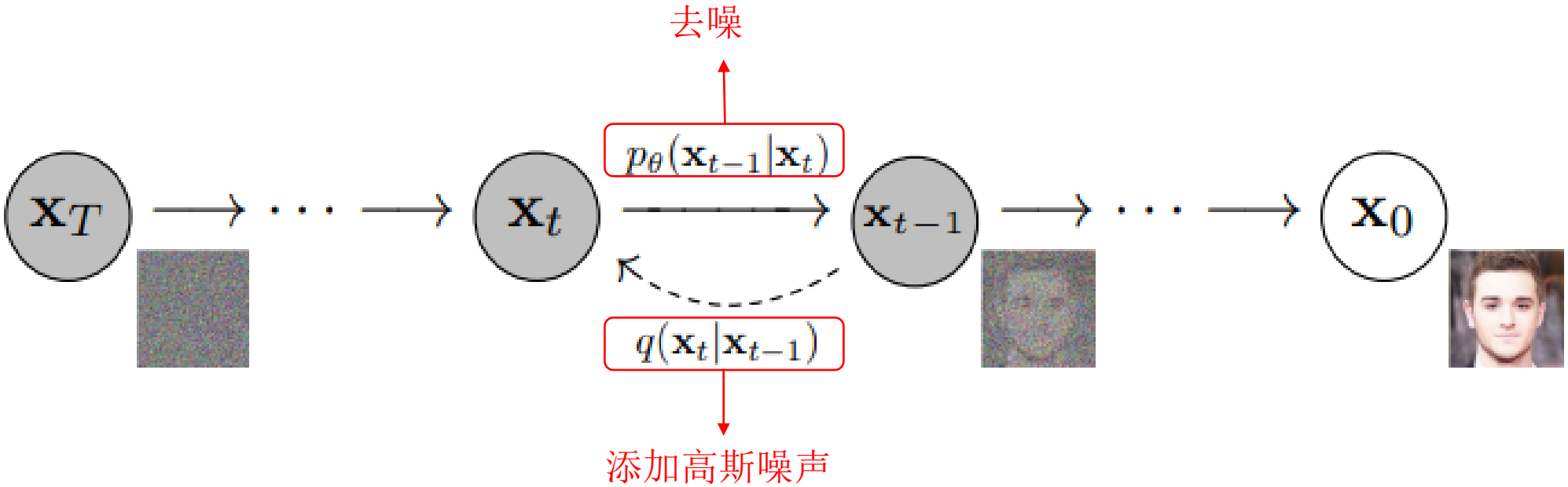
浙江大学数学科学学院

扩散模型



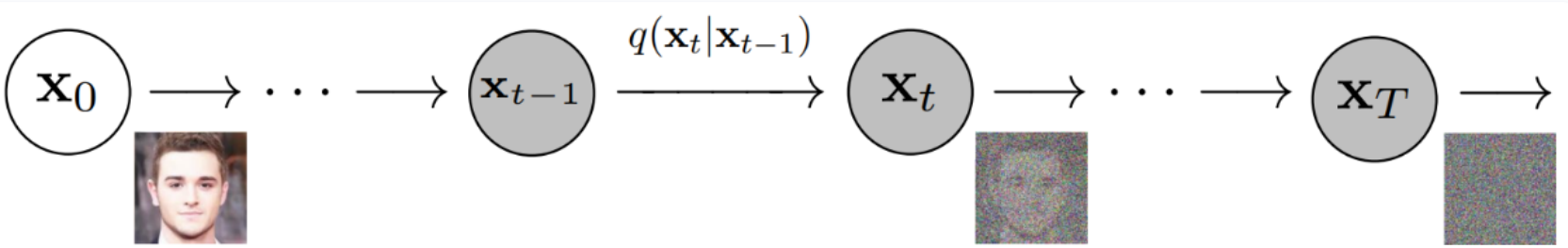


- 扩散模型是一类生成模型，用于生成与训练数据相似的数据
- 工作原理：
  1. 使用缓慢增加的噪声顺序破坏训练数据 （正向过程）
  2. 学习扭转这种破坏以形成数据的生成模型 （反向过程）



扩散模型 – 加噪(前向)过程

- 扩散模型可以从几个不同角度理解：基于 Markov 链、基于分数匹配、基于微分方程等。常用如下示意图表示：



前向传播过程： $\mathrm{x}_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_1$ ① 其中$\alpha_t = 1-\beta_t$， β_t $0.0001 \to 0.002$

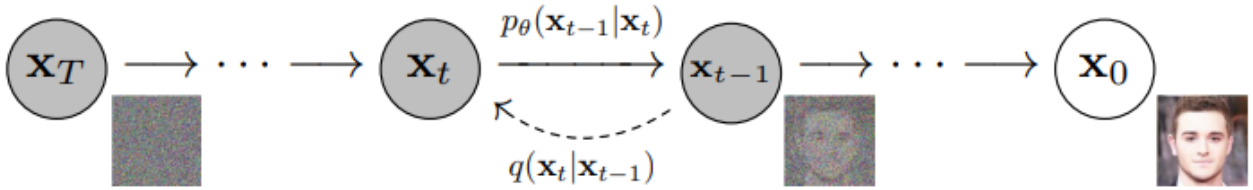
$$\mathrm{x}_{t-1} = \sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_2$$

②

② 代入 ① 即有 $\mathrm{x}_t = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_2$

$= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}_t$ ⟶ 有初始的x_0就可以得到任意x_t

扩散模型 – 去噪(反向)过程



贝叶斯公式：

$$q(x_{t-1}|x_t, x_0) = q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

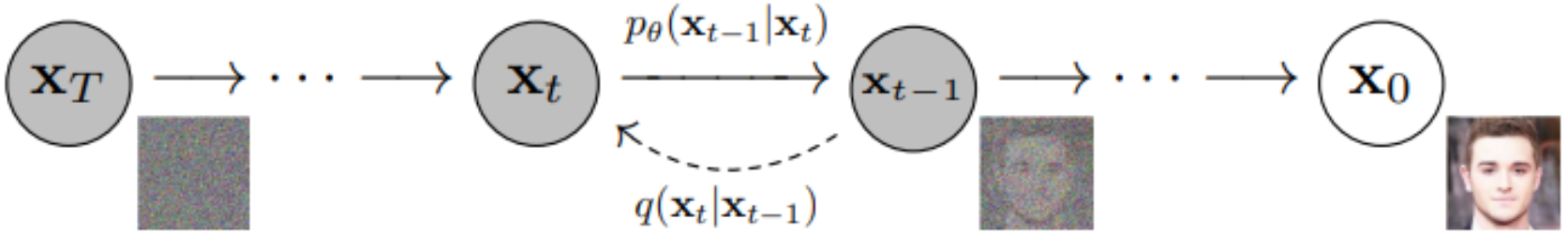
$$q(x_t|x_{t-1}, x_0) = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon$$

$$q(x_{t-1}|x_0) = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon$$

$$q(x_t|x_0) = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

$$q(x_{t-1}|x_t, x_0) \propto \exp\left(-\frac{1}{2}\left(\frac{\left(x_t - \sqrt{\alpha_t}x_{t-1}\right)^2}{\beta_t} + \frac{\left(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0\right)^2}{1-\bar{\alpha}_{t-1}} + \frac{\left(x_t - \sqrt{\bar{\alpha}_t}x_0\right)^2}{1-\bar{\alpha}_t}\right)\right)$$

