VLA内部学习资料

Ruijing Xu

一、网络资源

1.常用网站

Arxiv

Huggingface

2.Github

- **知识体系类仓库**

https://github.com/TianxingChen/Embodied-AI-Guide

https://github.com/jiangranlv/embodied-ai-start

- **文献综述类仓库**

https://github.com/yueen-ma/Awesome-VLA

https://github.com/jonyzhang2023/awesome-embodied-vla-va-vln

- **经典模型**

> CSDN上往往有关于经典论文的精读，非常利于学习

OpenVLA

pi系列

3.学习资源（默认CSDN）

- **深度学习**

（Bilibili）飞天闪客，一个小时从函数到Transformer

- **Transformer**

“Attention is all you need” 论文原文

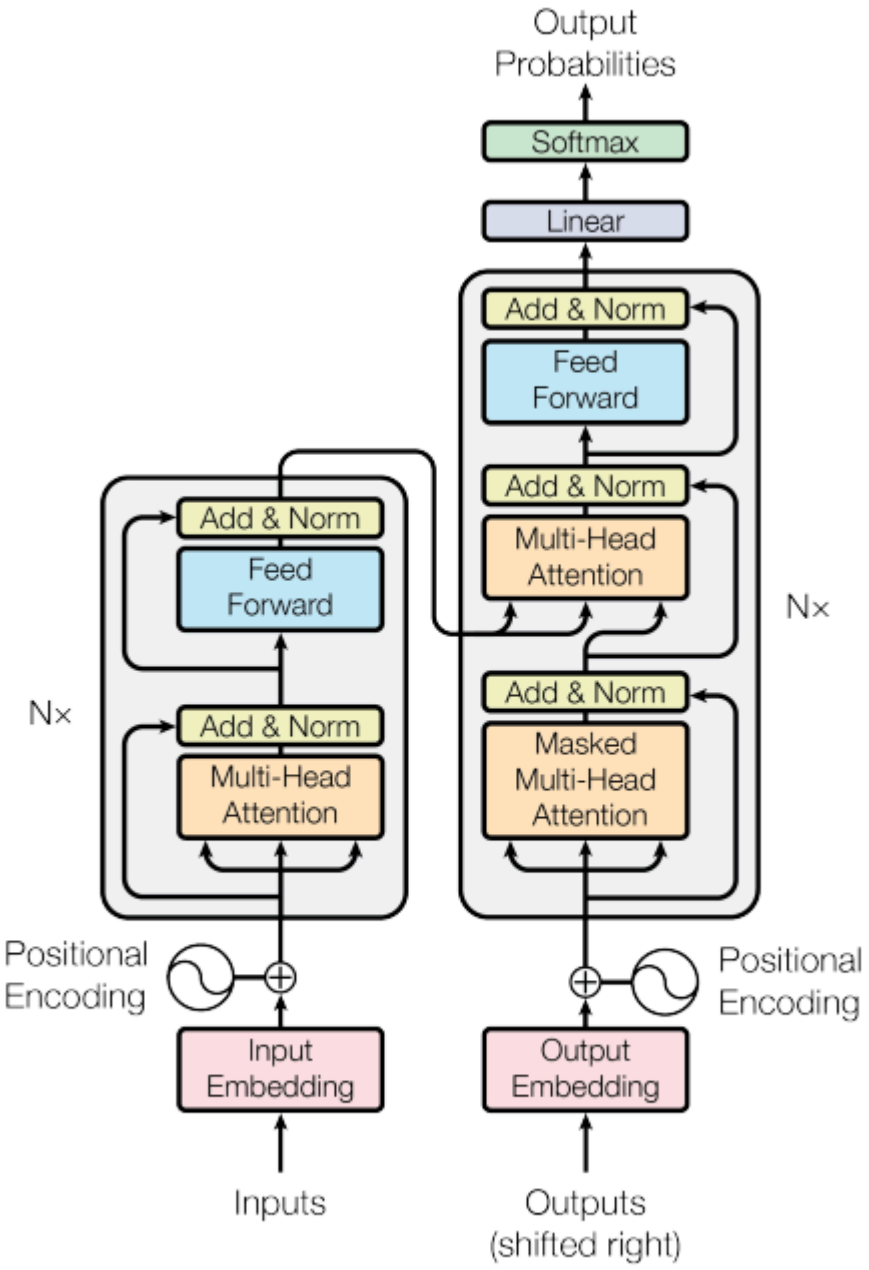


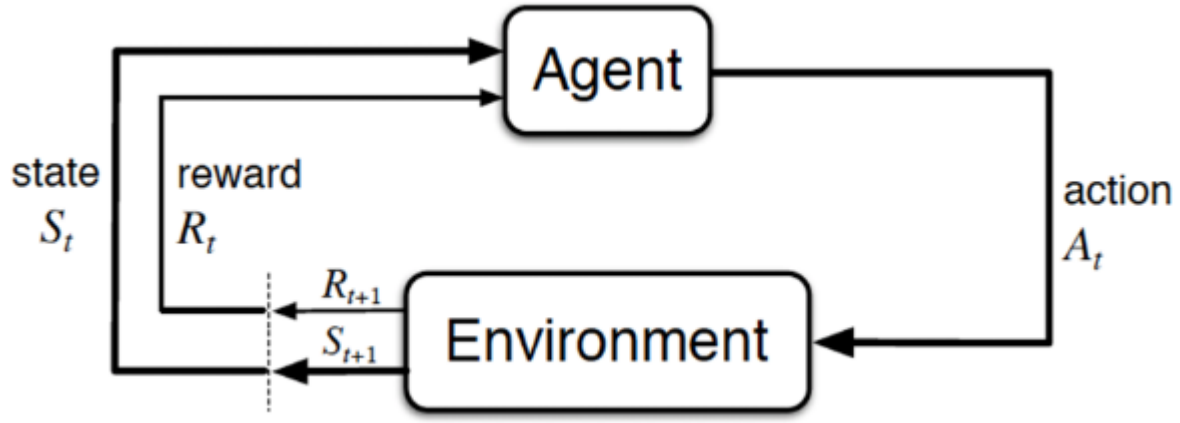
Figure 1: The Transformer - model architecture.

（Bilibili）论文逐段精读

（CSDN）一文读懂Transformer

- **强化学习**

强化学习极简入门：通俗理解MDP、DP MC TD和Q学习、策略梯度、PPO



