



北京航空航天大學
BEIHANG UNIVERSITY

Large Language Models

Contents

1 通用大语言模型

2 大语言模型基础

2.1 概述

2.2 分词

2.3 自回归解码

2.4 评估指标

3 大语言模型训练技术

3.1 预训练

3.2 监督微调 (SFT)

3.3 强化学习 (RLHF)

3.4 参数高效微调

3.5 混合专家 (MoE)

3 大语言模型推理/训练优化技术

3.1 推理/训练加速

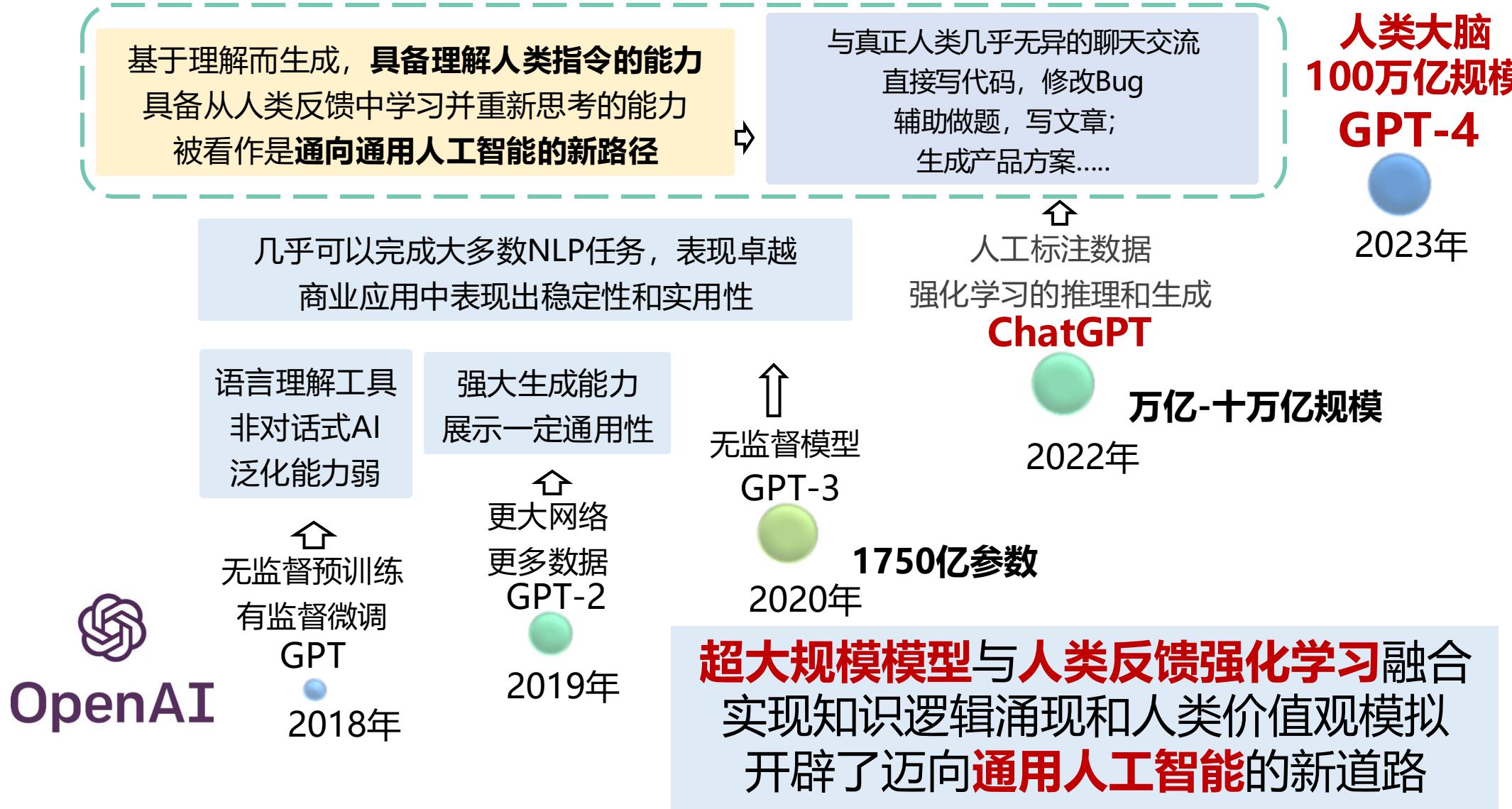
3.2 检索增强生成 (RAG)

Part

01

通用大语言模型

生成式人工智能



图像生成



“大漠孤烟直，长河落日圆”

代码生成

A screenshot of a ChatGPT interface. The user has uploaded a hand-drawn sketch of a website layout. The AI has generated a corresponding block of HTML and CSS code. The code includes basic structure like `<html>`, `<head>`, and `<body>`, along with CSS styles for headings, paragraphs, and other elements based on the sketch.

基于手绘草图
生成网站代码

PDF理解

A screenshot of a ChatGPT interface demonstrating its ability to analyze PDF files. The user has uploaded a PDF file, and the AI has responded with a message indicating it has loaded the file and is ready to answer questions about its contents. Below this, another message shows the AI responding to a request for a summary of the PDF's key points. The interface also shows a list of steps for analyzing the PDF, such as generating data and using the ChatWithPDF plugin.



百模大战

➤ GPT后各家LLM爆发式增长

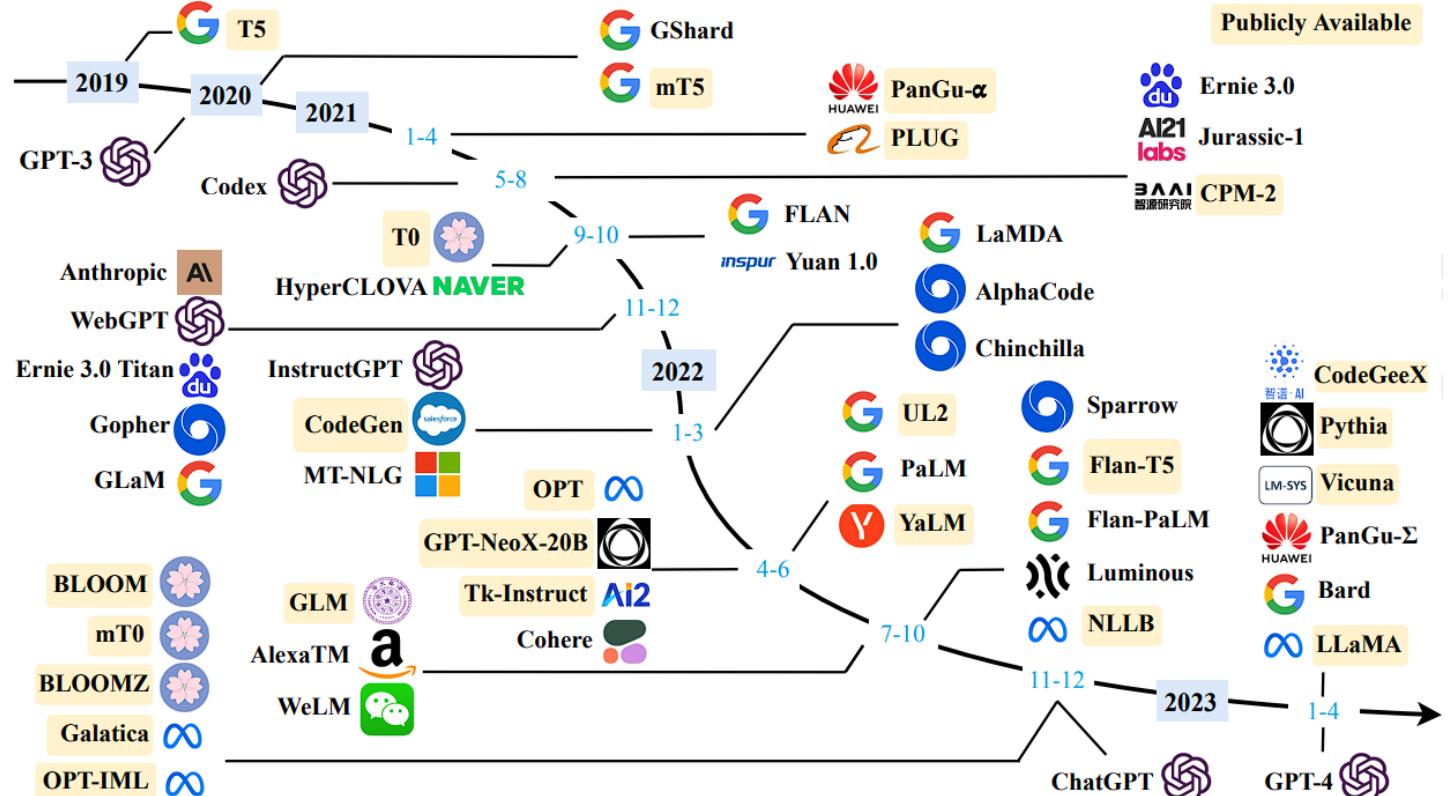
- 百度：文心
- 腾讯：混元
- 阿里：通义
- 字节：豆包
- 商汤：书生
- 华为：盘古

➤ 创业公司雨后春笋

- 月之暗面
- 智谱AI
- MiniMax

➤ 各大公司独立发展自己的基座模型

- 基于此构建出各种下游应用（医疗、金融、对话、搜索、Code）
- 赋能公司各项业务（搜广推等）



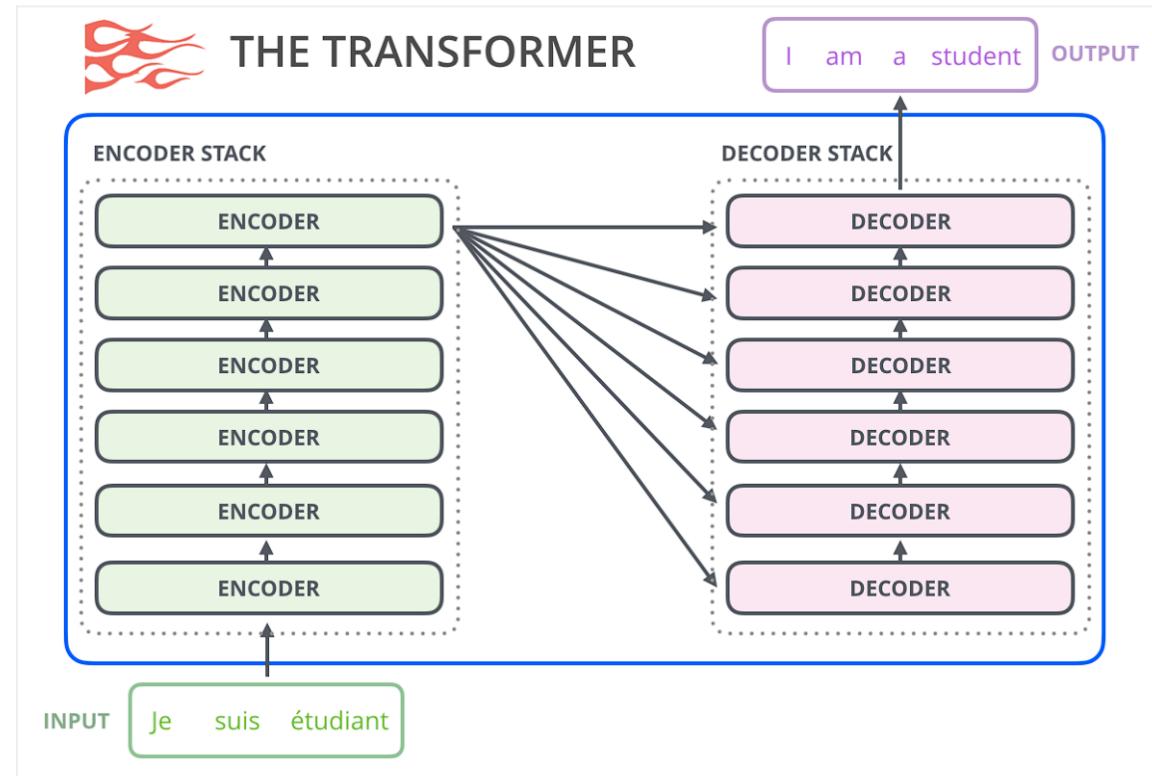
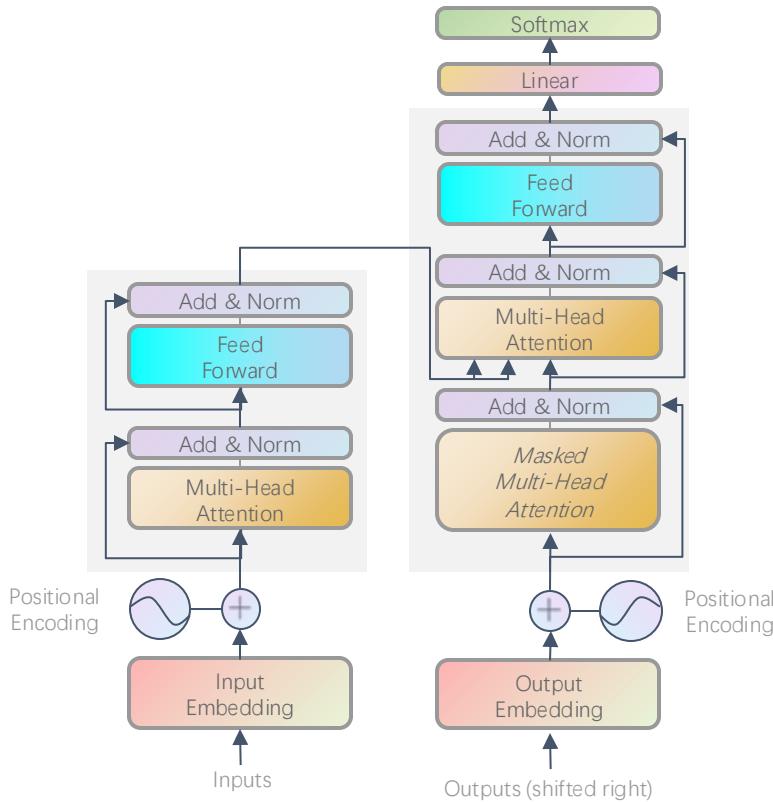
Part

02

LLM基础

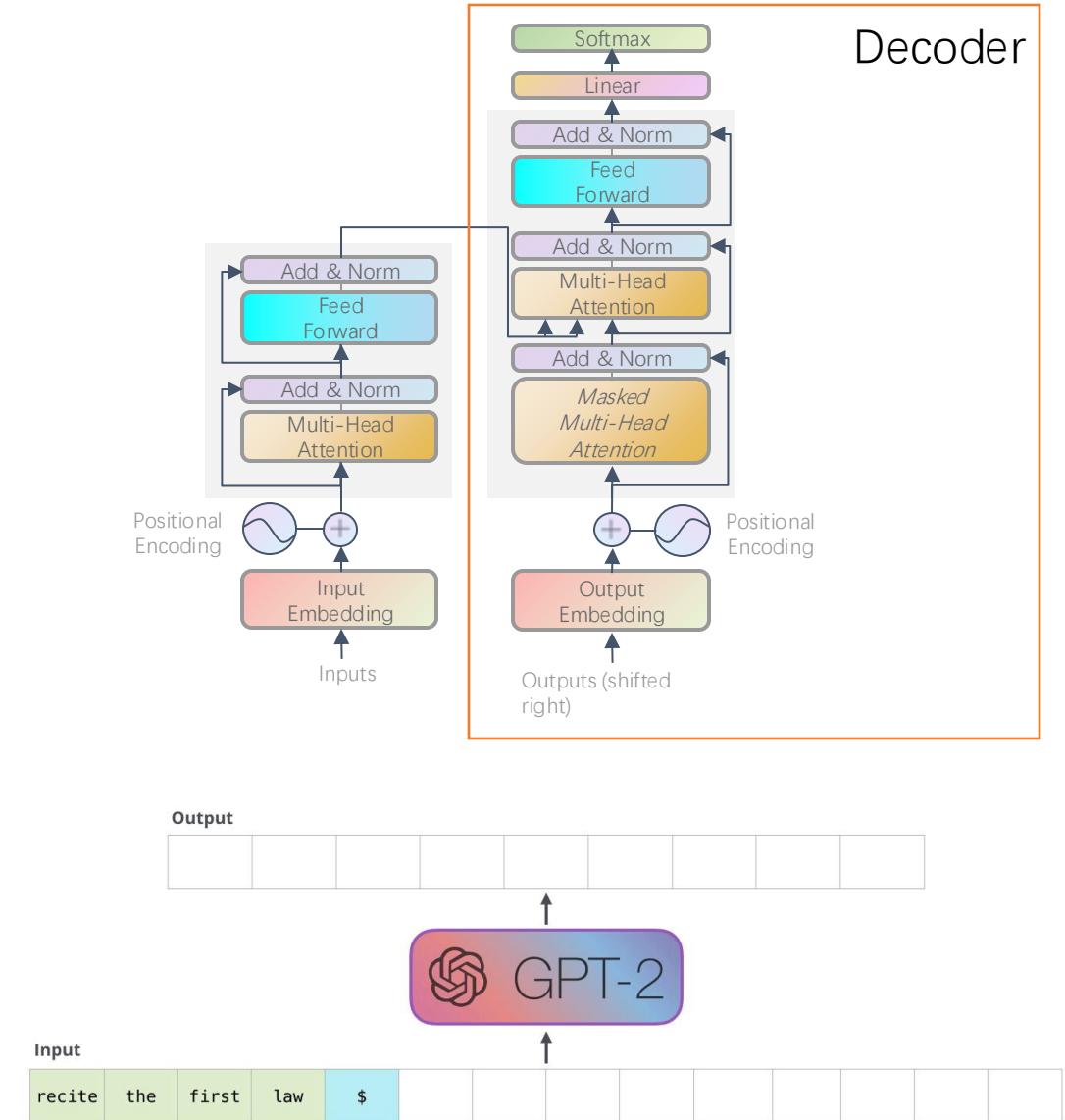
回顾：Transformer Block

- Transformer Encoder & Decoder 架构在机器翻译等问题上空前成功
- 后续工作仅使用Encoder (BERT) 或Decoder (GPT)