扩散模型 – 去噪过程(续)



$$q(x_{t-1}|x_t, x_0) \propto \exp\left(-\frac{1}{2}\left(\frac{\left(x_t - \sqrt{\alpha_t}x_{t-1}\right)^2}{\beta_t} + \frac{\left(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0\right)^2}{1-\bar{\alpha}_{t-1}} + \frac{\left(x_t - \sqrt{\bar{\alpha}_t}x_0\right)^2}{1-\bar{\alpha}_t}\right)\right)$$

$$= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)x_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right)x_{t-1} + C(x_t, x_0)\right)\right)$$

而 $\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2}x^2 - \frac{2\mu}{\sigma^2}x + \frac{\mu^2}{\sigma^2}\right)\right)$

$$\mu_t(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$$

$$= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{1-\bar{\alpha}_t}\epsilon_t\right)$$

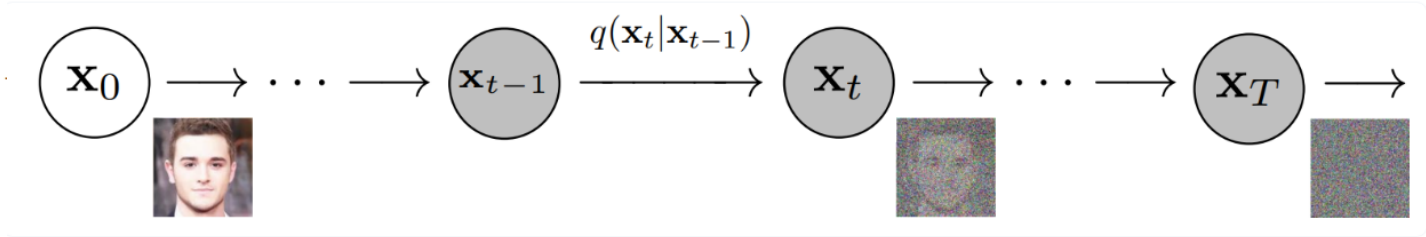
$$x_0 = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_t\right)$$

扩散模型 – 训练&采样过程

展示采样代码：demo_ddpm.html

Algorithm 1 Training

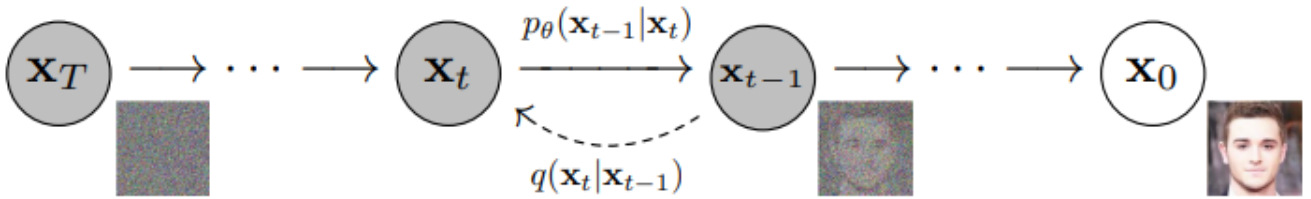
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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



$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}_t$$

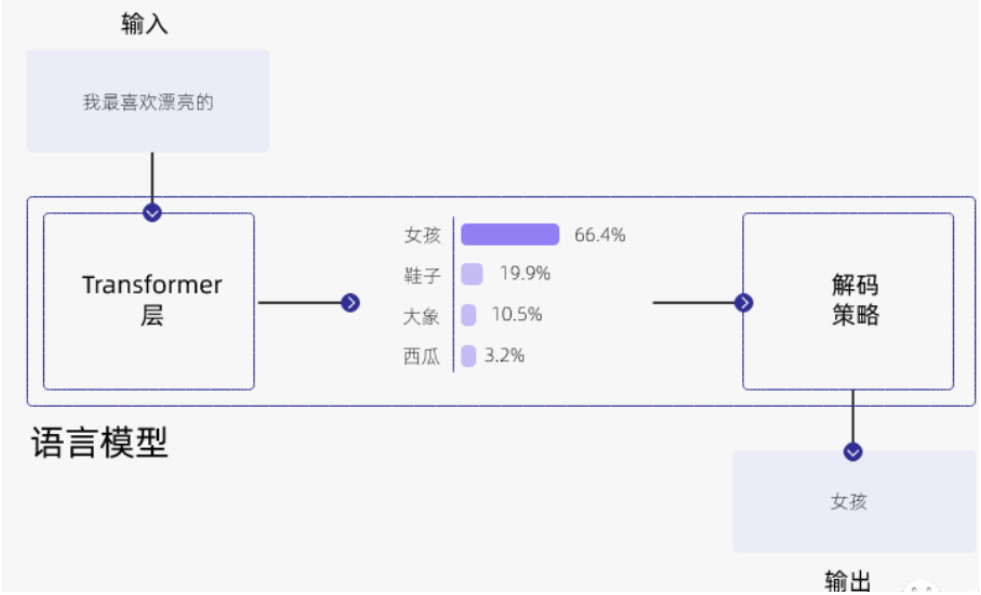
Algorithm 2 Sampling

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



$$\mu_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{1-\bar{\alpha}_t}\epsilon_t)$$

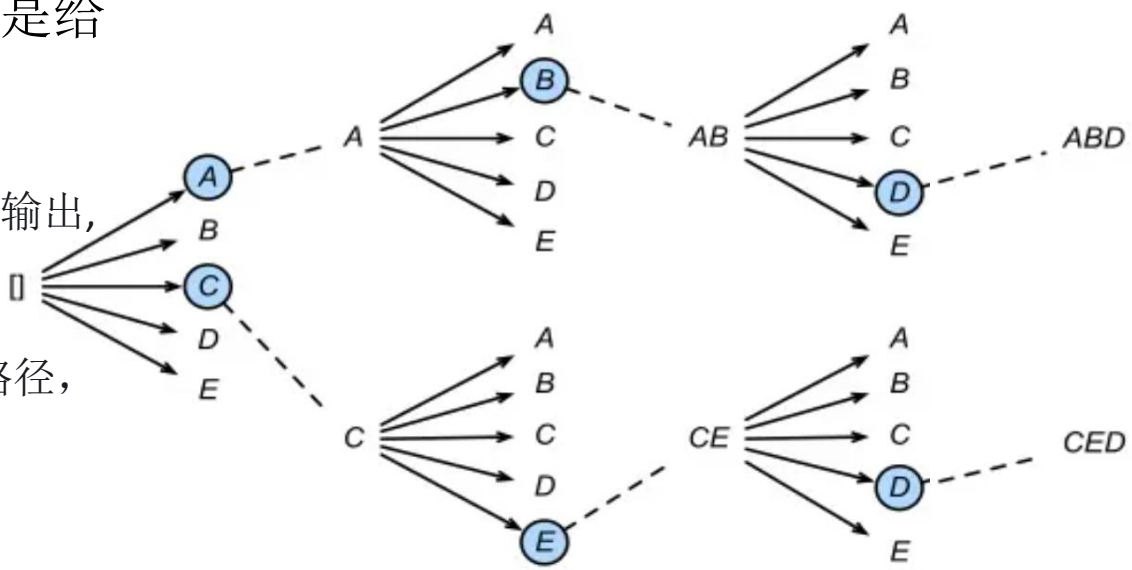
知识点：生成模型的采样策略



- 生成模型（包括GPT）均基于假设(定义$\omega_{1:0} = \boldsymbol{\phi}$)：

$$P(\omega_{1:T}|\omega_0) = \prod_{t=1}^{T} P(\omega_t|\omega_{1:t-1}, \omega_0),$$