进一步，RL为得到最优策略从而获取最大化奖励，有

- **基于值函数的方法**，通过求解一个状态或者状态下某个动作的估值为手段，从而寻找最佳的价值函数，找到价值函数后，再提取最佳策略
  比如Q-learning、DQN等，适合离散的环境下，比如围棋和某些游戏领域
- **基于策略的方法**，一般先进行策略评估，即对当前已经搜索到的策略函数进行估值，得到估值后，进行策略改进，不断重复这两步直至策略收敛

  比如策略梯度法(policy gradient，简称PG)，适合连续动作的场景，比如机器人控制领域
  以及Actor-Critic(一般被翻译为演员-评论家算法)，Actor学习参数化的策略即策略函数，Critic学习值函数用来评估状态-动作对，不过，Actor-Critic本质上是属于基于策略的算法，毕竟算法的目标是优化一个带参数的策略，只是会额外学习价值函数，从而帮助策略函数更好的学习

  此外，还有对策略梯度算法的改进，比如TRPO算法、PPO算法，当然PPO算法也可称之为是一种Actor-Critic架构，下文会重点阐述

- 首先，通过“**状态价值函数**”对当前状态进行评估

$$\begin{aligned} V_\pi(s) &= E_\pi[G_t|S_t=s] \\ &= E_\pi[R_{t+1}+\gamma G_{t+1}|S_t=s] \\ &= E_\pi[R_{t+1}+\gamma V_\pi(S_{t+1})|S_t=s] \end{aligned}$$