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Decoder Only的Transformer 结构
 - 工程原因：提高训练效率，具有工程优势，同等参数量与推理成本下是最优的选择
 - 理论原因：仍是一个开放问题（后续讨论）
- Tokenizer：
 - 输入数据（文本、图像、音视频）被转化为逐个Token（数字）再转化为可学习词嵌入
- 自回归解码：
 - LLM一次推理只预测下一个Token（词）
 - 预测的下一个Token接在输入Tokens的后面，继续预测下下个Token，直到输出代表结束的Token(<EOS>)。
 - 因此输出完整回答需要LLM多次推理
- 输出：
 - 类似分类任务，输出一个分布代表下一个Token



Tokenizer

- 将文本转化为整数序列，作为Transformer的输入
 - 本质就是把单词映射到索引
 - 不完全是一个词作为一整个Token
 - 每个单词平均被转化为1.5个token (sub words)
 - 不同LLM的Tokenizer不完全一样
 - <https://huggingface.co/spaces/Xenova/the-tokenizer-playground>

Token Embeddings (wte)

The diagram illustrates a word embedding matrix. A vertical orange line on the left indicates the "model vocabulary size" of 50,257 words. A horizontal orange line at the bottom indicates the "embedding size" of 100 dimensions. The matrix itself consists of 50,257 rows and 100 columns. Each row contains a word from the vocabulary, such as "aardvark", "aarhus", "aaron", followed by several ellipses, and ending with "zyzzyva". Each word is represented by a 100-dimensional vector in the column space.

原始文本

The GPT family of models process text using tokens, which are common sequences of characters found in text. The models understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens.

You can use the tool below to understand how a piece of text would be tokenized by the API, and the total count of tokens in that piece of text.

Tokens

The GPT family of models process text using tokens, which are common sequences of characters found in text. The models understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens.

You can use the tool below to understand how a piece of text would be tokenized by the API, and the total count of tokens in that piece of text.

Input ids 整数序列

[464, 402, 11571, 1641, 286, 4981, 1429, 2420, 1262, 16326, 11, 543, 389, 2219, 16311, 286, 3435, 1043, 287, 2420, 13, 383, 4981, 1833, 262, 13905, 6958, 1022, 777, 16326, 11, 290, 27336, 379, 9194, 262, 1306, 11241, 287, 257, 8379, 286, 16326, 13, 198, 198, 1639, 460, 779, 262, 2891, 2174, 284, 1833, 703, 257, 3704, 286, 2420, 561, 307, 11241, 1143, 416, 262, 7824, 11, 290, 262, 2472, 954, 286, 16326, 287, 326, 3704, 286, 2420, 13]

Index Embeddings: [vocabulary size*hidden size]

Word-based Tokenizer

- 把文本划分为一个个词

"Don't you love 😊 Transformers? We sure do."

- 最简单的是按照空格分开： 标点？

```
["Don't", "you", "love", "😊", "Transformers?", "We", "sure", "do."]
```

Do not?

```
["Don", "", "t", "you", "love", "😊", "Transformers", "?", "We", "sure", "do", "."]
```

- 引入规则？事情开始复杂起来

- 相同意思的词是不同的token (dog, dogs)

- 词表非常大，资源消耗巨大

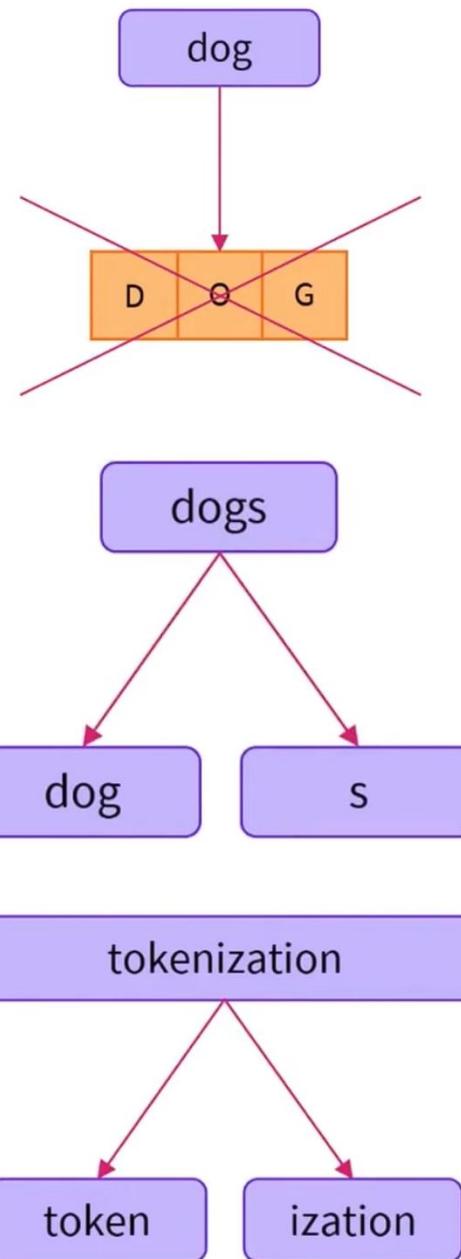
Character-based Tokenizer

- 干脆直接按照字母划分，最天然的计算机理解字符的方法
- 大大减小词汇表的内存占用
- 但！难以学习更有意义的表示
 - 单词信息量极少
 - 产生巨长的token序列，理解能力下降
 - 非西文字符词表依然巨大

Subword Tokenizer

- Words和character结合——subword
- 常用词只用一个Token表示
- 非常见词，如合成词等，用多个token表示
- 保证了词表在一个合适的范围

- 分词方法
 - Byte Pair Encoding
 - Unigram
 - SentencePiece



Byte Pair Encoding

- 使用基于规则的Word-based Tokenizer进行预Tokenizer

- 统计词频

```
( "hug" , 10) , ( "pug" , 5) , ( "pun" , 12) , ( "bun" , 4) , ( "hugs" , 5)
```

- 先得到base vocabulary

```
[ "b" , "g" , "h" , "n" , "p" , "s" , "u" ]
```

```
( "h" "u" "g" , 10) , ( "p" "u" "g" , 5) , ( "p" "u" "n" , 12) , ( "b" "u" "n" , 4) , ( "h" "u" "g" "s" , 5)
```

- 统计相邻token同时出现的频率，选取最高频率进行合并，更新词表

```
( "h" "ug" , 10) , ( "p" "ug" , 5) , ( "p" "u" "n" , 12) , ( "b" "u" "n" , 4) , ( "h" "ug" "s" , 5)
```

- 直到词表大小符合要求

- =合并次数+基础词表大小

- 缺点：

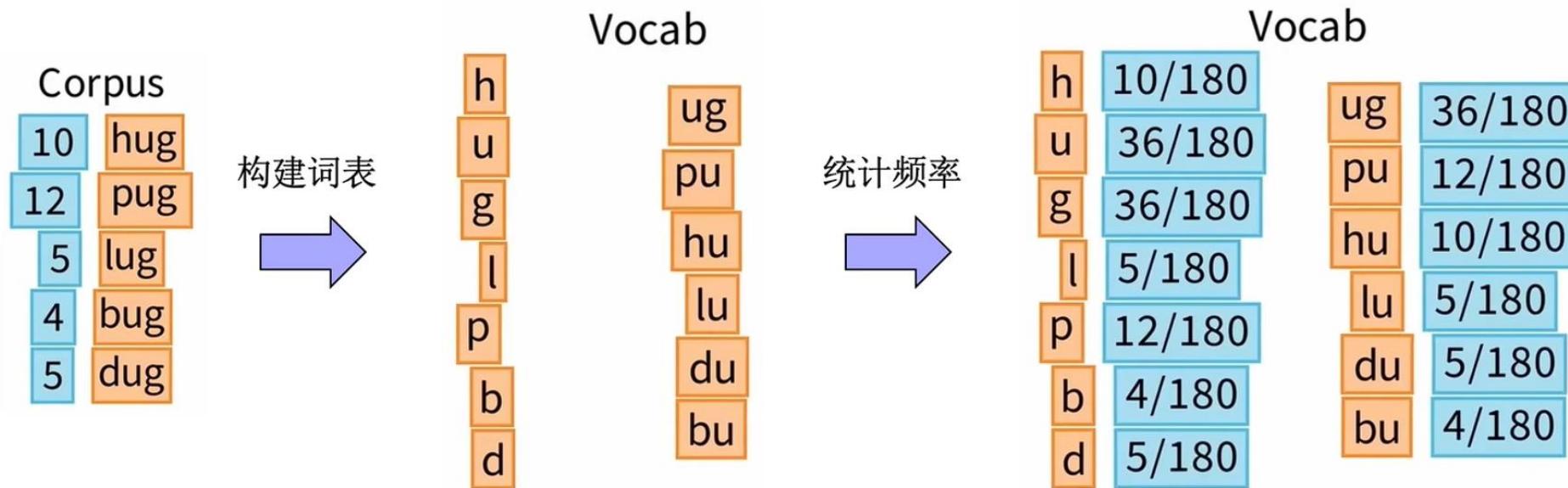
- 基本字符可能相当大，整个Unicode编码全是基本字符

- 改进？**Byte-level BPE**

- 字节视为基本token

- 两个字节合并就是Unicode

- 反过来，先初始化一个极大的词表，再逐个删除，每次删除固定比例的对总体 loss影响最小的Token
- 不单独使用，而是与SentencePiece结合
- 词汇切分时选取概率最大的切分方式（认为所有Token独立）
- 计算所有切分的负对数似然作为loss $\sum freq \times (-\log(P(word)))$



- 并非所有语言都通过空格切分
- 虽然可以应用特定语言的pre-tokenizer，但损失了通用性
- SentencePiece将输入视为字节流 (文本数据的一种底层表示形式，如UTF-8编码，具有语言无关性)
- 使用Byte-level PBE或unigram构建词汇表
- LLAMA、T5等模型应用的Tokenizer

自回归解码 (Casual Mask)

- LLM通常都是自回归解码

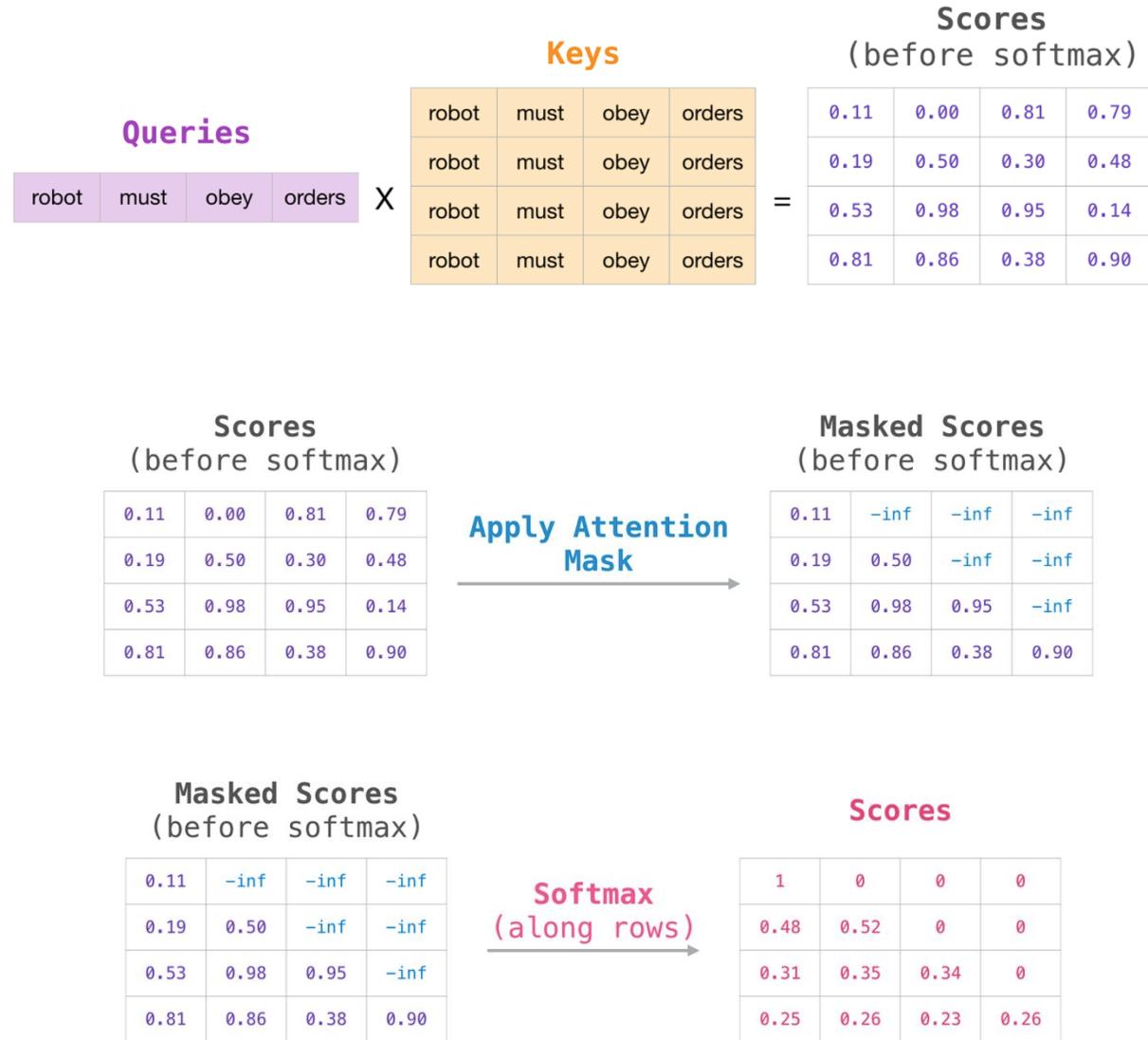
$$P(w_{1:T}|W_0) = \prod_{t=1}^T P(w_t|w_{1:t-1}, W_0), \text{with } w_{1:0} = \emptyset,$$

- 训练时如何对齐这种形式？

- Causal Mask
 - Labels 对应下一个 token

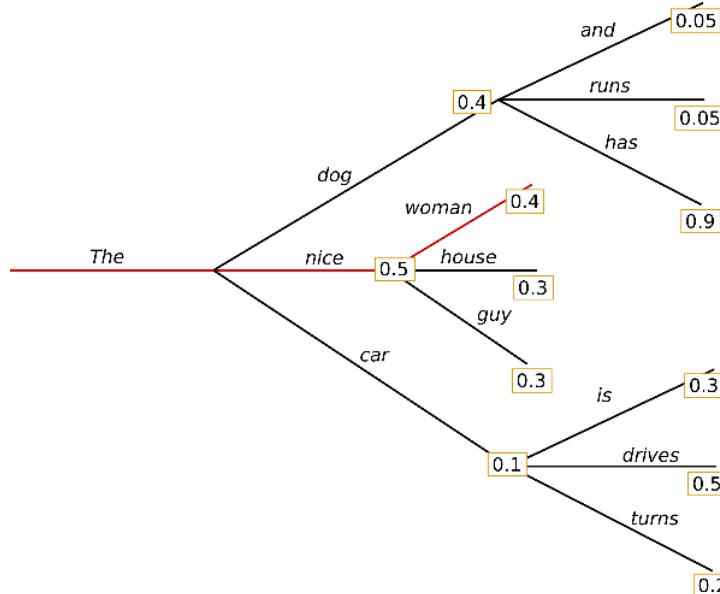
Features					
position:	1	2	3	4	
Example:	1	robot	must	obey	orders
	2	robot	must	obey	orders
	3	robot	must	obey	orders
	4	robot	must	obey	orders