- 以”预测下一个词” 为例，以上述目标为最优的模型总是给出语言模型中概率最大的词：
  - 贪婪搜索(**Greedy Search**):每一步都选择概率最大的 token 进行输出，计算速度快, 但只是局部最优，容易“一步错，步步错”
  - 束搜索(**Beam Search**):在每个时间步保留 k个最大的可能取值路径，k=1时，退化为贪婪搜索



- 实际应用中人们往往需要一些“出人意料”的结果。

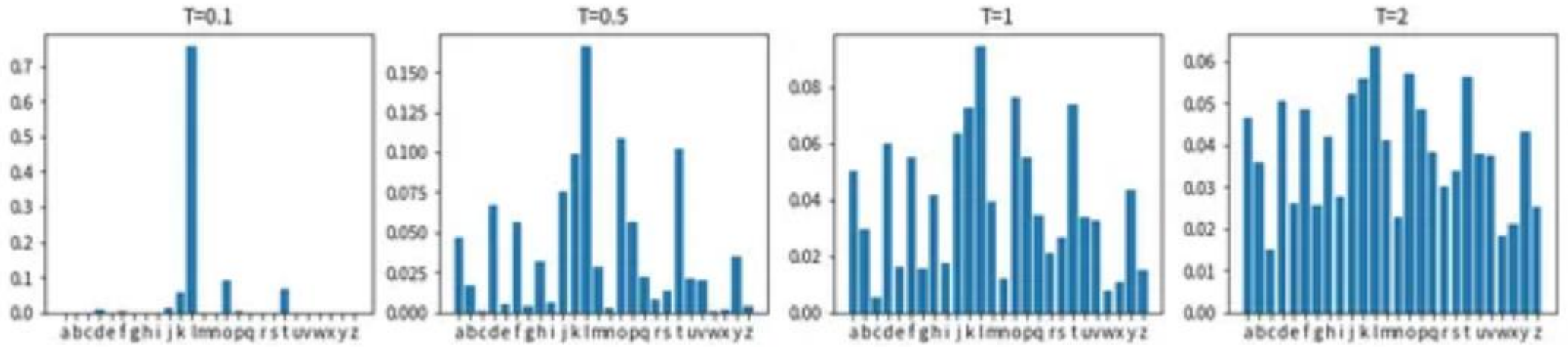
Temperature Sampling

根据概率分布情况，来随机生成下一个单词：

- 难点：如果按照全体词的概率分布来进行采样，还是有可能生成低概率的单词。
- 解决策略：在 softmax 函数中加入 Temperature 参数，强化顶部词的生成概率

$$p(i) = \frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^{K} e^{\frac{z_k}{T}}}$$