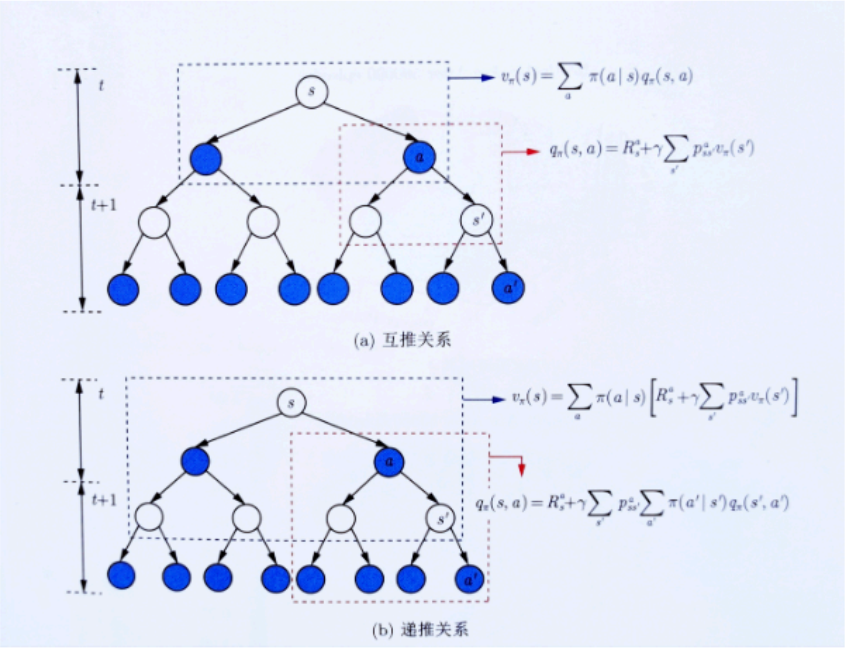
  相当于从状态s出发遵循策略π能获得的期望回报

- 其次，通过“**动作价值函数**”对动作的评估

$$\begin{aligned} Q_\pi(s,a) &= E_\pi[G_t|S_t=s,A_t=a] \\ &= E_\pi[R_{t+1}+\gamma G_{t+1}|S_t=s,A_t=a] \\ &= E_\pi[R_{t+1}+\gamma Q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a] \end{aligned}$$

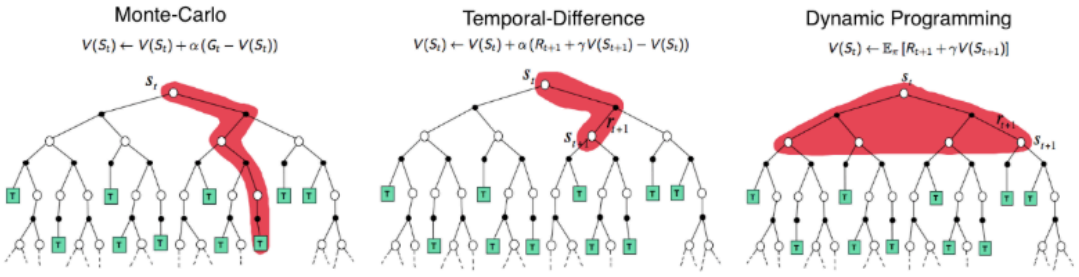
  相当于对当前状态s依据策略π执行动作a得到的期望回报，这就是大名鼎鼎的Q函数，得到Q函数后，进入某个状态要采取的最优动作便可以通过Q函数得到

接下来，把上面$V_\pi(s)$和$Q_\pi(s,a)$的计算结果互相代入，可得**马尔可夫决策的贝尔曼方程**



1. 因为有
   $V_\pi(s)=\sum_{a\in A}\pi(a|s)Q_\pi(s,a)$
   $Q_\pi(s,a)=R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)V_\pi(s')$
   故有
   $$V_\pi(s)=\sum_{a\in A}\pi(a|s)\left[R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)V_\pi(s'))\right]$$
2. 因为
   $Q_\pi(s,a)=R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)V_\pi(s')$
   $V_\pi(s)=\sum_{a\in A}\pi(a|s)Q_\pi(s,a)$
   故有
   $$Q_\pi(s,a)=R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)\left[\sum_{a'\in A}\pi(a'|s')Q_\pi(s',a')\right]$$