Labels
must
obey
orders
<eos>



自回归解码——解码方案

- 推理 (generate) 时如何解码:
- 贪心? 每次选取概率最高的一个单词
 - The nice woman
 - 最终得分 $0.5 * 0.4 = 0.2$
 - 问题: 忽略 'has' 这种高概率单词 ($0.4 * 0.9 = 0.36$) ,
即忽视词组。



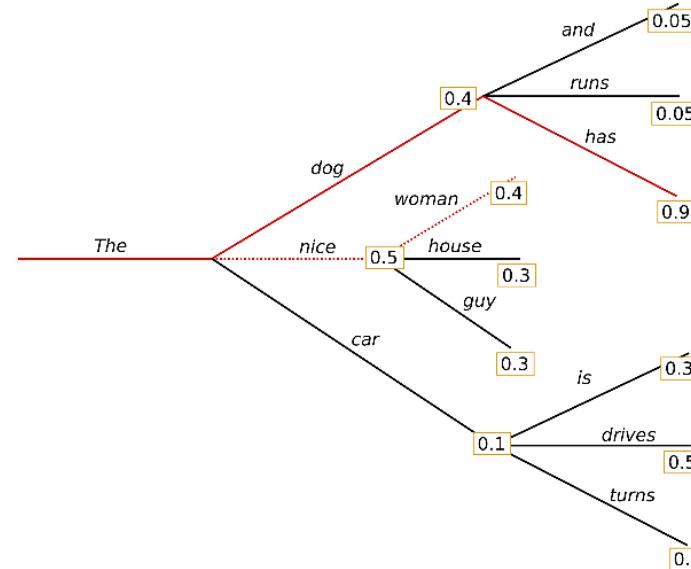
I enjoy walking with my cute dog

Output:

I enjoy walking with my cute dog, but I'm not sure if I'll ever be able to walk with my dog. I'm not sure if I'll ever be able to walk with my dog.
I'm not sure

自回归解码——解码方案

- Beam Search (波束搜索) , Num_beam=2
 - 保留可能的假设并最终选择总体概率最高
 - 降低丢失隐藏的高概率单词序列的风险
 - 比贪心搜索能找到概率更高的输出序列
- 实际的开放性生成中，这都不是最佳方案
 - 在所需生成长度基本可预测的任务中Beam Search表现良好（机器翻译等）、但实际上可能差异很大（对话）
 - 严重受重复生成的影响，并且没有好的控制方法 (n-gram)
 - 人类喜欢惊喜！



I enjoy walking with my cute dog

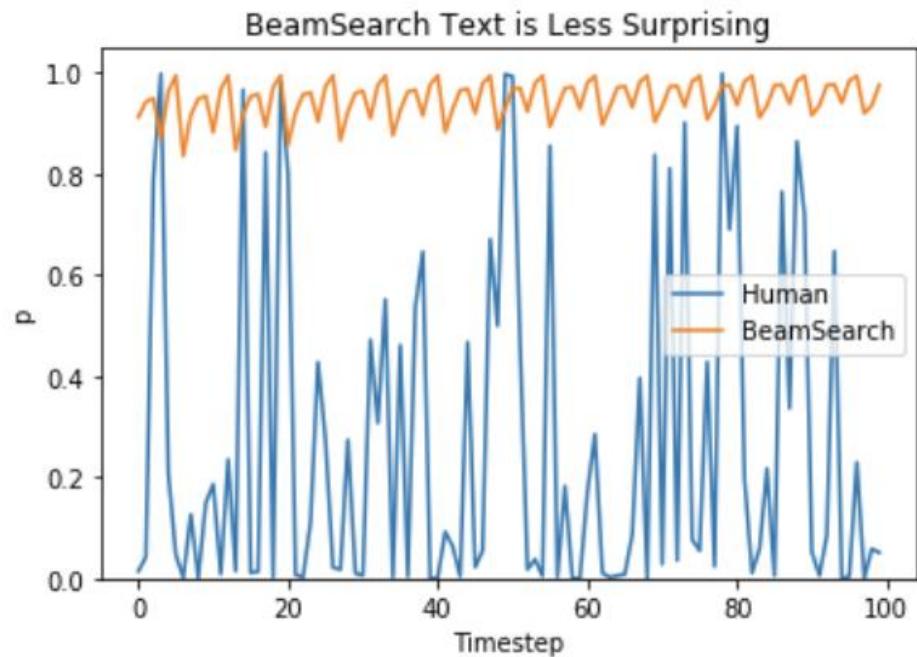
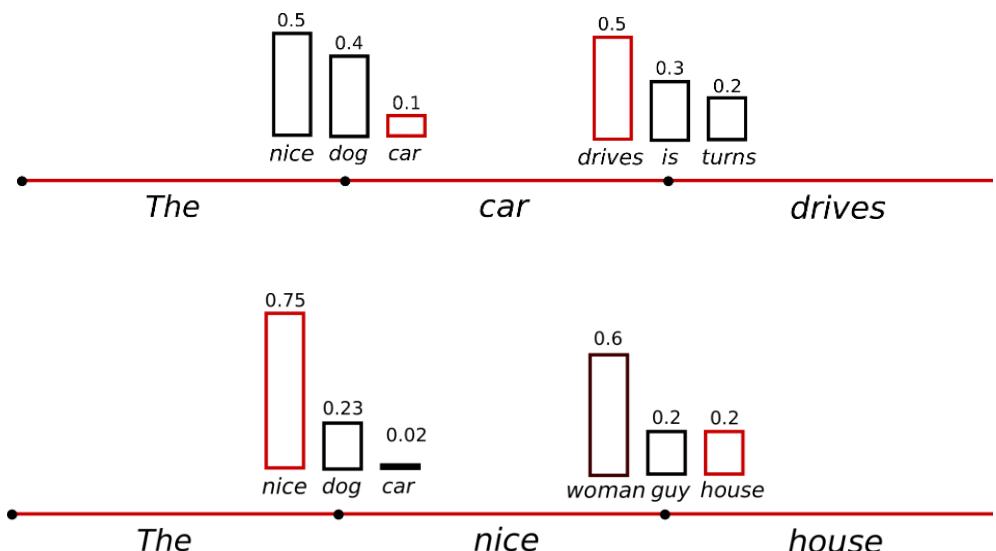
Output:

I enjoy walking with my cute dog, but I'm not sure if I'll ever be able to walk with him again.
I'm not sure if I'll ever be able to walk with him again. I'm not sure

自回归解码——采样

- 人类喜欢惊喜，不喜欢可预测。不喜欢无聊
- 高质量的人类语言不遵循高概率下一个单词的分布
- 按照输出的概率分布随机采样单词
- 问题：生成的文本可能会不连贯
 - 设置temperature: T越小越稳定

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_{j=1}^n e^{y_j}}$$



I enjoy walking with my cute dog

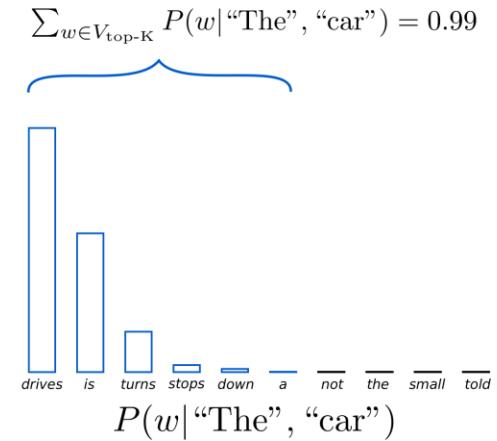
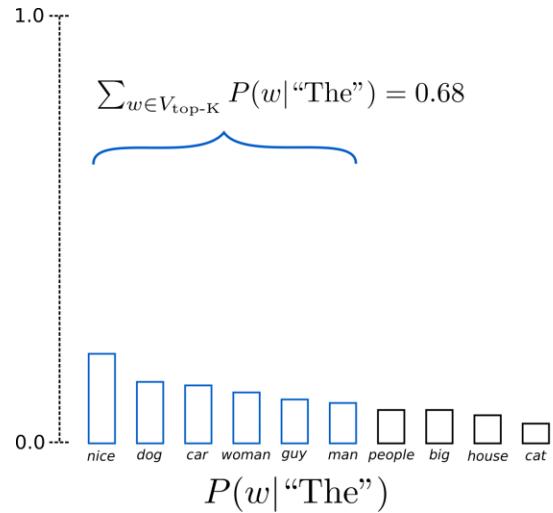
Output:

I enjoy walking with my cute dog for the rest of the day, but this had me staying in an unusual room and not going on nights out with friends (which will always be wondered for a mere minute or so at this point).

自回归解码——采样

Top-K采样

- 在所有后续词中选取K个最有可能的进行随机采样
- 有效过滤掉一些奇奇怪怪的单词
- 问题：如何选择K，K只能固定
 - 概率分布尖锐，过大的K导致乱码
 - 概率分布平坦，过小的K扼杀创造力



I enjoy walking with my cute dog

Output:

I enjoy walking with my cute dog for the rest of the day, but this time it was hard for me to figure out what to do with it. (One reason I asked this for a few months back is that I had a

自回归解码——采样

➤ Top-p采样（核采样）

- 从累积概率超过p的最小可能单词集中选择
- 天然的动态单词集
- Top-K和Top-P可以结合使用

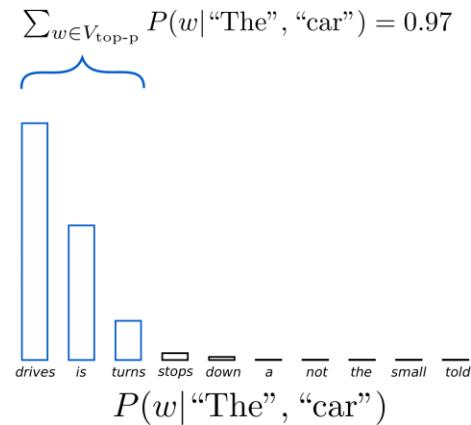
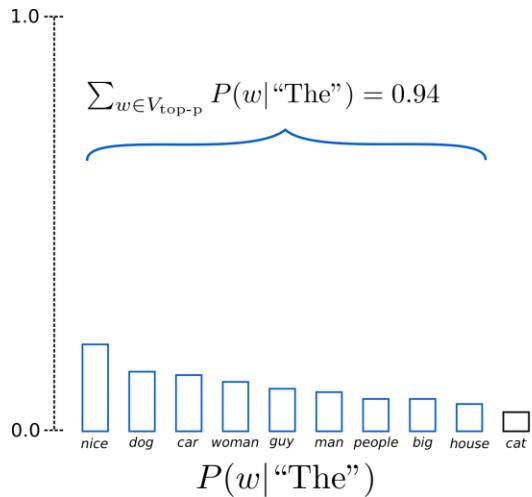
➤ Top-P天下无敌？

- 有证据表明贪心和BeamSearch出现重复序列的问题只是由于模型的训练方式引起的
- Top-P有时也会受重复序列影响
- 具体生成方式需要根据应用场景动态调整

I enjoy walking with my cute dog

Output:

I enjoy walking with my cute dog for the rest of the day, but this had me staying in an unusual room and not going on nights out with friends (which will always be my yearning for such a spacious screen on my desk



- 困惑度(Perplexity, PPL)
 - 给定一段高质量的文档作为测试集，模型生成这段文本的概率越高，就认为模型的困惑度越小，模型也就越好。
- 例如，测试集：“北航人工智能”
 - $PPL = P(\text{北}) * P(\text{航}|\text{北}) * P(\text{人}|\text{北航}) * P(\text{工}|\text{北航人}) * P(\text{智}|\text{北航人工}) * P(\text{能}|\text{北航人工智})^{-1/n}$
- 实际操作时往往以滑动窗口的方式计算

$$\text{perplexity}(W) = P(w_1, w_2, w_3, w_4, \dots, w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, w_3, w_4, \dots, w_n)}}$$

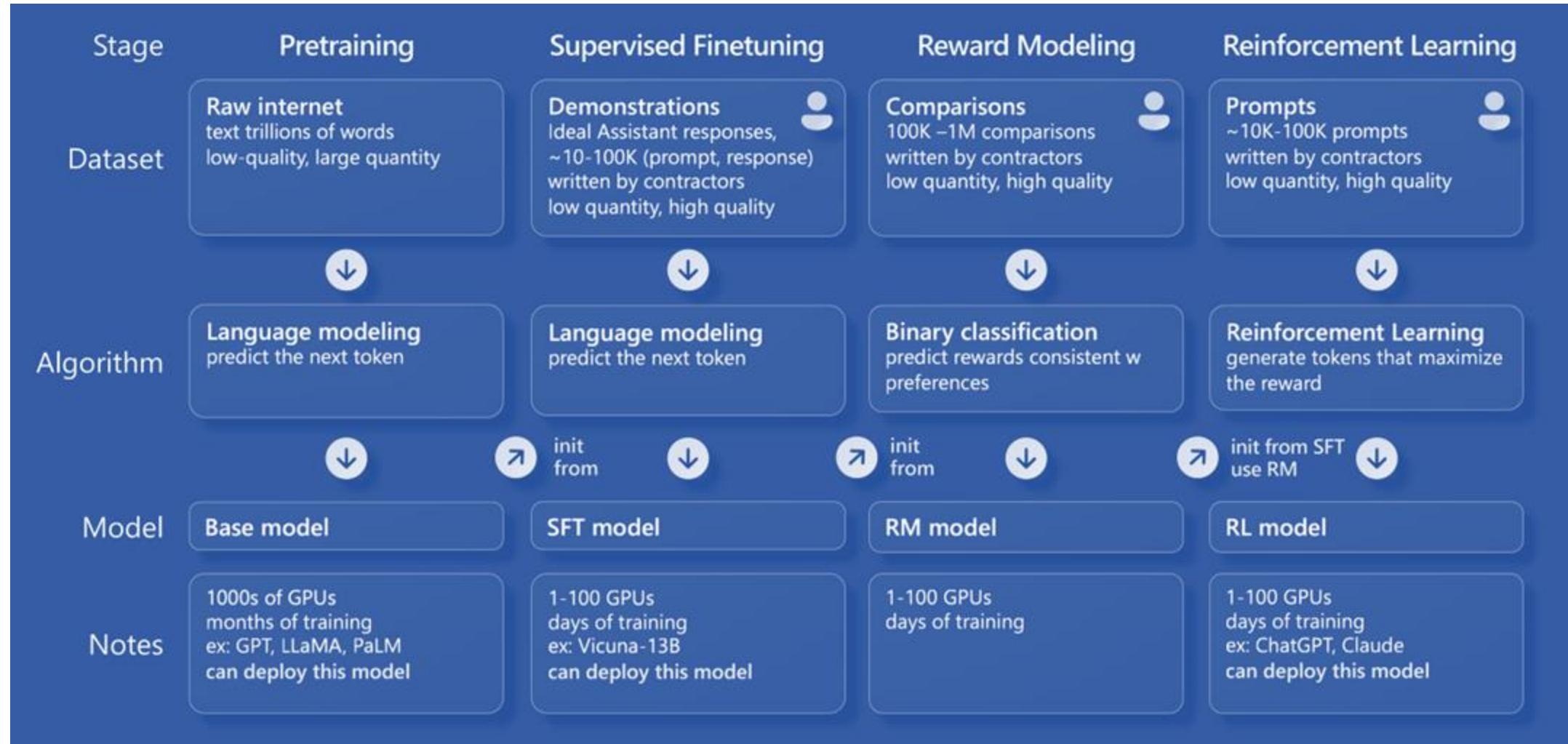
- 其他指标：BLEU等

Part

03

LLM训练技术

如何训练一个LLM



Pretraining 预训练

- 耗费极大计算资源（数千GPU训练数月）
 - 极多的数据量投喂
 - 网络爬取的大量低质量数据
 - GitHub、wiki等网站的高质量数据

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: Pre-training data. Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

GPT-3

- 1000-10000 V100 GPU上训练1月
 - 词表大小50,257
 - 上下文长度2048 token
 - 训练集大小 300B tokens

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

LLAMA

- 2048 A100 GPU上训练21天
 - 词表大小 32,000
 - 上下文长度 2048 token
 - 训练集大小 1-1.4T tokens

- **Scaling laws**: 描述模型性能 L , 参数量 N , 训练数据大小 D 与计算量 C 的关系。LLM的第一性原理

$$L = -\frac{1}{T} \sum_{t=0}^T \sum_{i=1}^{|\mathcal{D}|} \hat{y}_{t+1}^i \log p(y_{t+1}^i | w_1, w_2, \dots, w_t; \theta)$$