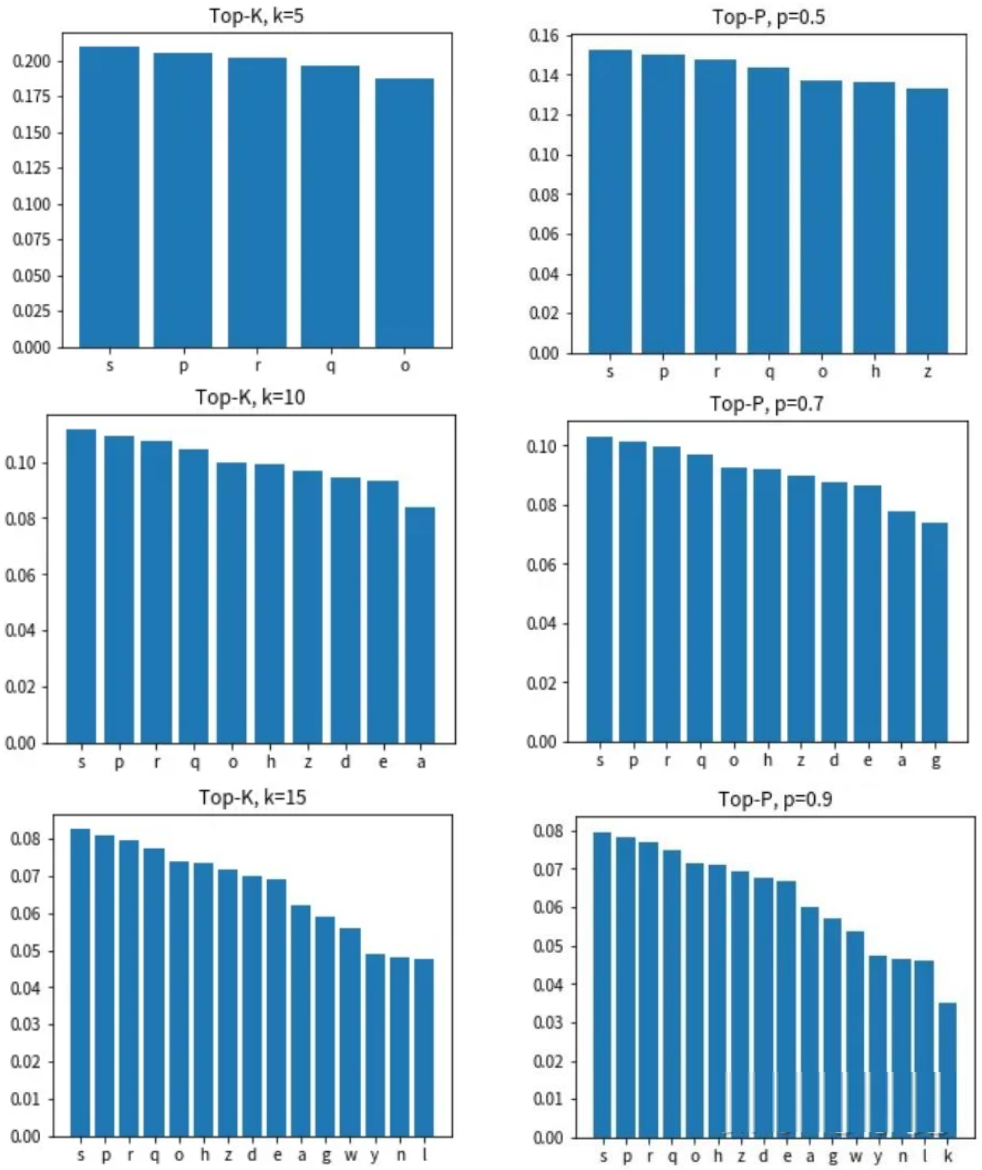
这里，当T<1时可以增加顶部词汇的生成概率，T越大越倾向于均匀分布



实用策略

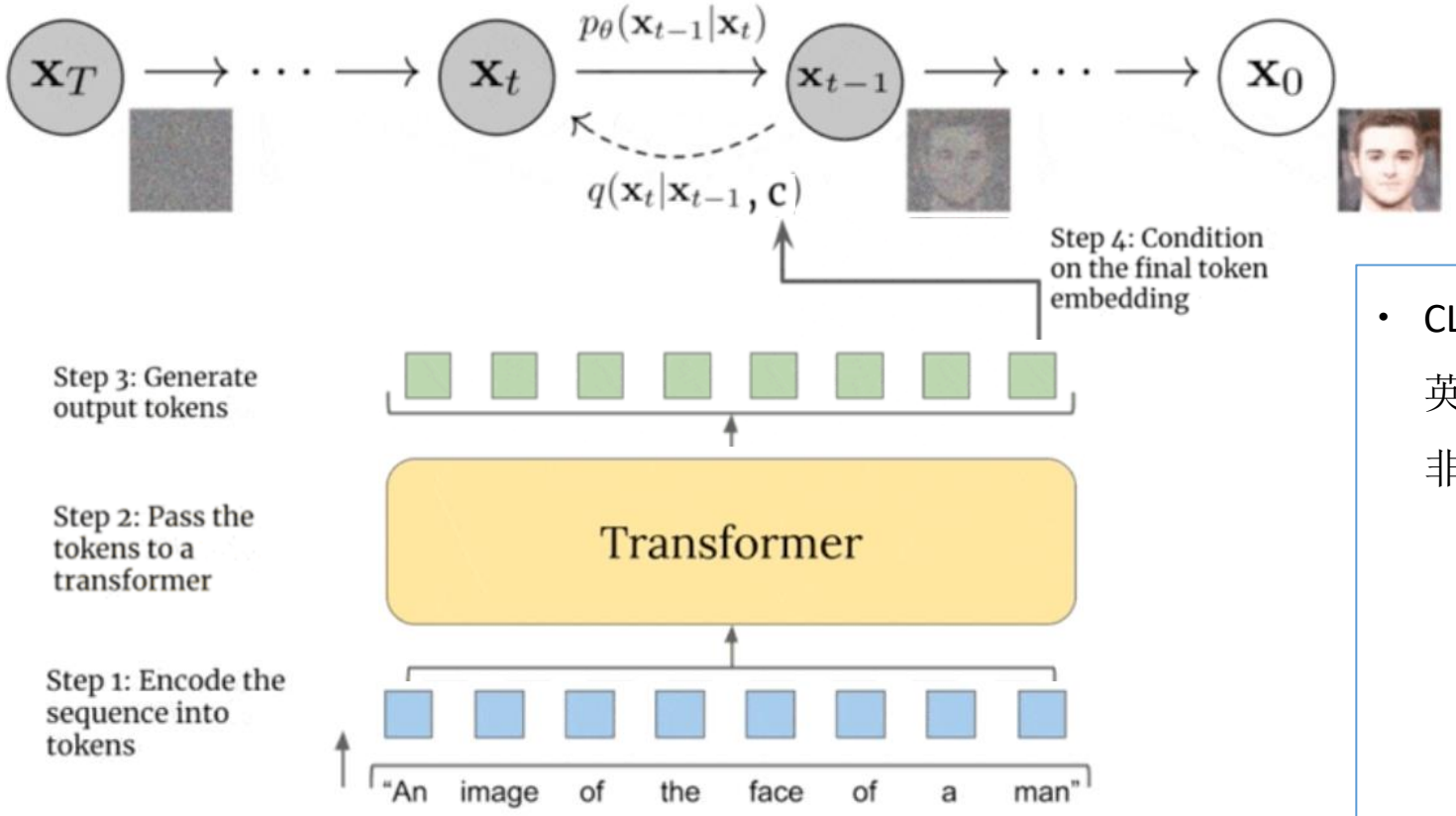
在 Temperature Sampling 的过程中，即便选取合适的 T，还有可能生成低概率的单词。实用中还需要结合一些策略：

- Top-K Sampling：直接挑选概率最高的 K 个候选者(可以避免低概率词的生成)，重新根据 softmax 计算这 K 个单词的概率后，再根据概率分布情况进行采样，生成输出。缺点是K值的选择需要依赖于经验或调参
- Top-P Sampling (Nucleus sampling)：预设一个概率界限 p 值(同样依赖于经验或调参)，然后将所有候选者根据概率大小从高到低排列，依次选取。当候选者的累积概率到达 p 值时停止，从已经选取的单词中进行采样生成输出。