动态规划、蒙特卡洛、时序差分的区别：



顺带再举一个例子，好比行军打仗时，为了得到更好的行军路线，将军派一人前去探路

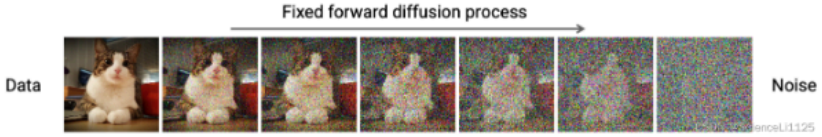
- MC的做法相当于一条道走到黑 没走个10公里不回头
- DP相当于所有道比如10条道 每条道都走个1公里 不错过任何一条可能成为最好道的可能，最后10条道都走完1公里后才返回汇报/反馈
- TD则相当于先选一条道走个1公里即返回汇报/反馈，之后再走下一条道的1公里

- **Diffusion**

Diffusion：通过扩散和逆扩散过程生成图像的生成模式

1. 扩散过程

如上文所说，扩散过程 $x_0 \rightarrow x_T$ 对图片逐渐加噪。x_0 是从真实数据集中采样得到的一张图片，对 x_0 逐渐添加具有高斯分布的噪声，图片逐会变得模糊，当 T 足够大时，x_T 为标准正态分布。由于每次添加的噪声是已知的，即 $q(x_{t+1} \mid x_t)$ 是已知的，因此可以根据马尔科夫过程的性质，递归得到 $q(x_T \mid x_0)$ 。

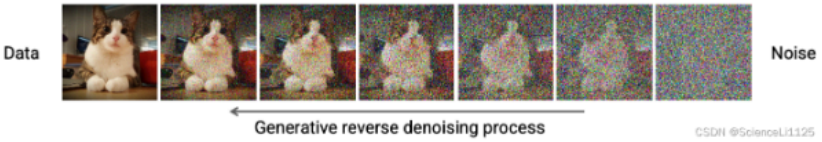


此处不再赘述公式推导的过程，详见 What are Diffusion Models?、扩散模型（Diffusion Model）——由浅入深的理解、由浅入深了解 Diffusion Model、Diffusion扩散模型大白话讲解，看完还不懂？不可能！ 等。

2. 逆扩散过程

如上文所说，逆扩散过程 $x_T \rightarrow x_0$ 对图片去噪复原来生成图像。由于扩散过程的 T 取足够大时，x_T 为标准正态分布，因此 要想生成图片可以直接从标准正态分布的图像逆扩散回去就可以得到想要的图像。于是问题的关键就是学习逆扩散过程。

逆扩散不像前向过程每一步是固定的，逆扩散很难从后一张图像中去噪得到前一张图像，即 $q(x_{t-1} \mid x_t)$ 是未知的。因此只能用 $p_\theta(x_{t-1} \mid x_t)$ 来近似代替 $q(x_{t-1} \mid x_t)$，于是 逆扩散的学习过程就成了训练 $p_\theta(x_{t-1} \mid x_t)$ 网络，文中称为 U-Net。



虽然 $q(x_{t-1} \mid x_t)$ 是未知的，但 $q(x_{t-1} \mid x_0 x_t)$ 是可知的。因此我们可以用 $q(x_{t-1} \mid x_0 x_t)$ 来指导 $p_\theta(x_{t-1} \mid x_t)$ 进行训练。此处不赘述公式推导的过程。