- L 为模型在测试集上的交叉熵损失; (越小越好)
- C 是指训练模型时的Flops, 浮点运算次数
- 每个token会涉及一次前向传播, 一次反向传播, 反向传播的Flops是前向传播的两倍
- $C \approx 6ND$
- 对于计算量 C , 模型参数量 N 和数据集大小 D , 当不受其他两个因素制约时, 模型性能 L 与每个因素都呈现幂律关系

- attention: $4Bs^2h + 8Bsh^2$
 - $Q, K, V: 6Bsh^2$
 - 就是个linear层 $h \rightarrow h$, input: $[Bs, h] * [h, h]$, 复杂度 $O(Bsh^2)$,
 - 乘法算作一次计算, 加法算作一次计算 [cite iv.], 也有说是 $2=2(fp16)*1(1次乘法)$ [cite i.] 所以 linear 的FLOPs = $2Bsh^2$
 - attention矩阵计算 ($Q*K$) : $2Bs^2h$
 - $B * ([s, h] * [h, s]) \rightarrow Bss$, 复杂度 $O(Bssh)$
 - Softmax: 忽略? 因为明显比attention矩阵少一阶?
 - Bss 次 exp 计算, 乘法计算, 加法计算
 - attention*V: $B * ([s, s] * [s, h])$ flops = $2Bs^2h$
 - Output(linear): $2Bsh^2$
- FFN: $16Bsh^2$
- 总: $B * [(4s^2h + 24sh^2) * l + 2shV]$

- 对于计算量 C ，模型参数量 N 和数据集大小 D ，当不受其他两个因素制约时，模型性能 L 与每个因素都呈现幂律关系

- 幂律关系？

$$L(x) = \left(\frac{c}{x}\right)^\alpha$$

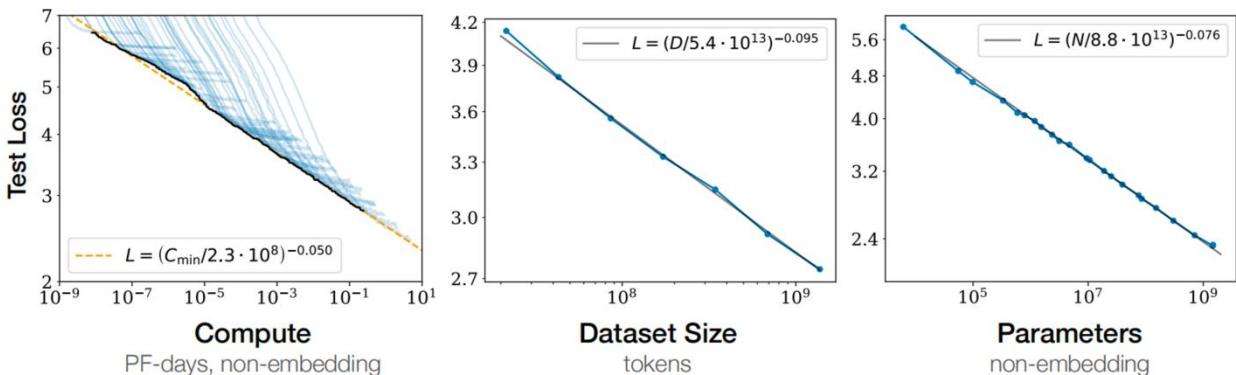
- X 是变量， c 和 α 是常数

- 一般用 10^x 替代 X 然后两边取对数转换为线性关系

$$y' = \alpha \log c - \alpha x'$$

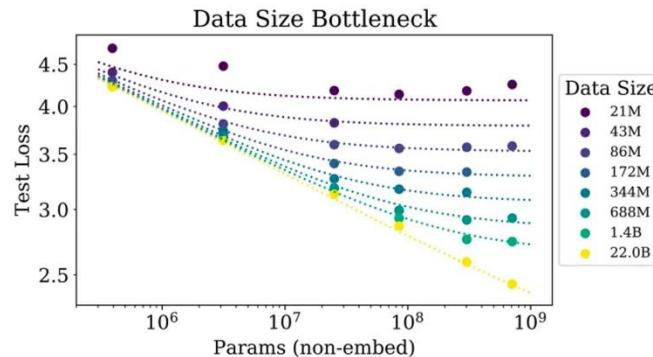
- LLM的Scaling Laws来源于实验。作用？

- 预测模型效果，便于调整训练策略和超参数
 - 计算最佳BS, Step等超参
- 合理的分配资源
- 分析大语言模型的极限：
 - 观察早期训练曲线能粗略估计后续表现

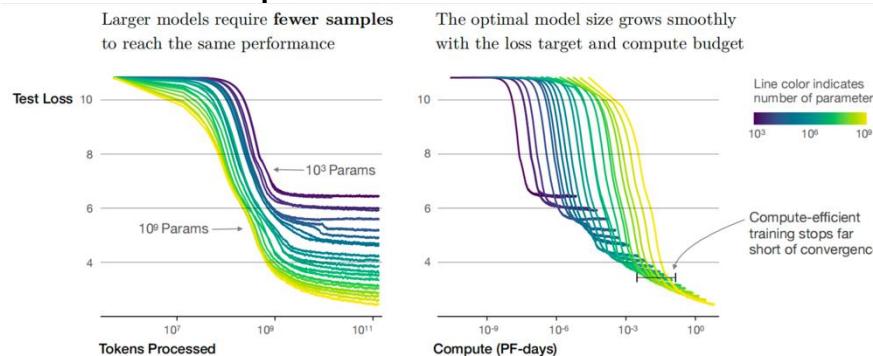


➤ 结论：

- 模型表现和规模强相关和shape弱相关
 - 数据量和参数量，固定一个，模型表现随另一个增加的收益会衰减（给出了最佳模型与数据的关系）

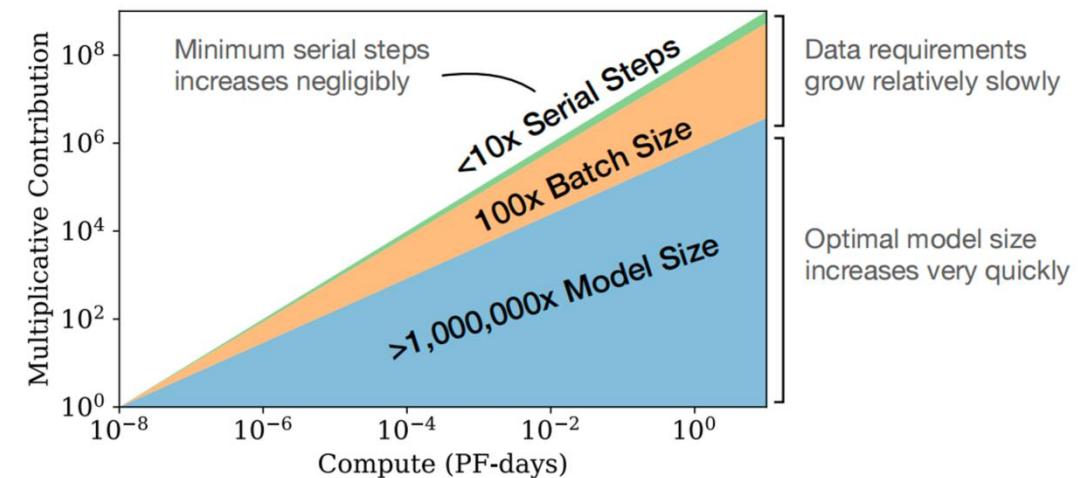


- 同样的数据集，大模型比小模型达到同等效果需要训练的step更少

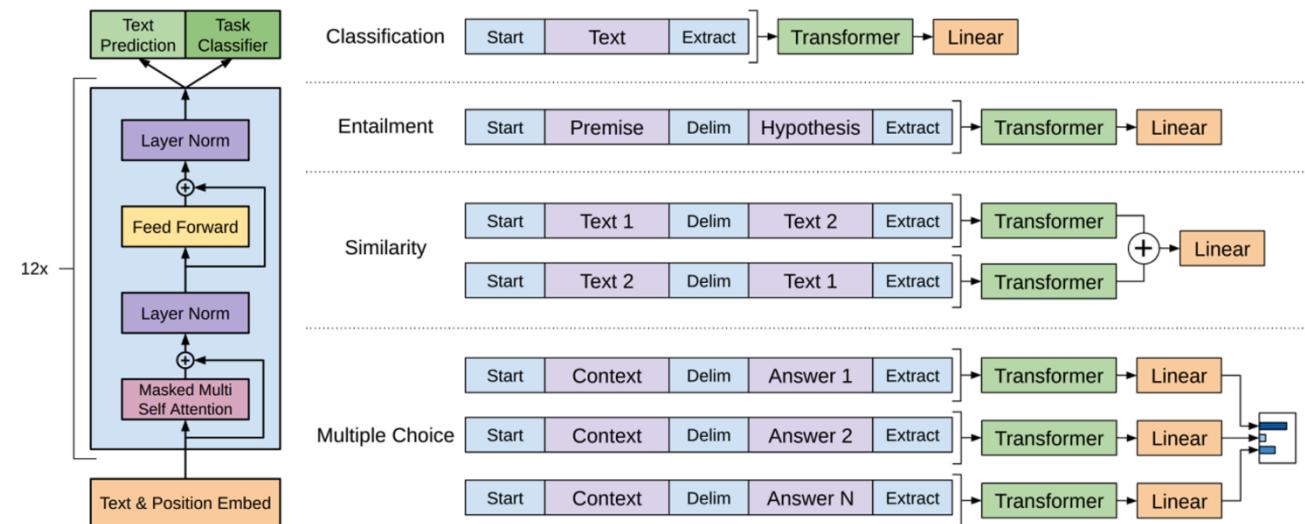


➤ 结论：

- 计算量固定，性价比最高的计算方式是训练大模型
 - 计算量增加
 - 最佳的模型参数量迅速增长
 - 最佳BS和Step增长缓慢



- 长时间大数据进行预训练后，模型学习到了通用表示
- 因此仅需微调仅可在下游任务上适用
- LLM转变了训练范式，以情感分类为例
 - 旧方法：
 - 收集一堆正面和负面的情感信息，以此训练
 - 新方法：
 - 忽略情感分类这一步，用例子微调



- Transformer 被迫在语言建模过程中同时处理大量任务，仅仅为了预测下一个标记，它就得被迫去了解很多关于文本的结构和其中的不同概念
- 通过**Prompt Engineering**，让模型误以为它其实是在完善文档，从而让它执行特定的任务
- 让模型看起来像是在完善文档
- 提供少量样本的提示
- 将要查询的问题放在最后
- 模型将完成任务，生成某种答案
- 全量微调与高效微调

Context (passage and previous question/answer pairs)

Tom goes everywhere with Catherine Green, a 54-year-old secretary. He moves around her office at work and goes shopping with her. "Most people don't seem to mind Tom," says Catherine, who thinks he is wonderful. "He's my fourth child," she says. She may think of him and treat him that way as her son. He moves around buying his food, paying his health bills and his taxes, but in fact Tom is a dog.

Catherine and Tom live in Sweden, a country where everyone is expected to lead an orderly life according to rules laid down by the government, which also provides a high level of care for its people. This level of care costs money.

People in Sweden pay taxes on everything, so aren't surprised to find that owning a dog means more taxes. Some people are paying as much as 500 Swedish kronor in taxes a year for the right to keep their dog, which is spent by the government on dog hospitals and sometimes medical treatment for a dog that falls ill. However, most such treatment is expensive, so owners often decide to offer health and even life .. for their dog.

In Sweden dog owners must pay for any damage their dog does. A Swedish Kennel Club official explains what this means: if your dog runs out on the road and gets hit by a passing car, you, as the owner, have to pay for any damage done to the car, even if your dog has been killed in the accident.

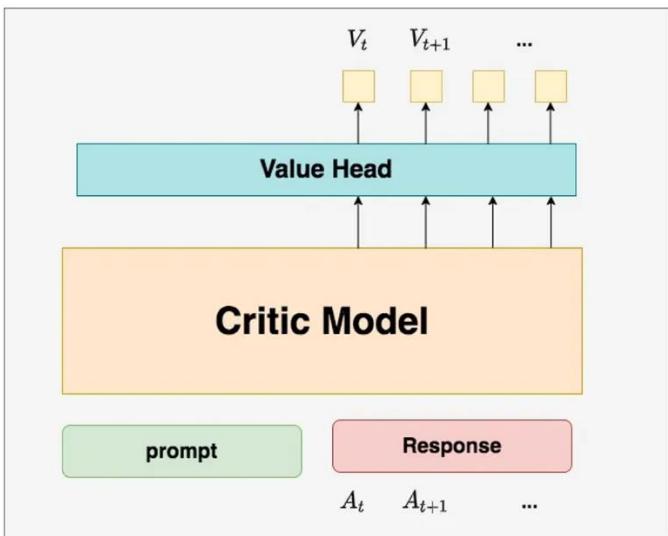
Q: How old is Catherine?
A: 54

Q: where does she live?
A:

GPT-2 is "tricked" into performing a task by completing the document

- 使得LLM对齐人类偏好
- Reward Model训练
 - 使用SFT后的LLM对问题生成4到9个不同回答
 - 标注人员对其进行打分排序
 - 训练一个较小的Reward Model
 - 输出对应的 V_t 的值
 - Pair wiss loss

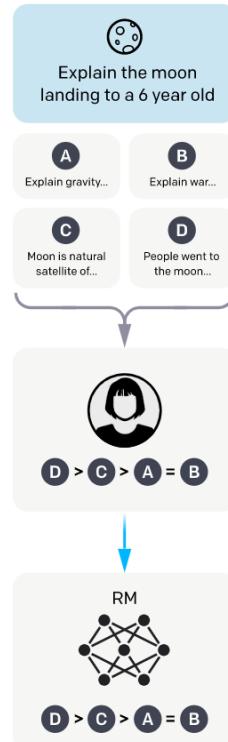
$$L_r(\theta) = -\frac{1}{N} \sum_1^N \log(\text{sigmoid}(r_\theta(\text{query}, \text{chosen}) - r_\theta(\text{query}, \text{rejected})))$$



Step 2

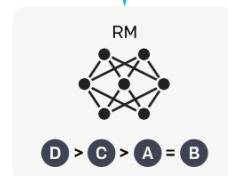
Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.

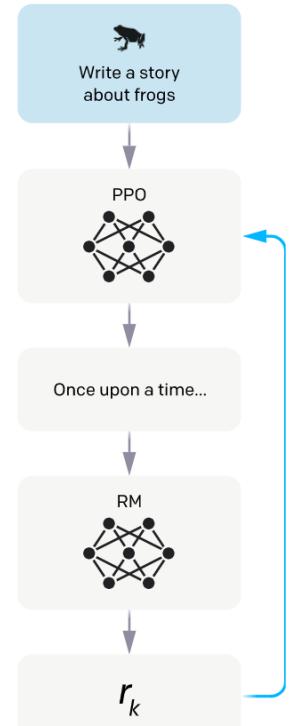
This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using reinforcement learning.

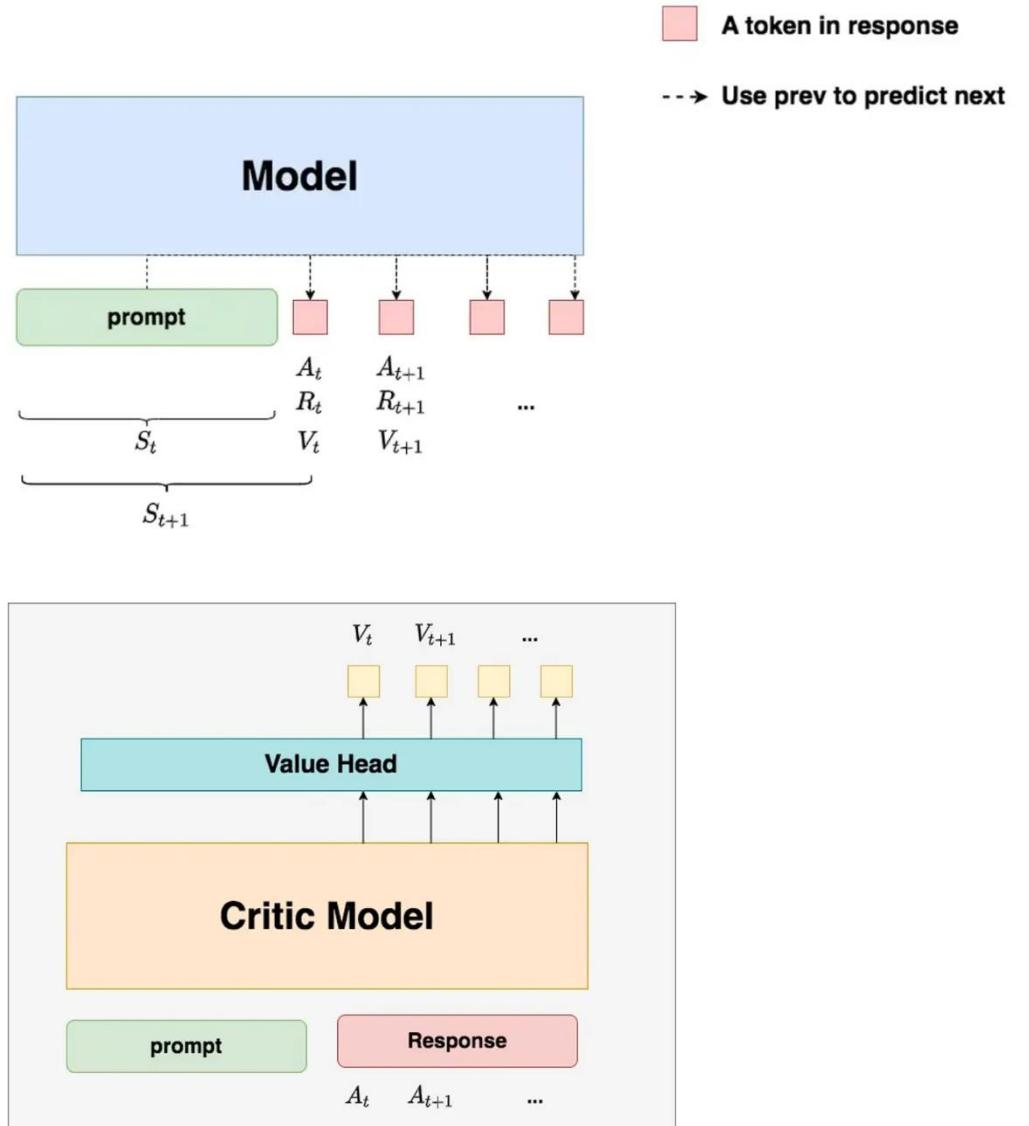
A new prompt is sampled from the dataset.