典型应用：文生图

GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models

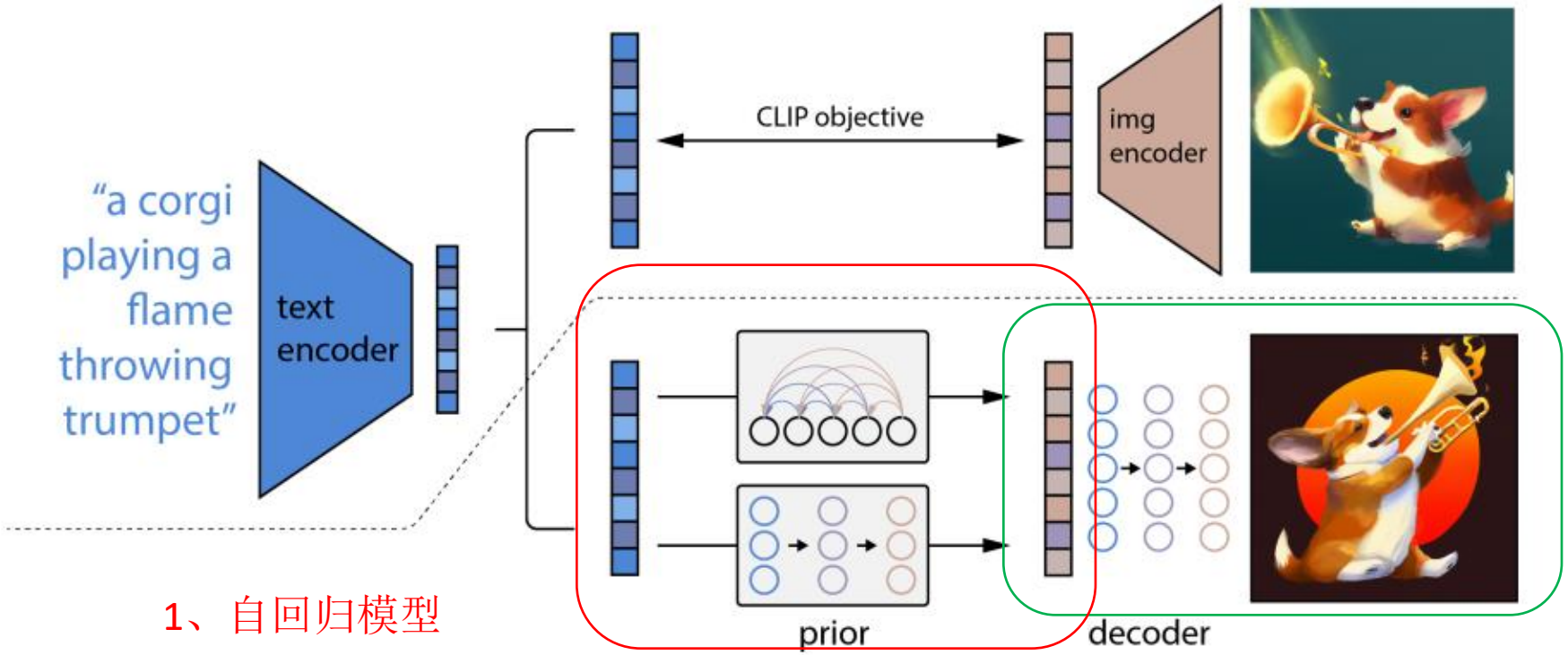


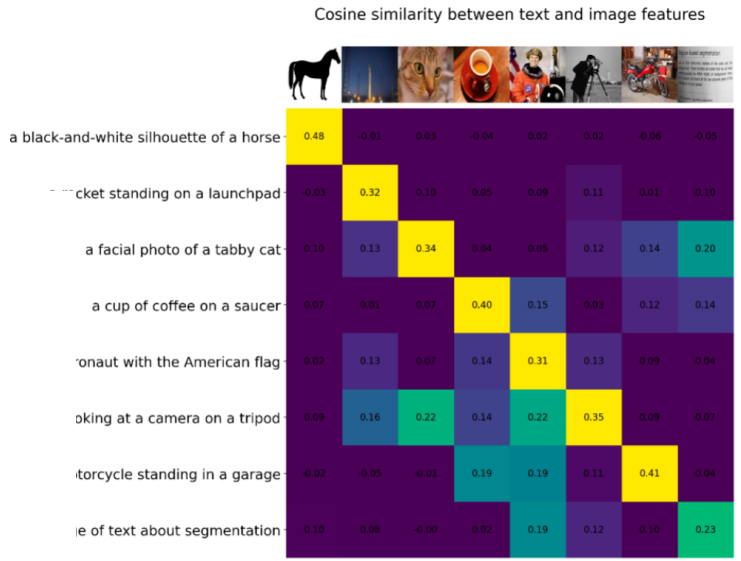
$$\hat{\mu}_\theta(x_t|y) = \mu_\theta(x_t|y) + s \cdot \Sigma_\theta(x_t|y)\nabla_{x_t} \log p_\phi(y|x_t)$$

- CLIP：把文本和图像映到同一个空间中。如：英文单词dog 和狗的图片在CLIP 下的像是两个非常接近的向量。
  - 一个较为新颖的方法，有一定的技术推进性。
  - 开源代码参考，代码逻辑清晰可读性高
  - 训练数据大小为 84 G，近 300 万条训练数据
  - 6 节点 48 张 A100 大约需要 3 小时
  - 在多模态领域具有启发意义
- 开源项目 - OpenCLIP

典型应用：文生图

DALLE2: Hierarchical Text-Conditional Image Generation with CLIP Latents





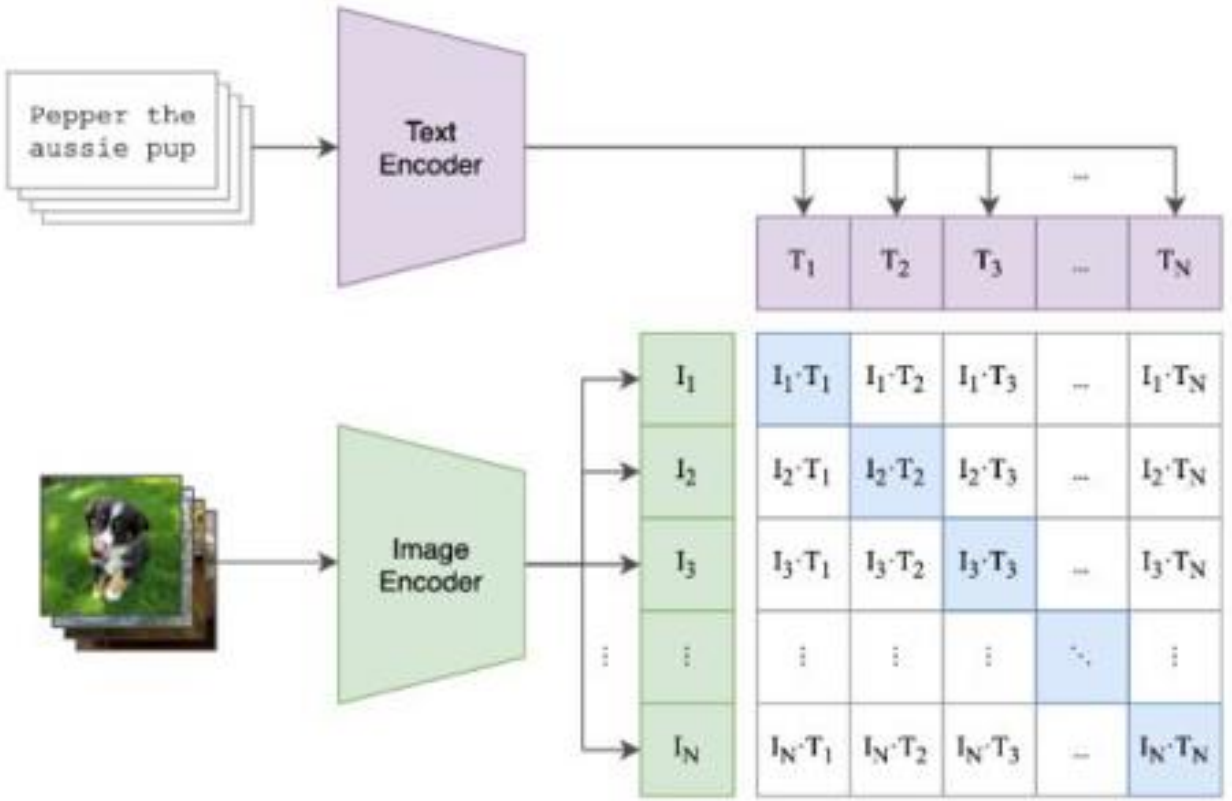
1、自回归模型

2、Diffusion model

$$\hat{\mu}_\theta(x_t|c) = \mu_\theta(x_t|c) + s \cdot \Sigma_\theta(x_t|c)\nabla_{x_t}\left(f(x_t) \cdot g(c)\right)$$