Diffusion Policy

- **Sim2Real**

Sim2Real学习总结：A Short Survey

- **经典的小型机器人/臂实体**

UMI

Mobile ALOHA

LeRobot

二、经典概念解释

- **ReLU**

🌟 1. ReLU 是什么？（一句话）

ReLU（Rectified Linear Unit）是一种简单的非线性激活函数：

$$\mathrm{ReLU}(x) = \max(0, x)$$

也就是：

- x > 0 → 输出 x
- x ≤ 0 → 输出 0

就是这么简单。

- **LLaMA**

LLaMA = Meta 发布的大语言模型（Large Language Model）。

全称：**L**arge **La**nguage **M**odel from **A**Meta（取谐音 llama）。

它本质上是一种 **Transformer-based 自回归语言模型（decoder-only）。**

LLaMa系列模型详解（原理介绍、代码解读）：LLaMa

- **ViT**

1. ViT简介

ViT 全称 Vision Transformer，不同于传统的基于CNN的网络结果，是基于transformer结构的cv网络。

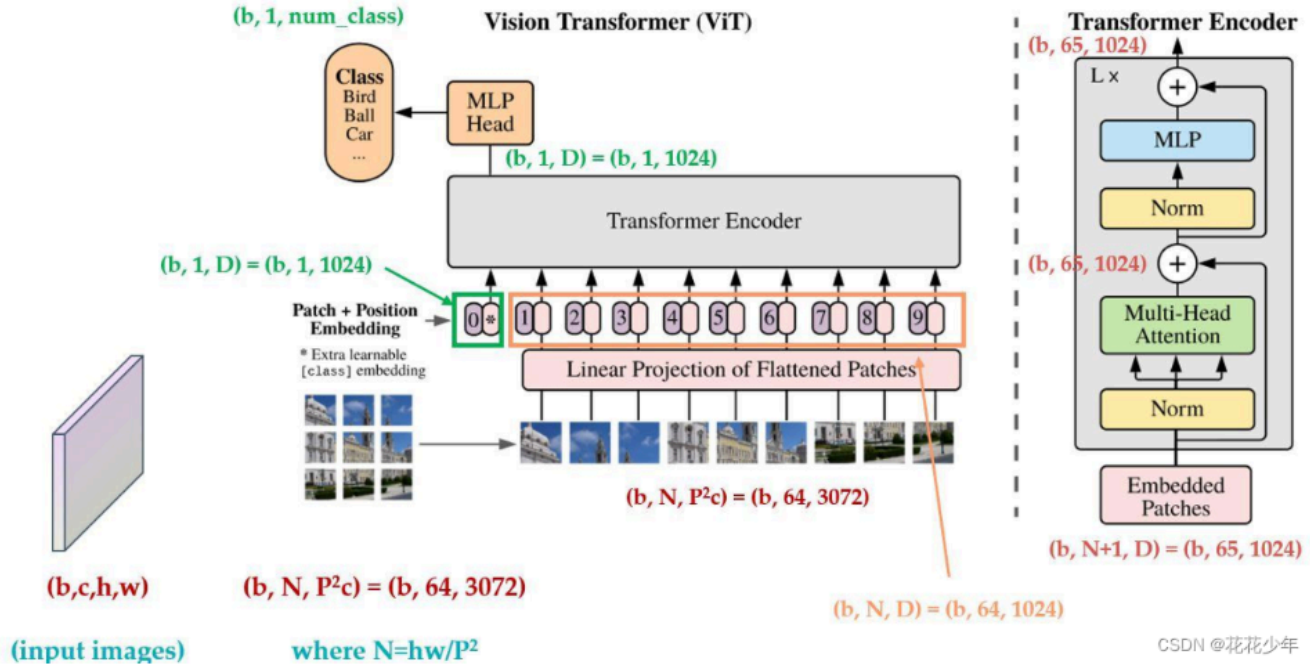
2. ViT模型特点

ViT模型主要应用于图像分类领域。因此，其模型结构相较于传统的Transformer有以下几个特点：

- 数据集的原图像被划分为多个Patch后，通过Patch Embedding将二维Patch（不考虑channel）转换为一维向量，再加上**类别向量**与**位置向量**作为模型输入。
- 模型主体的Block结构是基于Transformer的Encoder结构，但是调整了Normalization的位置，其中，最主要的结构依然是Multi-head Attention结构。
- 模型在Blocks堆叠后接**全连接层**，接受类别向量的输出作为输入并用于分类。通常情况下，我们将最后的全连接层称为Head，Transformer Encoder部分为backbone。

ViT模型利用Transformer模型在处理上下文语义信息的优势，将图像转换为一种"变种词向量"然后进行处理，而这种转换的意义在于，多个Patch之间本身具有空间联系，这类似于一种"空间语义"，从而获得了比较好的处理效果。

3. ViT数据流



三、Pi系列基础模型

pi系列原生的项目仓库

- **Pi 0**

网站

github仓库

论文原文

论文精读

复现

- **Pi 0.5**

网站

论文原文

论文精读

目前疑似并未完全开源。