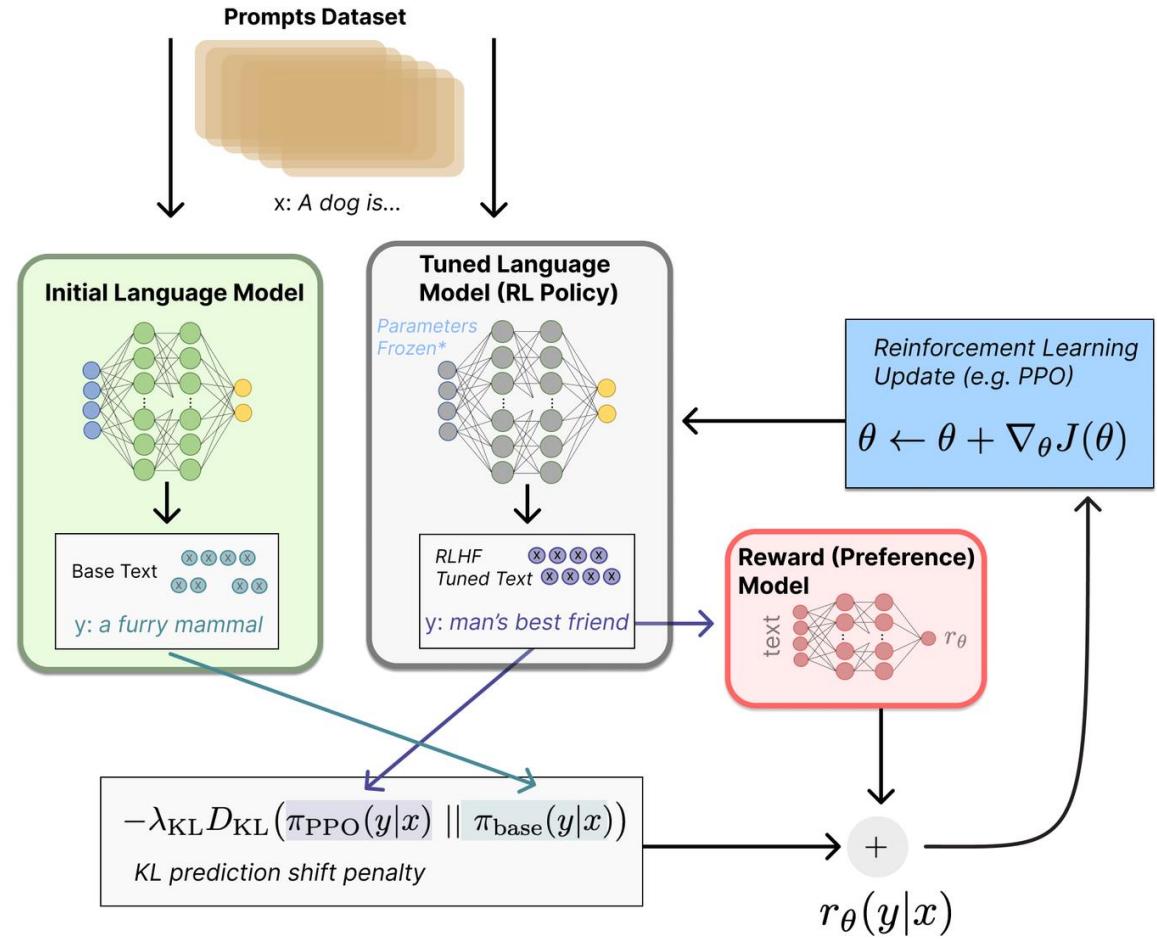
➤ RHLF语境下的强化学习

- **Action**: 在 t 时刻, 模型根据上文, 产出一个 token, 动作空间即为词表
- **State**: 当前生成的token串, 状态空间即为所有的token组合
- **Agent (policy)** : 大模型本身
- 轨迹 τ : 一个完整的response生成
- Value: RM
- **Ref model**: SFT模型, 用来辅助Actor的优化, 保证其不偏离SFT模型
- **Actor model**: 目标策略模型, 使用SFT初始化

RL in NLP

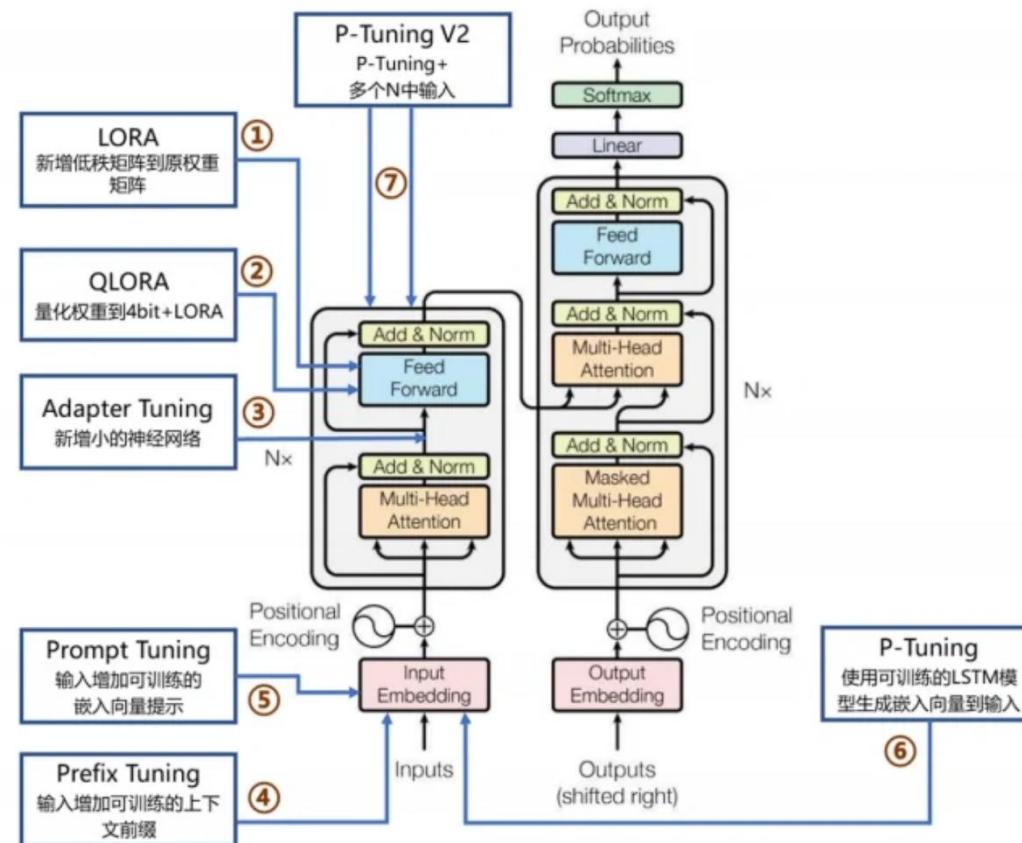


- 近端策略优化框架PPO
- 加载四个模型：
 - actor model, ref_model (SFT LLM)
 - reward model, critic model (RM)
- Actor根据prompt生成一个answer
- actor model和不更新参数的ref_model计算KL散度计入loss
 - 约束actor model的输出分布
- Reward Model 根据answer输出整个句子的得分
- Critic Model输出逐位置 V
- 根据PPO更新Actor及Critic



PEFT (Parameter-Efficient FineTuning)

- **全量微调**: 在预训练大模型的基础上调整所有层和参数，需要的计算资源大
- **高效微调PEFT**: 最小化微调参数的数量和计算复杂度，来提高预训练模型在新任务上的性能，从而缓解大型预训练模型的训练成本。



LoRA: Low-Rank Adaptation

- Insight: 模型是过参数化的，它们有更小的内在维度，模型主要依赖于这个低的内在维度 (low intrinsic dimension) 去做任务适配。

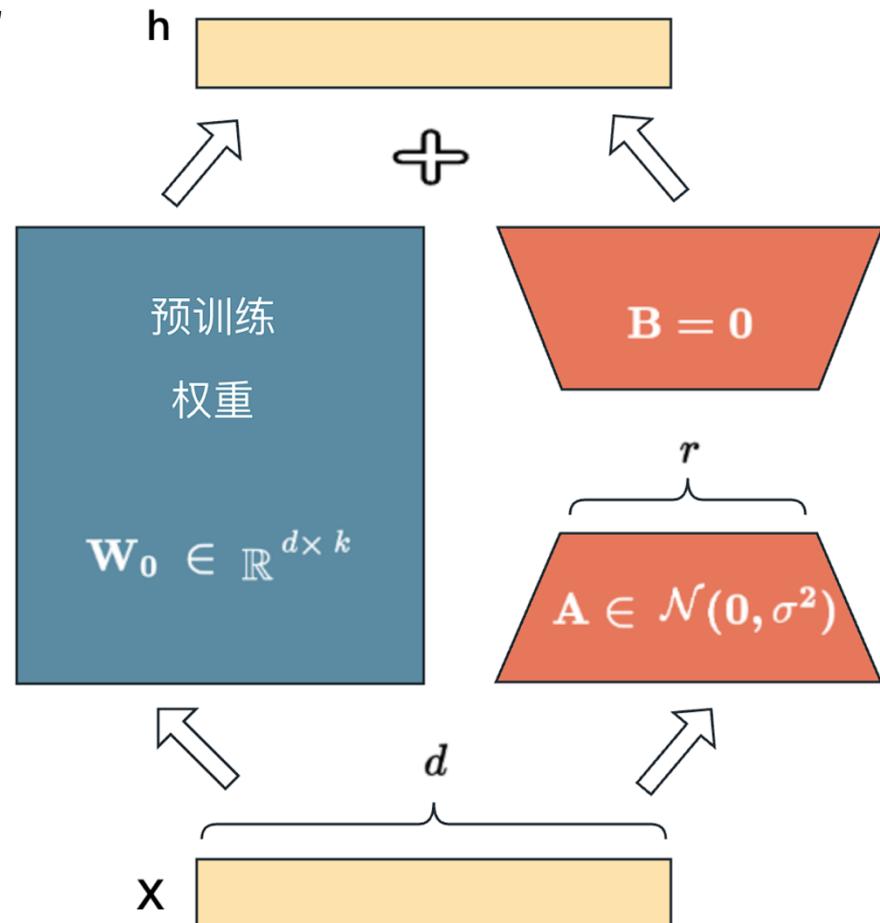
- 微调时，将参数的更新分离：

$$W_0 + \Delta W$$

- 冻结原本的权重矩阵，增加两个低秩的可学习参数矩阵（降维矩阵A和升维矩阵B）：

$$W_0 + \Delta W = W_0 + BA, B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$$

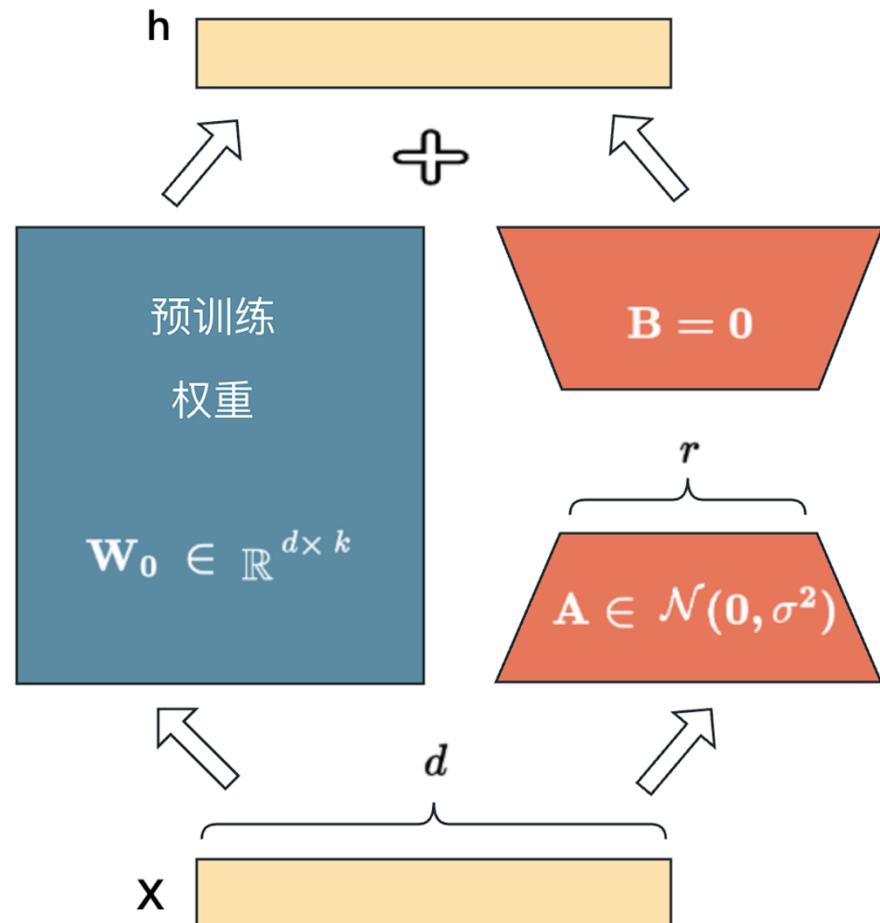
- r 一般远小于hidden size，取8或16
- 推理时两个权重可以直接合并——不带来任何额外的Inference Latency
- 用高斯分布初始化矩阵A，B矩阵初始化为0
- 缺点：通用性损害，需要精调数据集，或者应用其他魔法手段（自蒸馏等）
- 额外参数alpha: 权重系数
 - $\frac{\alpha}{r}$ rescale ΔW
 - Why?