a teddy bear on a skateboard in times square

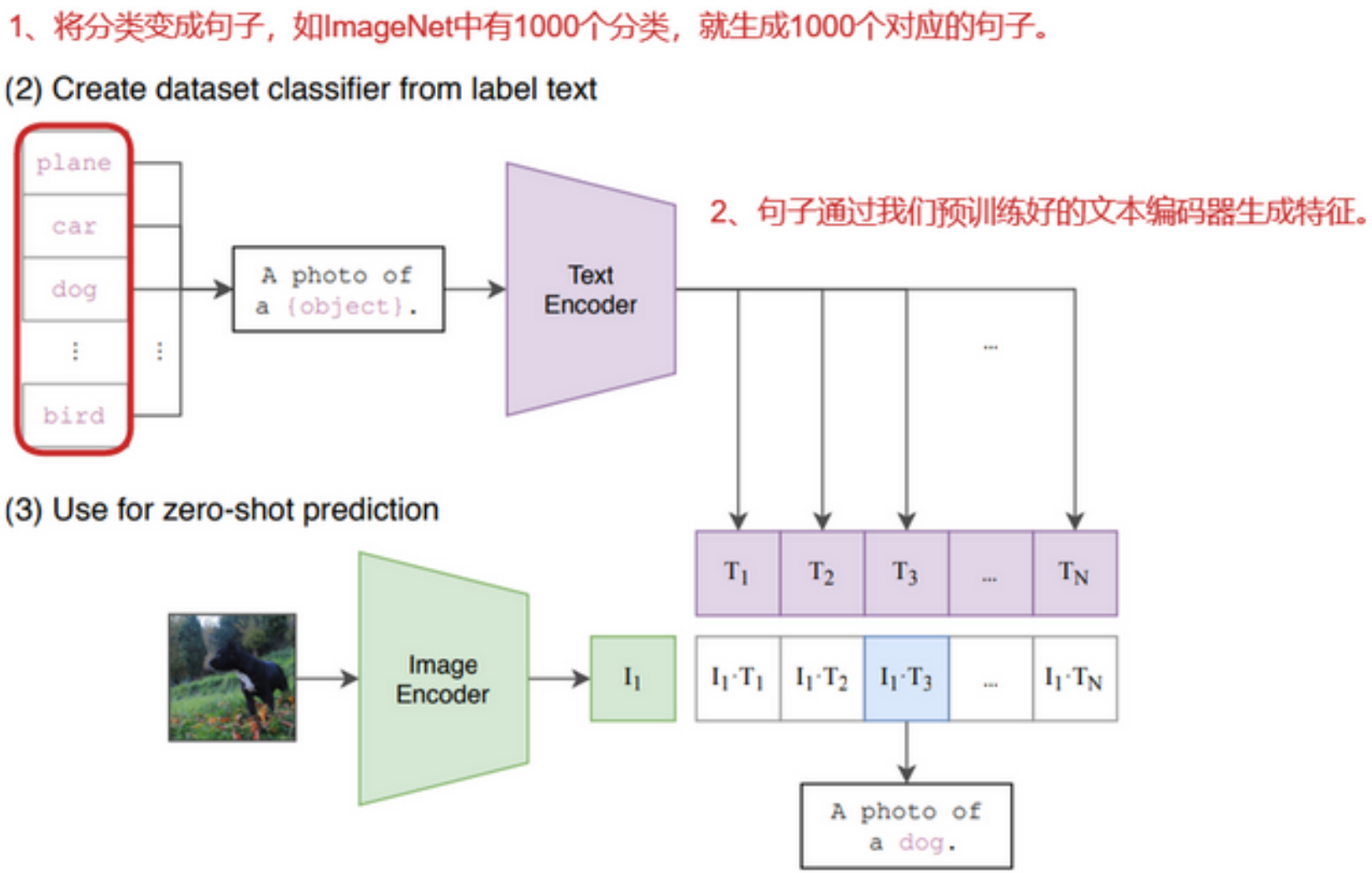
CLIP训练：Contrastive pre-training



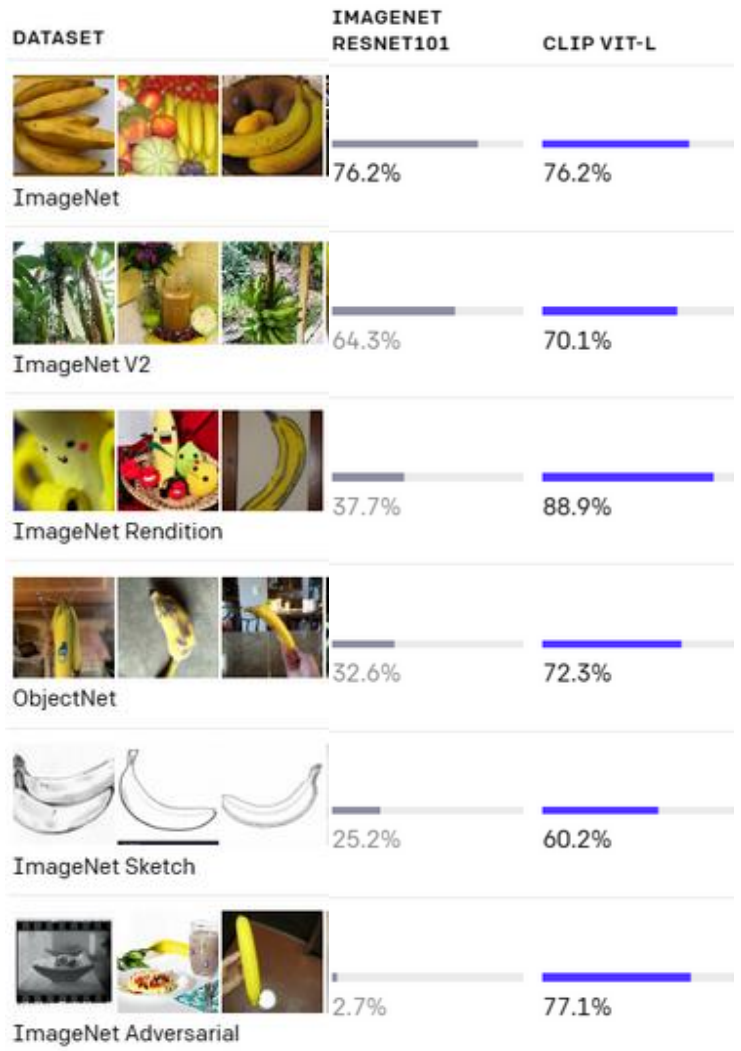
1. 输入图片 => 图像编码器 => 图片特征向量
2. 输入文字 => 文本编码器 => 文本特征向量
3. 对两个特征进行线性投射，得到相同维度的特征，并进行L2归一化
4. 计算两个特征向量的相似度（夹角余弦）
5. 对n个类别进行softmax，确定个正样本和个负样本，并最大化正样本的权重。

CLIP的使用(下游任务？)：

1、将分类变成句子，如ImageNet中有1000个分类，就生成1000个对应的句子。



3、将图像特征与每个文本特征计算相似性，将最相似的选出来作为结果。



可控AI绘画

- 基于扩散模型的生成方式是流行趋势
- 随机性优势导致生成结果有丰富的想象空间
- 镜子的两面：不可控性，对工程需求不友好

1. 分类器引导的采样

> Xihui Liu, et al.
> More control for free! image synthesis with semantic diffusion guidance.
> *CoRR*, abs/2112.05744, 2021.

扩散模型的关键步骤就是对 $\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t$ 的采样，当我们期望控制生成的结果时，我们的问题就变成了对 $\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y$ 的采样。
首先根据贝叶斯公式可以写出

$$p(\boldsymbol{x}_{t-1} \mid y) = \frac{p(\boldsymbol{x}_{t-1})p(y \mid \boldsymbol{x}_{t-1})}{p(y)}.$$

为了用上已知的 $\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t$ 的分布，在这个式子里的每一项里添加上条件 $\boldsymbol{x}_t$，得到

$$p(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y) = \frac{p(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)p(y \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_t)}{p(y \mid \boldsymbol{x}_t)} = \frac{p(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)p(y \mid \boldsymbol{x}_{t-1})}{p(y \mid \boldsymbol{x}_t)},$$

其中用到了 $p(y \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_t) = p(y \mid \boldsymbol{x}_{t-1})$，这是因为 $\boldsymbol{x}_t$ 是比 $\boldsymbol{x}_{t-1}$ 更加“模糊”的图片，它对分类不会有帮助。

分类器引导的采样(续)

$$p(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y) = \frac{p(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)p(y \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_t)}{p(y \mid \boldsymbol{x}_t)} = \frac{p(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)p(y \mid \boldsymbol{x}_{t-1})}{p(y \mid \boldsymbol{x}_t)}$$

利用泰勒展开可以写出

$$\ln p(y \mid \boldsymbol{x}_{t-1}) - \ln p(y \mid \boldsymbol{x}_t) \approx (\boldsymbol{x}_{t-1} - \boldsymbol{x}_t) \cdot \nabla_{\boldsymbol{x}_t} \ln p(y \mid \boldsymbol{x}_t),$$