LoRA: Low-Rank Adaptation

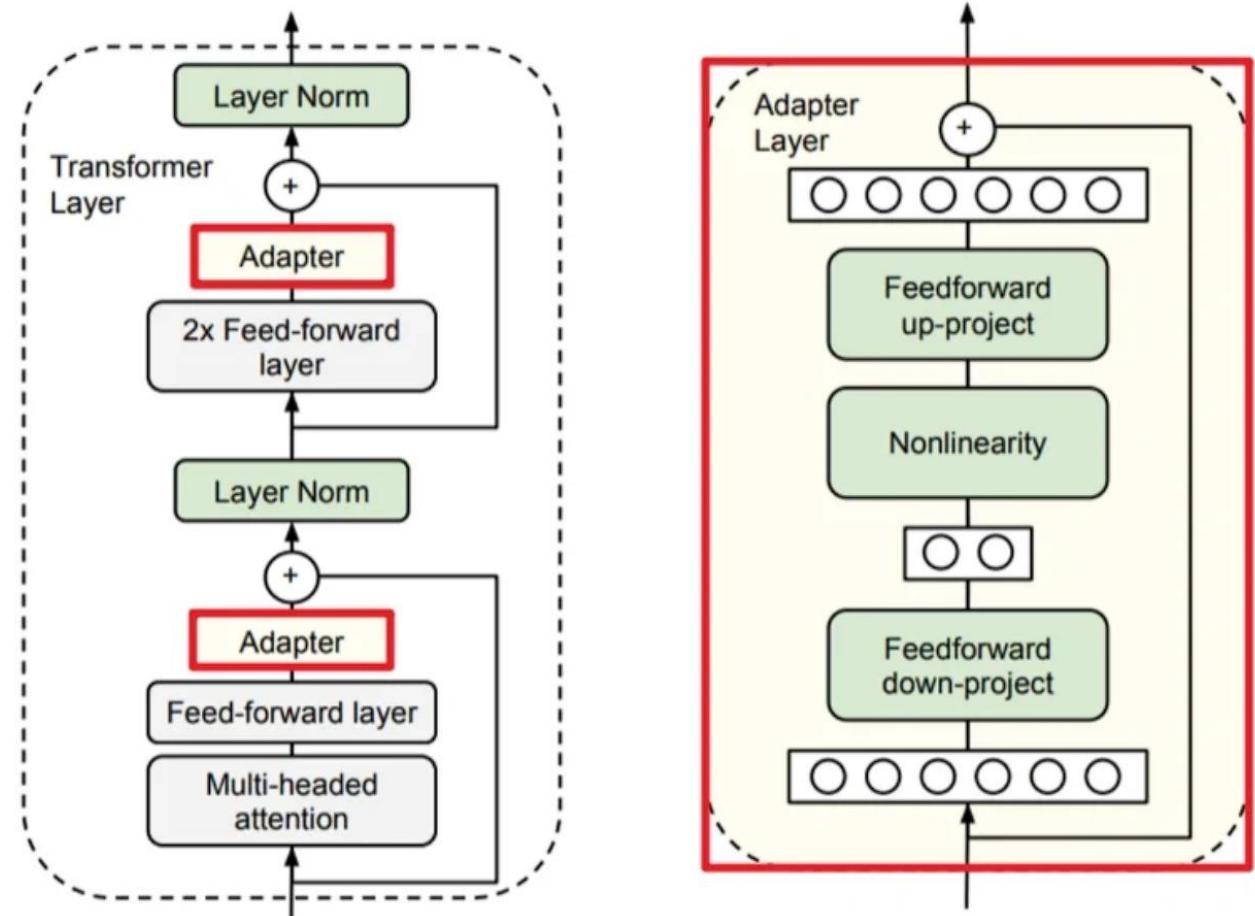
➤ 多个解释

- 当 r 较小时， 提取的是信息含量最丰富的维度，此时信息精炼，但不全面；当 r 较大时，低秩近似越逼近，此时信息更加全面，但带来的噪声也越多；通过rescale，来调整 forward 过程中新知识对模型的影响
- $1/r$ 只作为A的初始化scale， α 单独作为scale系数
 - 一般和 r 等比例放大缩小
 - 把rank增大带来的数值增大消除掉
 - A^*B 中的加法次数和rank线性相关
 - 通常 α 被设为 1，这样模型的初始行为几乎不会被破坏。一些实现中会使用 $\alpha>1$ （例如 16 或 32），以增加低秩调整的影响力。



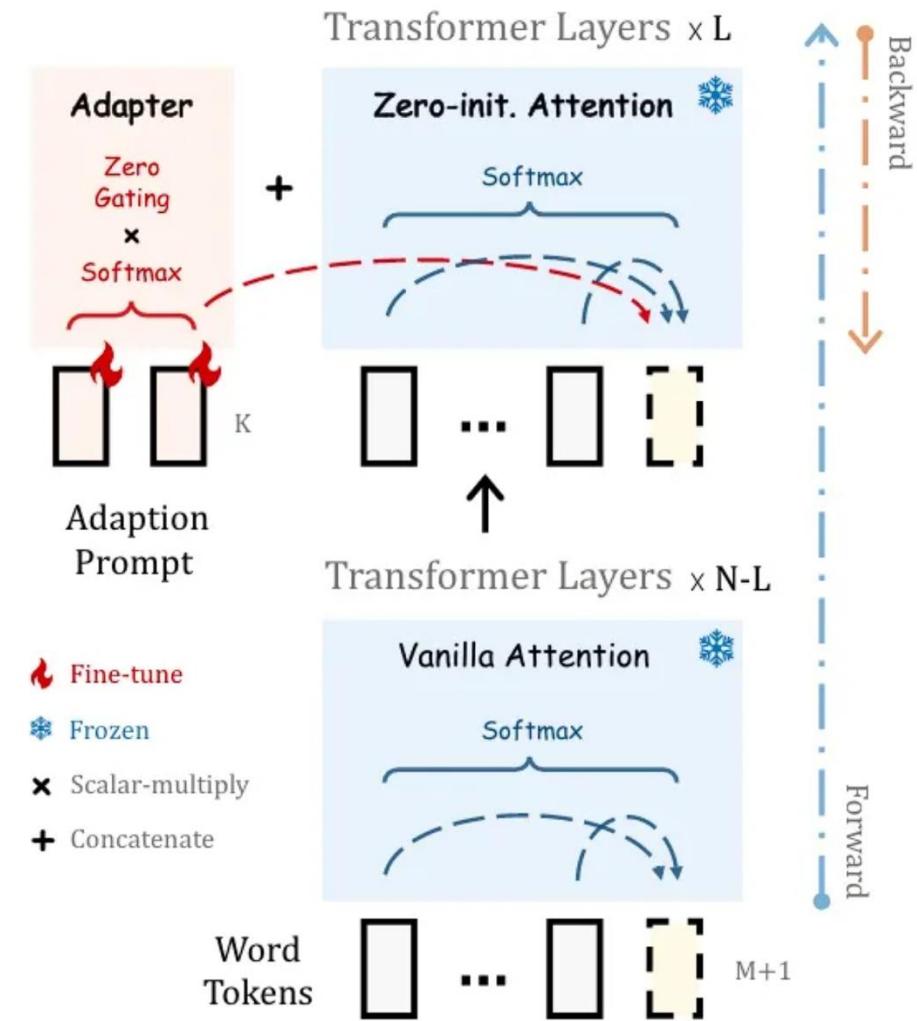
Adapter-Tuning

- 直接在各模块后面添加一个由FFN构成的Adapter



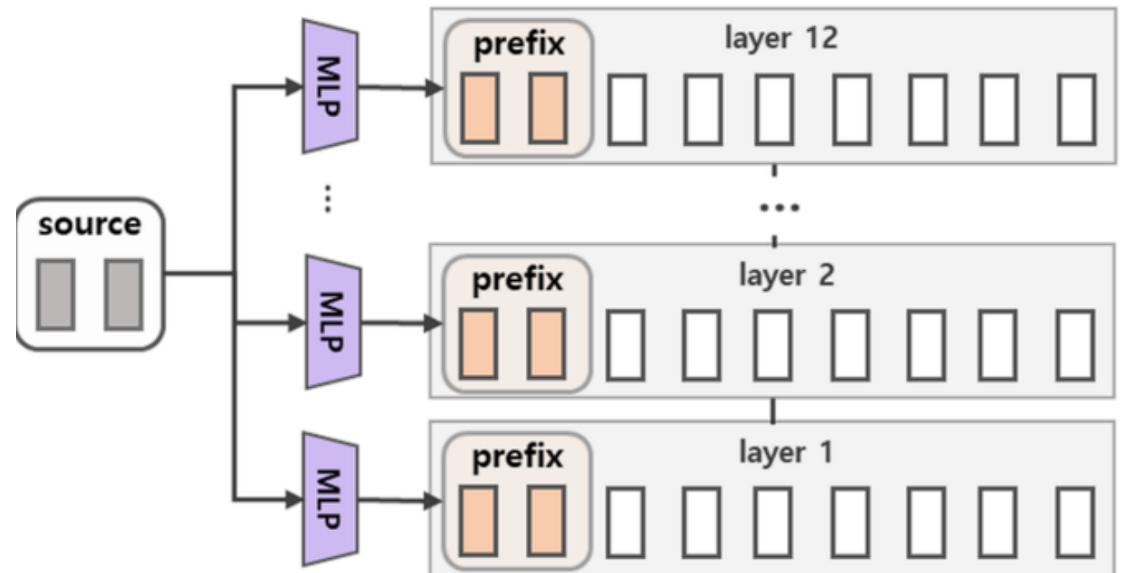
LLAMA-Adapter

- 冻结LLM主干
- 在最后L层初始化额外prompt
 - 就是learnable embedding
- Adapter fusion 模块Gate稳定训练
 - Output: 原本该TRM的attention模块输出
 - Q: 原本该TRM的Query
 - K, V: learnable embedding
 - Output= Output + Atten(Q,K,V)*Gate
 - Output继续做后续的FFN
 - Gate初始化为0
- 采用了一套可学习的适应性embedding，并将其预置在较高转换层的输入文本标记上。然后，提出了一个具有零门控的零启动注意力机制，该机制可以自适应地将新的教学线索注入到LLaMA中，同时有效地保留了其预训练的知识



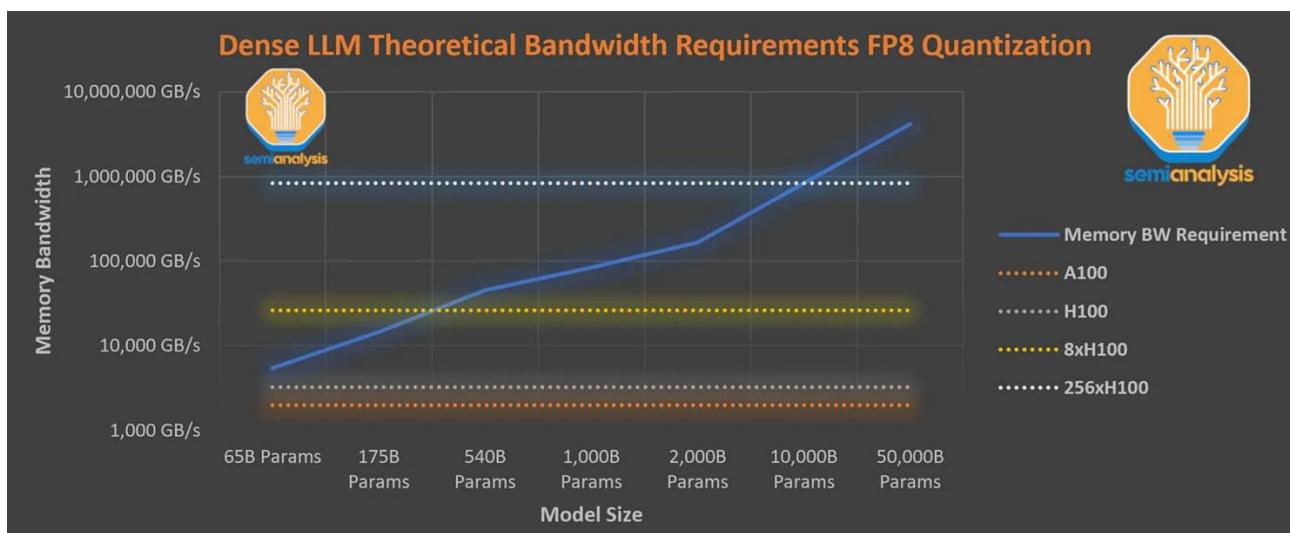
Prompt Tuning & Prefix Tuning

- 通常手工调整prompt以应对不同任务 (hard prompt tuning)
- 把prompt换成可学习参数参与优化，用固定的token代替prompt (soft prompt tuning)
 - 生成任务特定的prompt

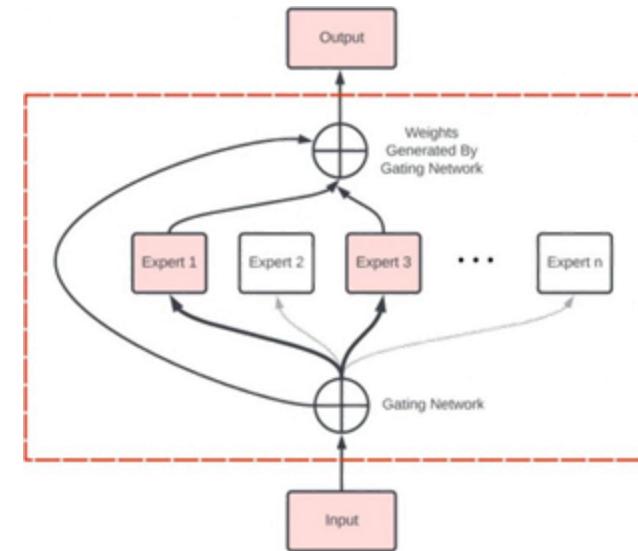


MoE

- 神经网络的稀疏性：Transformer 模型中 FFNs 稀疏激活问题
 - 神经元的多语义性：被多个语义同时激活带来矛盾的学习曲线
 - 计算资源的有限性



即使是 8x H100 也不足以以每秒 33.33 个 token 的速度服务于 1 万亿参数的密集模型



MoE架构：仅调用专家模型子集实现高效推理

Mixture-of-Experts

MoE

- 稀疏MoE层：替代FFN
- 门控网络\路由网络：决定Token被发送到哪

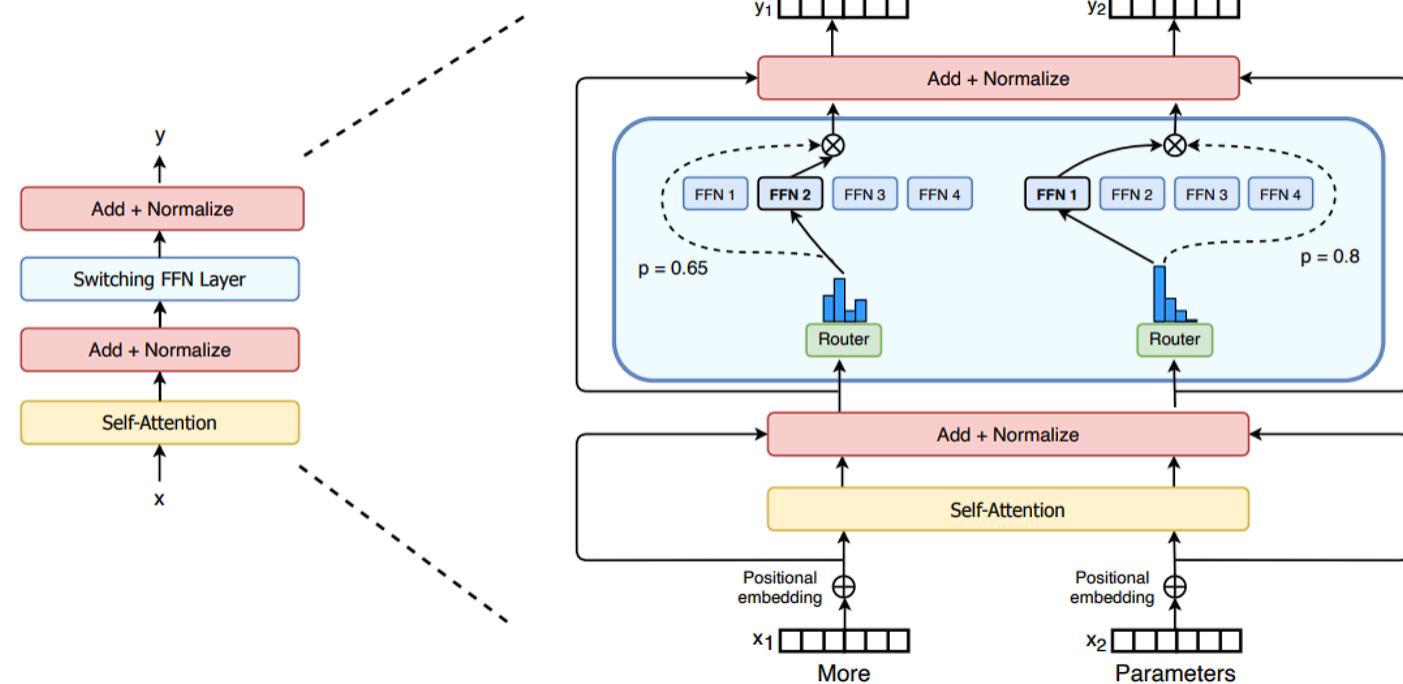
$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

MoE实现过程中的负载均衡问题：

- 修改成Top-K门控
- 随机路由
- 引入辅助loss
- 设定专家容量

MoE加速：专家并行

- 不同专家被放置在不同节点
- 其余权重所有节点共享



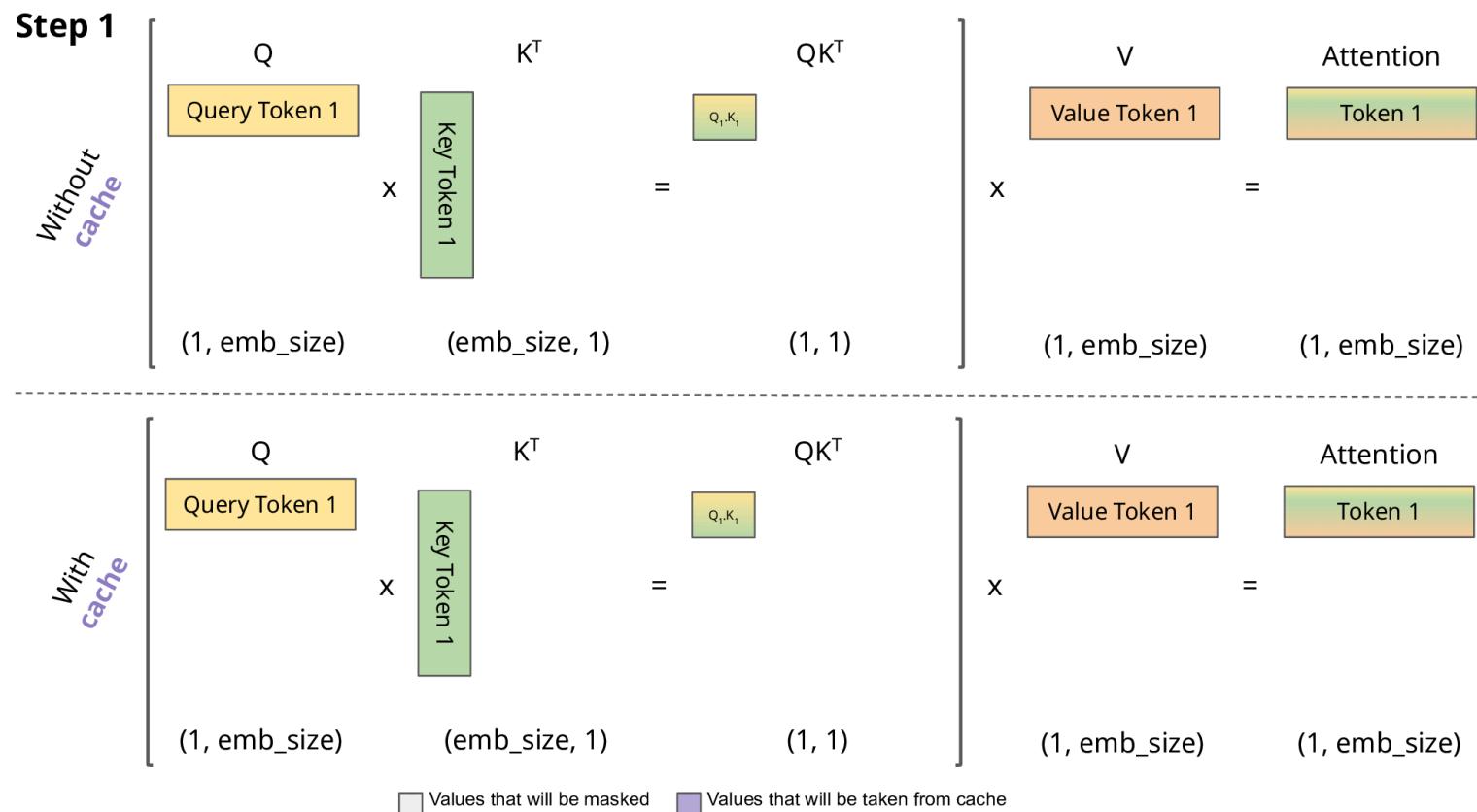
Part

05

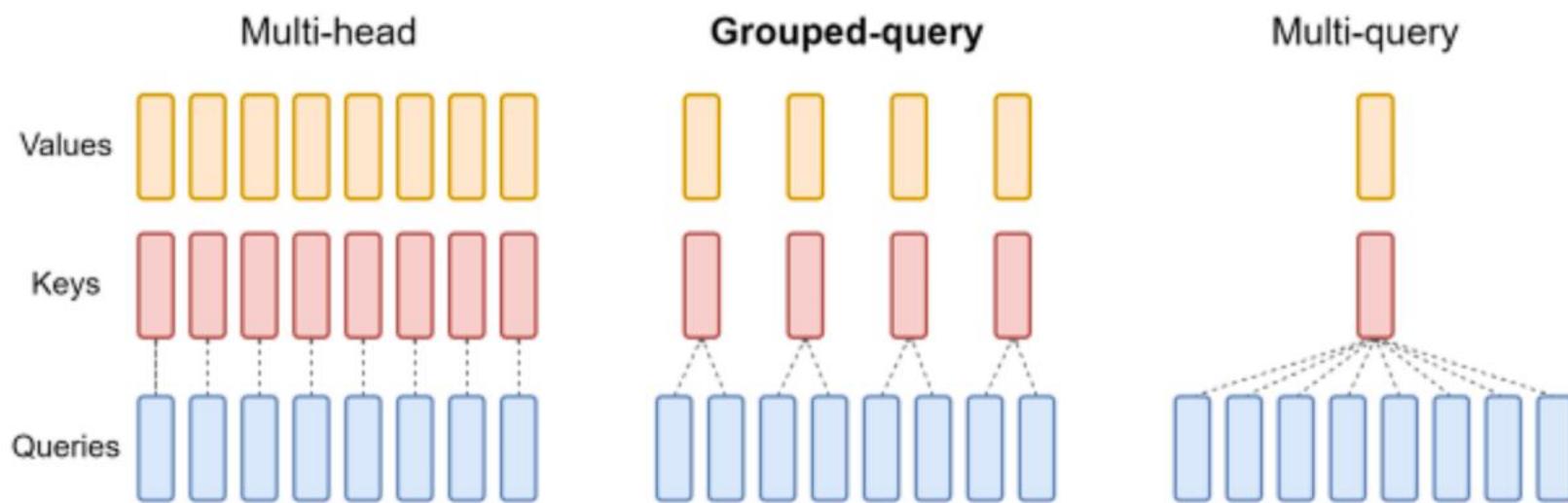
LLM推理/训练优化技术

每步推理都将前文整句输入模型是一种效率低下的方式，原因是存在相同结果的重复推理。令前一次待推理的文本长度为S，下一次为S+1，由于网络中的各项参数已经固定，因此两次推理对于前S个token的计算结果是完全相同的，包括Embedding映射，每一层、每一个注意力头下的KQV映射，注意力权重，以及后续的FFN层都在重复计算。

- 在自回归解码过程中缓存之前的K和V：
空间换时间
- 存在问题：算算要多少空间？
 - Hidden_size = 4096,
 - TRM=32个
 - 需要
 $4096 * 2\text{byte} * 2(K, V) * 32 = 512\text{KB}$
 - 考虑句子长度1024Token
 - 512MB, A100的缓存放不下
- 啥办？



- MQA: Multi Query Attention
 - 多个Head共享K, V权重
- GQA: Grouped Query Attention
 - 将Head分组，每组共享一个KV



Flash Attention

- Transformer模型实际复杂度是 N^2 ，希望优化其复杂度以逼近 $O(N)$
- Flash Attention
 - **Fast with IO**: 不减少总运算量，因为瓶颈在IO
 - **省显存**: 传统Attention需要 N^2 存储注意力矩阵，FlashAttention使其降至 $O(N)$
 - **精确**: 不以近似计算代替，输出完全等同于原本的Attention
- 传统Attention:
 - 从显存中取QK计算->将结果S写回显存->从显存读S计算softmax->将结果P写回显存->从显存读取P和V进行计算->将结果O写回显存。
- FlashAttention: **分块计算**，把一些中间矩阵（如S和P）的计算全都放在L1缓存上，从而避免多次显存读写实现了加速。

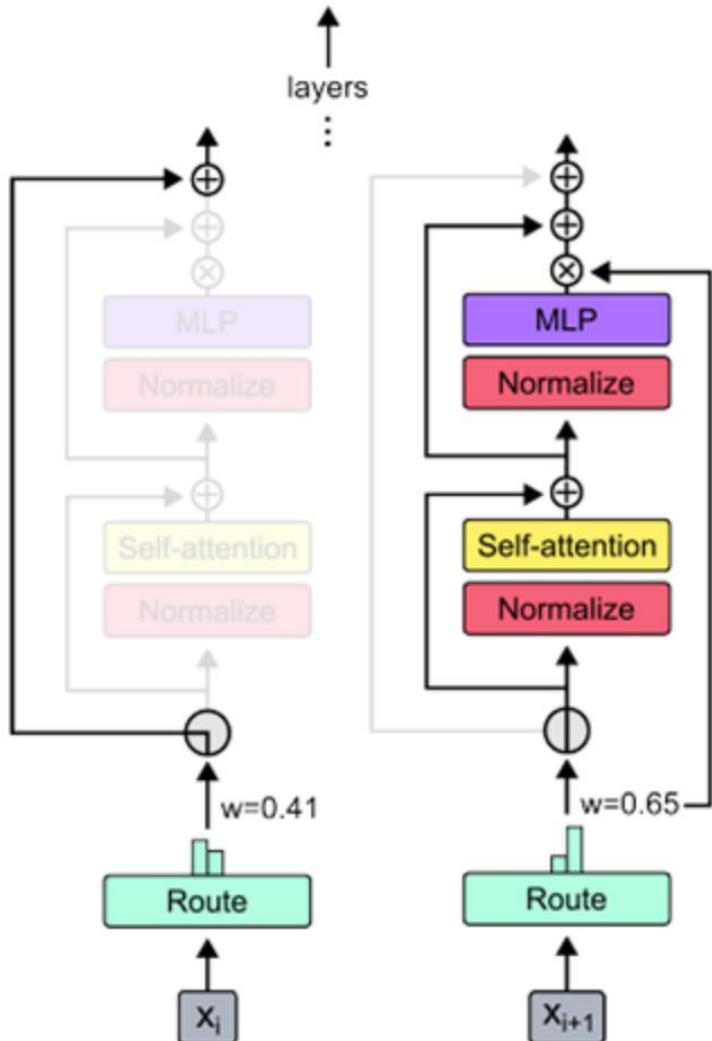
Algorithm 1 FLASHATTENTION

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM, on-chip SRAM of size M .

- 1: Set block sizes $B_c = \lceil \frac{M}{4d} \rceil$, $B_r = \min(\lceil \frac{M}{4d} \rceil, d)$.
- 2: Initialize $\mathbf{O} = (0)_{N \times d} \in \mathbb{R}^{N \times d}$, $\ell = (0)_N \in \mathbb{R}^N$, $m = (-\infty)_N \in \mathbb{R}^N$ in HBM.
- 3: Divide \mathbf{Q} into $T_r = \lceil \frac{N}{B_r} \rceil$ blocks $\mathbf{Q}_1, \dots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide \mathbf{K}, \mathbf{V} in to $T_c = \lceil \frac{N}{B_c} \rceil$ blocks $\mathbf{K}_1, \dots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \dots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.
- 4: Divide \mathbf{O} into T_r blocks $\mathbf{O}_1, \dots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, divide ℓ into T_r blocks $\ell_1, \dots, \ell_{T_r}$ of size B_r each, divide m into T_r blocks m_1, \dots, m_{T_r} of size B_r each.
- 5: **for** $1 \leq j \leq T_c$ **do**
- 6: Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.
- 7: **for** $1 \leq i \leq T_r$ **do**
- 8: Load $\mathbf{Q}_i, \mathbf{O}_i, \ell_i, m_i$ from HBM to on-chip SRAM.
- 9: On chip, compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.
- 10: On chip, compute $\tilde{m}_{ij} = \text{rowmax}(\mathbf{S}_{ij}) \in \mathbb{R}^{B_r}$, $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij} - \tilde{m}_{ij}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\tilde{\ell}_{ij} = \text{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^{B_r}$.
- 11: On chip, compute $m_i^{\text{new}} = \max(m_i, \tilde{m}_{ij}) \in \mathbb{R}^{B_r}$, $\ell_i^{\text{new}} = e^{m_i - m_i^{\text{new}}} \ell_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$.
- 12: Write $\mathbf{O}_i \leftarrow \text{diag}(\ell_i^{\text{new}})^{-1} (\text{diag}(\ell_i) e^{m_i - m_i^{\text{new}}} \mathbf{O}_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\mathbf{P}}_{ij} \mathbf{V}_j)$ to HBM.
- 13: Write $\ell_i \leftarrow \ell_i^{\text{new}}$, $m_i \leftarrow m_i^{\text{new}}$ to HBM.
- 14: **end for**
- 15: **end for**
- 16: Return \mathbf{O} .

Mixture-of-Depths

- 不是所有问题都需要消耗全部的脑力思考，LLM亦然
- 使得LLM动态学习是否要将某个Token通过当前Transformer Layer。还是直接通过残差连接绕过
- 训练一个简单的路由网络来执行这个分类
- 让每个分支选择Top-K个Token确保了负载均衡
- 引入辅助loss和MLP解决自回归解码的问题
 - 解码时可不知道当前token是不是topk之内的，毕竟后面还有token没有解码



- LLM运行缓慢的根本问题是自回归解码
- 解码过程中只利用了最后一个Token的分布，浪费！
- 利用一个小得多的辅助模型迅速多次串行解码，LLM一次forward用来判别多个Token；
- 显著加速，被GPT所采用

[START] japan ' s benchmark bond n

[START] japan ' s benchmark nikkei 22 75

[START] japan ' s benchmark nikkei 225 index rose 22 76

[START] japan ' s benchmark nikkei 225 index rose 226 : 69 7 points

[START] japan ' s benchmark nikkei 225 index rose 226 : 69 points , or 0 1

[START] japan ' s benchmark nikkei 225 index rose 226 : 69 points , or 1 : 5 percent , to 10 , 9859

[START] japan ' s benchmark nikkei 225 index rose 226 : 69 points , or 1 . 5 percent , to 10 , 989 :

[START] japan ' s benchmark nikkei 225 index rose 226 : 69 points , or 1 . 5 percent , to 10 , 989 :

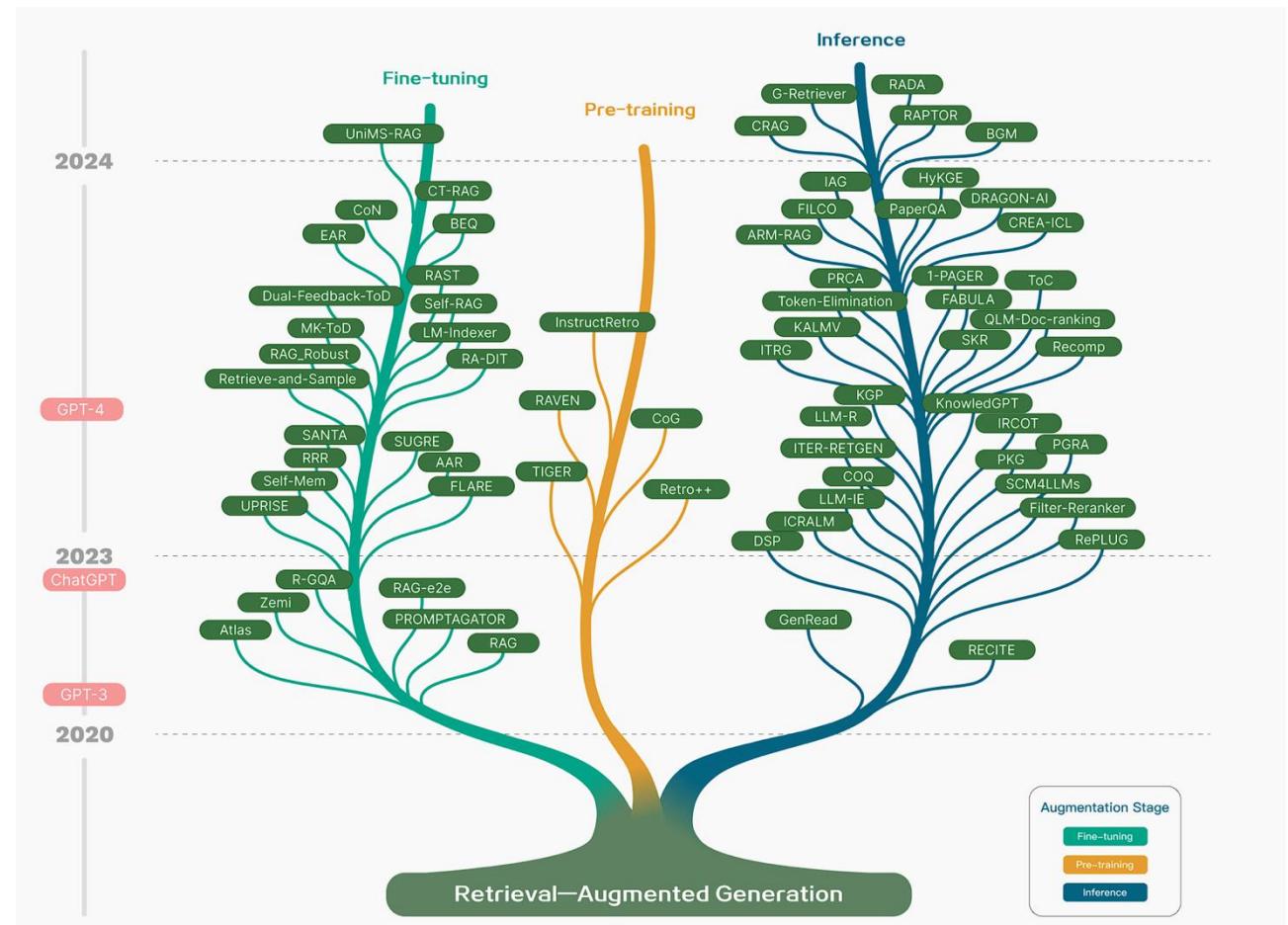
[START] japan ' s benchmark nikkei 225 index rose 226 : 69 points , or 1 . 5 percent , to 10 , 989 :