而 $\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t \sim N(\boldsymbol{\mu}(\boldsymbol{x}_t), \sigma_t^2 \boldsymbol{I})$，所以

$$\begin{aligned} p(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y) &\propto \mathrm{e}^{-\|\boldsymbol{x}_{t-1} - \boldsymbol{\mu}(\boldsymbol{x}_t)\|^2 / 2\sigma_t^2 + (\boldsymbol{x}_{t-1} - \boldsymbol{x}_t) \cdot \nabla_{\boldsymbol{x}_t} \ln p(y|\boldsymbol{x}_t)} \\ &\propto \mathrm{e}^{-\|\boldsymbol{x}_{t-1} - \boldsymbol{\mu}(\boldsymbol{x}_t) - \sigma_t^2 \nabla_{\boldsymbol{x}_t} \ln p(y|\boldsymbol{x}_t)\|^2 / 2\sigma_t^2}, \end{aligned}$$

从而 $\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, y \sim N(\boldsymbol{\mu}(\boldsymbol{x}_t) + \sigma_t^2 \nabla_{\boldsymbol{x}_t} \ln p(y \mid \boldsymbol{x}_t), \sigma_t^2 \boldsymbol{I})$。

分类器引导的采样(续)

于是为了生成满足条件 y 的图片，只需要按

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}(\boldsymbol{x}_t) + \sigma_t^2 \nabla_{\boldsymbol{x}_t} \ln p(y \mid \boldsymbol{x}_t) + \sigma_t \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \boldsymbol{I})$$

进行采样。在这里 $p(y \mid \boldsymbol{x}_t)$ 可以用一个分类网络 $\phi(\boldsymbol{x})$ 近似代替，对于一对数据 $(\boldsymbol{x}_0, y)$，我们希望

$$\phi(\boldsymbol{x}_t) = y,\ \boldsymbol{x}_t = \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\varepsilon}.$$

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}(\boldsymbol{x}_t) + \sigma_t^2 \nabla_{\boldsymbol{x}_t} \ln p(y \mid \boldsymbol{x}_t) + \sigma_t \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \boldsymbol{I})$$

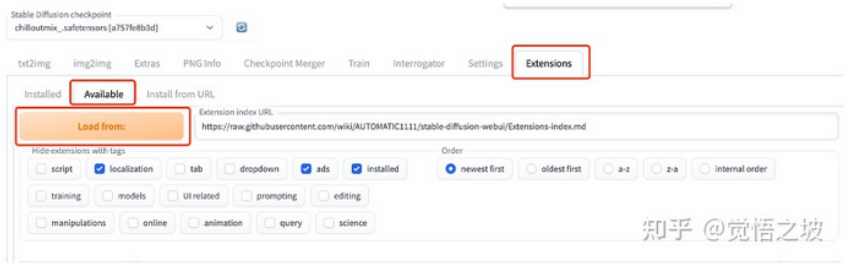
在这里，y 不必是类别类型的条件。当它是文本描述、参考图等类型的条件时，我们只需要用合适的神经网络 $\phi(\boldsymbol{x}_t, y, t)$ 来代替 $\ln p(y \mid \boldsymbol{x}_t)$，我们叫它「引导函数」。

2.ControlNet

1.首先你需要安装sdwebui，如果没安装的请点击（mac版教程），（win版教程）安装。

2.安装 Controlnet插件

打开stable-diffusion-webui，点击最右侧的extension，接着点击available，再点击“Load From:”按钮。



这时候页面下面就会出现一系列的可用插件。

注意：

这里有时候点击load from按钮之后，页面会报错说fail连接失败。

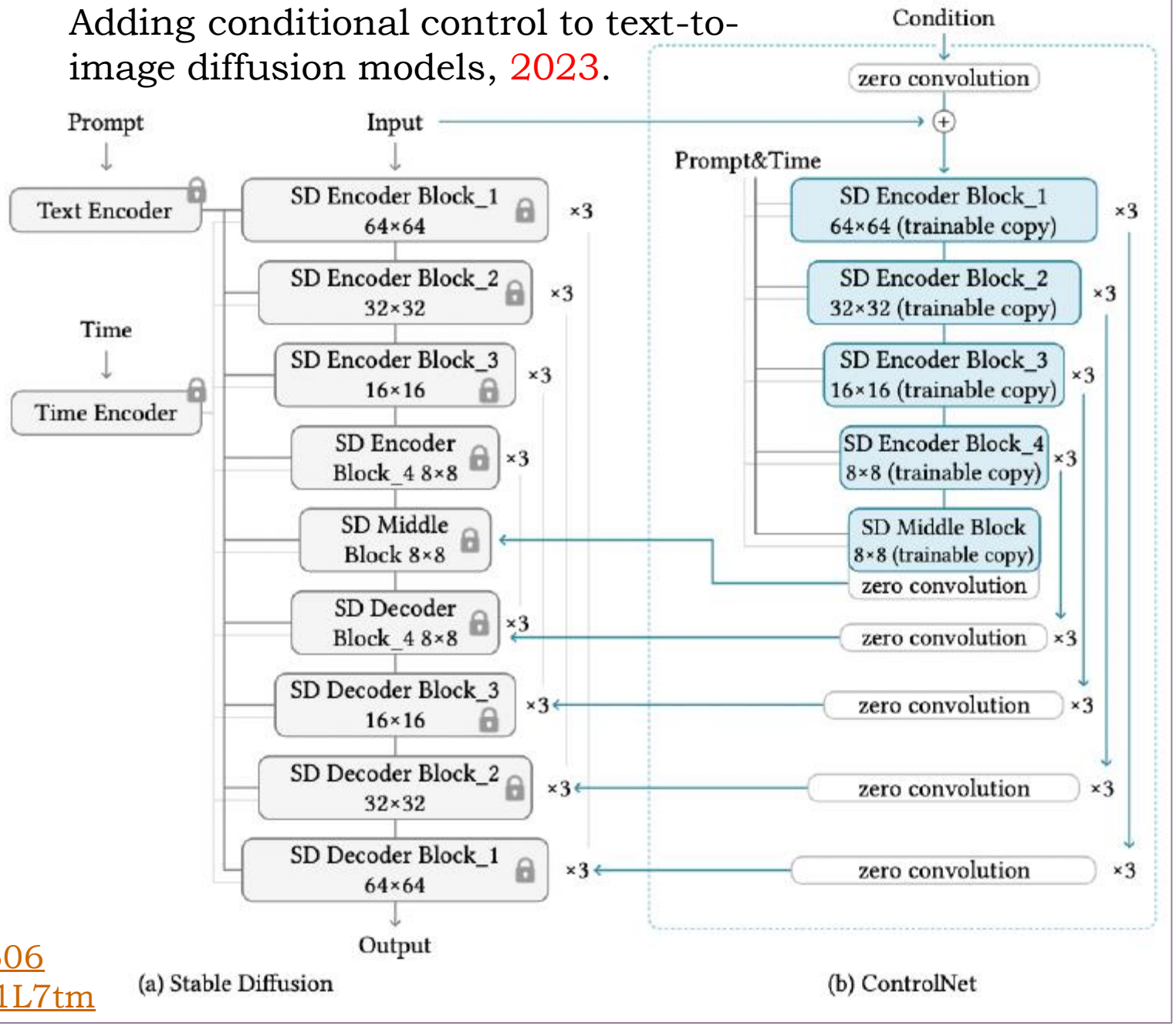
别急，刷新页面，调试好网络，开启/切换内外网，再点击一次load from，试多几次就好。

接着，Ctrl/command + F 查找“controlnet”，找到controlnet插件，并点击右侧的install安装这个插件。

安装完成后，插件面板会显示 intalled into xxx，说明已经安装好了。

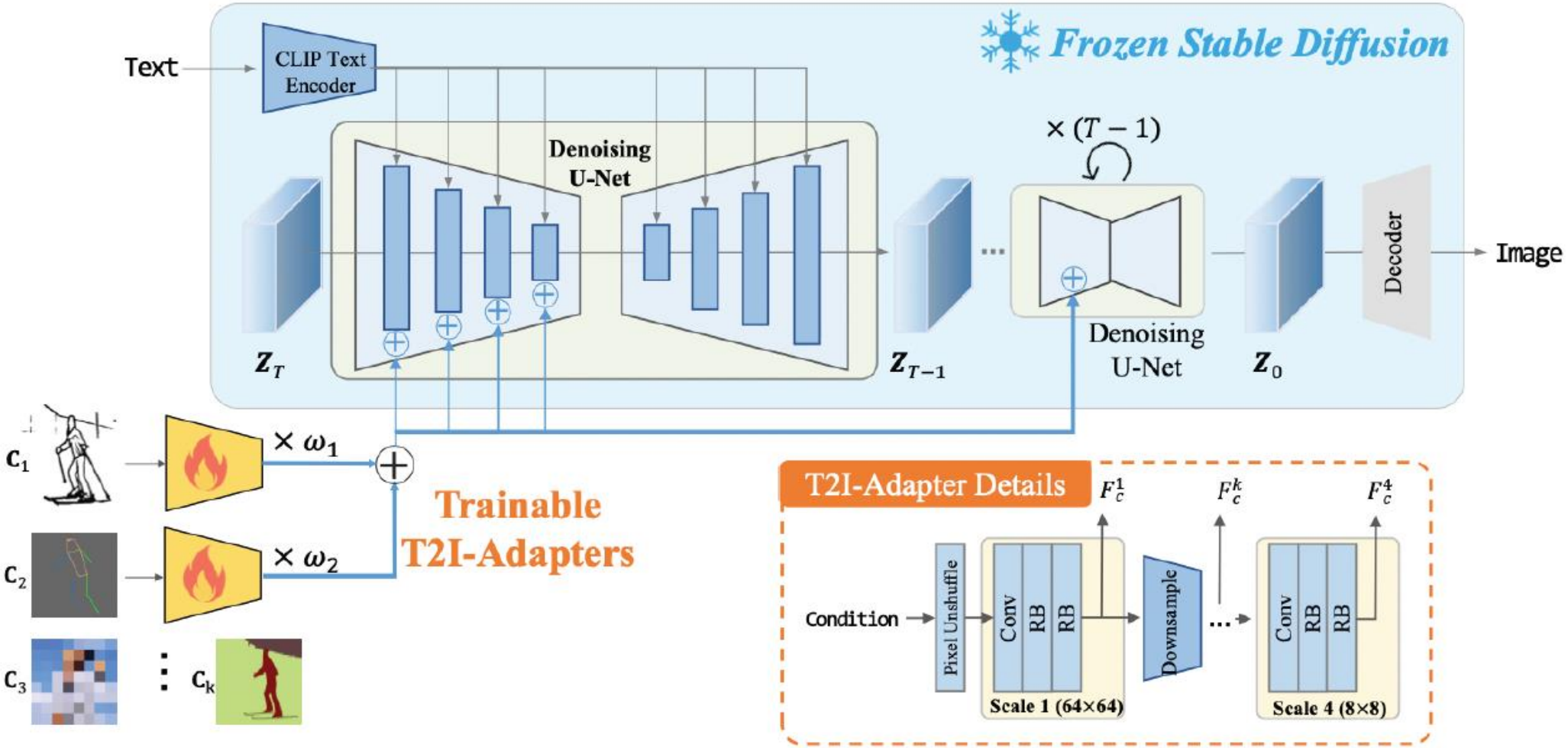
https://zhuanlan.zhihu.com/p/612094506
https://www.bilibili.com/video/BV1fo4y1L7tm

Adding conditional control to text-to-image diffusion models, 2023.



T2I-Adapter

T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models, 2023.



Uni-ControlNet

Uni-ControlNet 把控制条件分成两类：

- 局部条件：加到UNet 编码器的结果上，再与解码器跳跃连接
- 全局条件：产生一个Extended Prompt 加入网络

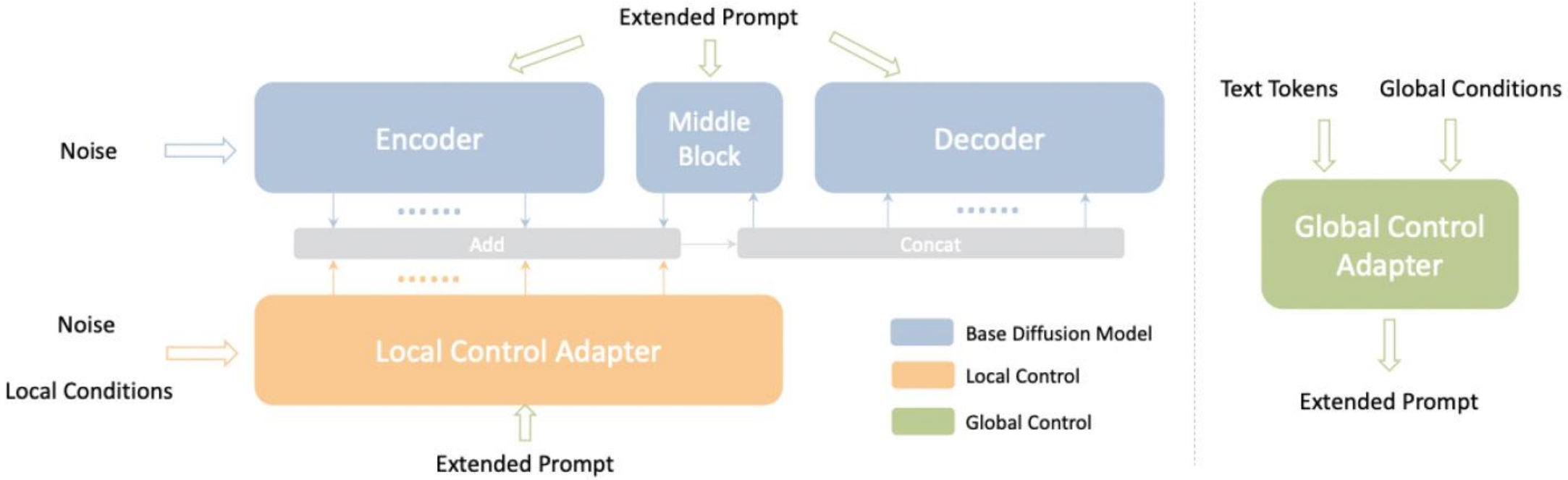


图: Uni-ControlNet 的结构

Uni-ControlNet 的生成结果展示

A man walking on the street, sunny day

A yellow flower in a white plate with a fork and a knife

A house on hills, blue sky

Local

+

Local

A cute cat and balloons

A cat on the sofa in an old room, art painting

A photo of a cowboy riding a horse on the street