生成与验证步骤

- (1) 辅助模型生成候选序列
- 使用辅助模型生成多个候选 token 序列，或者单步生成多个候选 token (称为 speculative tokens)。
- (2) 主模型验证候选序列
- 主模型仅需验证这些候选 token 是否符合其概率分布。
- (3) Token 接收与拒绝
 - 接受：如果主模型验证通过，则这些 token 会被直接作为生成结果。
 - 拒绝：如果辅助模型生成的 token 不符合主模型的分布，则主模型重新生成这些 token。
- (4) 继续迭代
 - 辅助模型继续生成新的 token，主模型验证，直到生成完成。

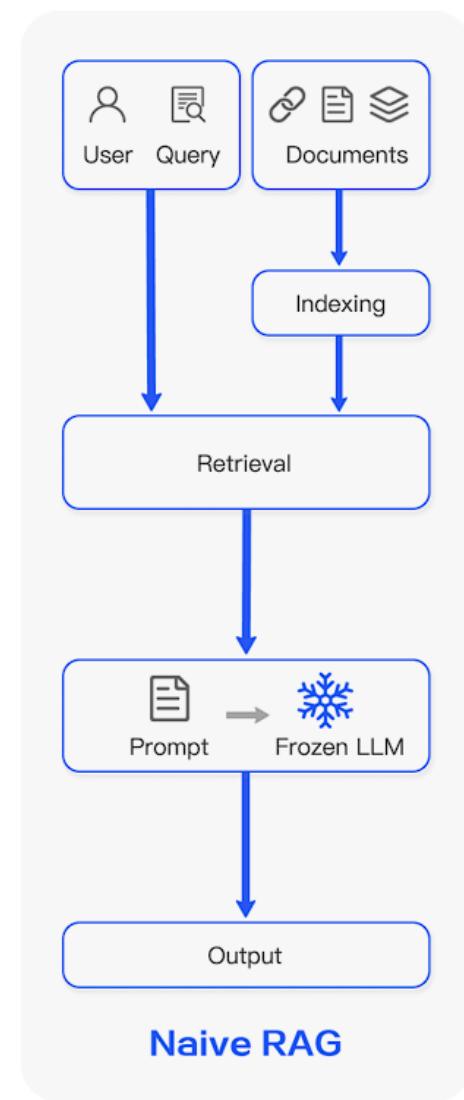
RAG: Retrieval Augmented Generation

- RAG: 检索增强生成，目前最火的LLM应用方案
 - 检索技术 + LLM 提示
- 单纯的LLM:
 - 知识的局限性
 - 幻觉问题
 - 数据安全性
- RAG三个关键阶段:
 - 将目标文档整理成知识库
 - 利用编码模型基于问题在知识库中检索相关文档
 - 使用检索到的上下文作为条件，系统生成文本
- RAG vs Fine-tuning
 - 互不排斥，相互补充



RAG: Retrieval Augmented Generation

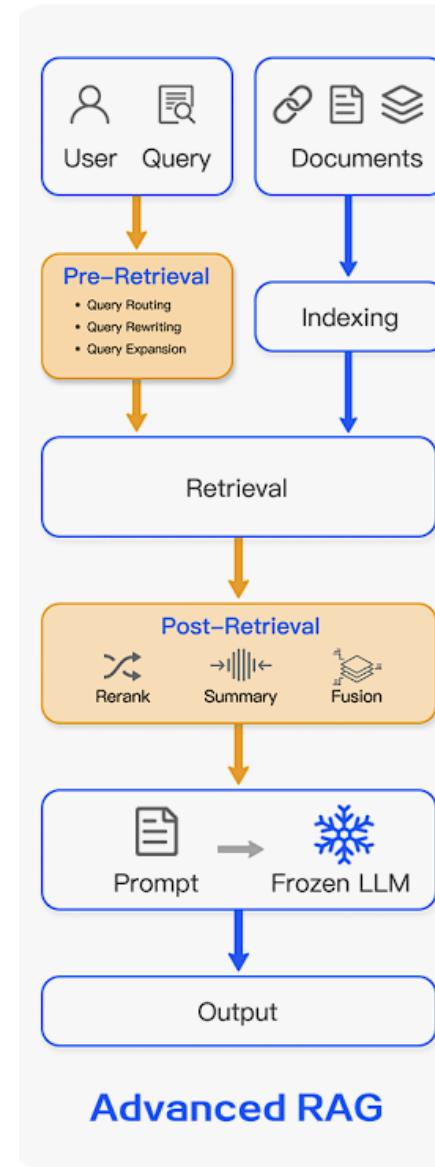
- 朴素RAG：创建索引、检索和生成
 - 数据进行分块并且编码成向量
 - 用户查询和数据向量计算相似度取topK
 - 组合成新prompt生成
- 缺点：
 - 检索质量低：不是检索的所有的内容都有用
 - 没有检索到时生成内容不可控
 - 难以处理段落之间的一致性，多个段落可能包含相似内容



RAG: Retrieval Augmented Generation

➤ Advanced RAG:

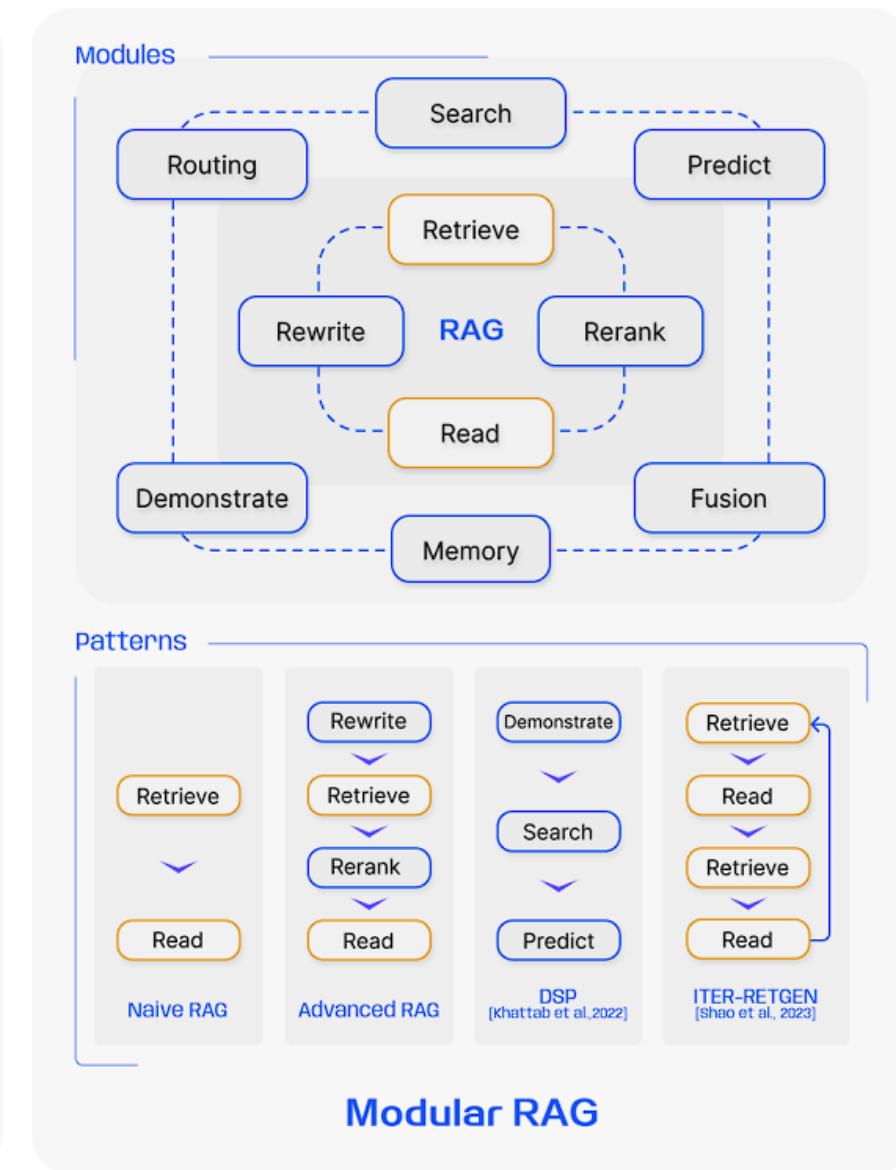
- 检索前处理:
 - 优化数据索引: 增强数据粒度、优化索引结构、添加元数据信息、对齐优化
- Embedding:
 - 微调嵌入: 增强检索内容和查询之间的相关性
 - 动态嵌入: 根据上下文进行调整
- 检索后处理
 - 重排序: 向量相似度不可靠, 得到子集后, 再用LLM询问进行重排序
 - prompt压缩: 减少上下文长度, 减少文档噪声
- Pipeline优化:
 - 探索性混合搜索
 - 递归搜索
 - 子查询



RAG: Retrieval Augmented Generation

➤ 模块化 RAG:

- 集成了多种方法来扩展功能模块
- 搜索模块
- 内存模块
- 额外生成模块
- 对齐模块
- 验证模块





北京航空航天大學
BEIHANG UNIVERSITY

人工智能研究院
Institute of Artificial Intelligence

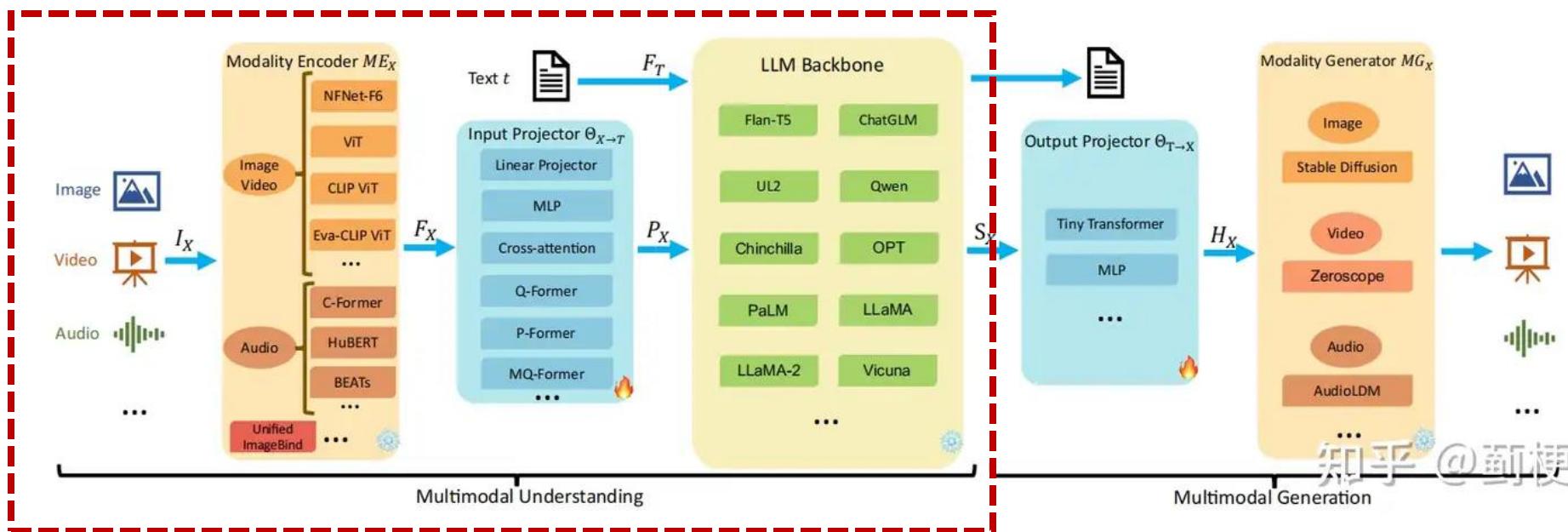
多模态大语言模型：基础原理与前沿应用

阮受炜

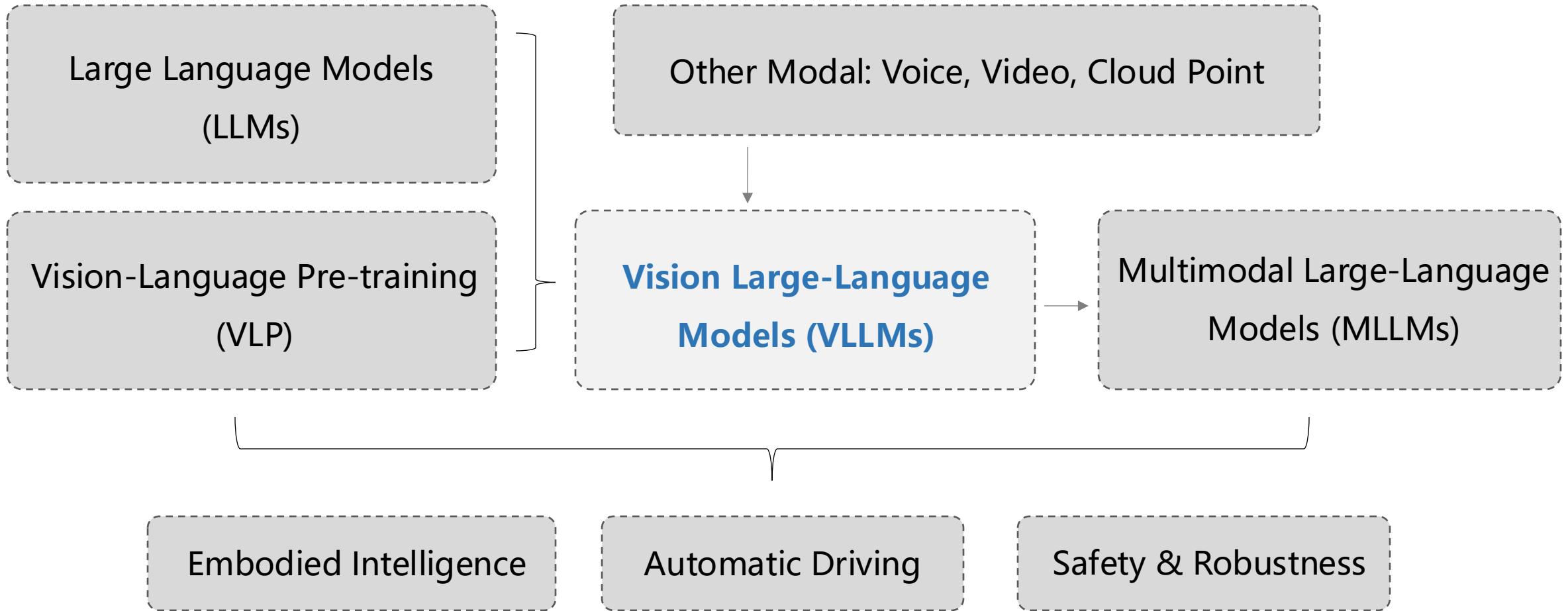
2024年5月

Why & What is MLLM

- 自然语言是人类对生活环境中不同模态信息（视觉、音频、动作）的一种高度抽象，能够将复杂的概念和信息以人类可理解的形式呈现
- 多模态大语言模型（MLLMs）本质上是一种结合了多种感知模式（如图像、声音等）的大规模语言模型
- 从输入输出形式而言，多模态大语言模型以文本、图像等不同模态信息作为输入，并使用文本模态作为输出



A RoadMap



Contents

1 预备知识

1.1 大语言模型

2 视觉-语言预训练 (VLP)

2.1 Vision-Language (VL) 发展历史

2.2 多模态任务 / 下游任务

2.3 训练目标及模型架构总结

2.4 早期代表性工作

2.5 编码器架构：CLIP及相关改进工作

2.6 编码-解码器架构：BLIP

3 视觉-大语言模型 (VLLMs)

3.1 BLIP-2

3.2 MiniGPT4V

3.3 视觉指令微调：LLaVa系列工作

4 多模态大语言模型 (VLLMs) 及应用

4.1 3D模态：LION

5 多模态大语言模型的安全研究

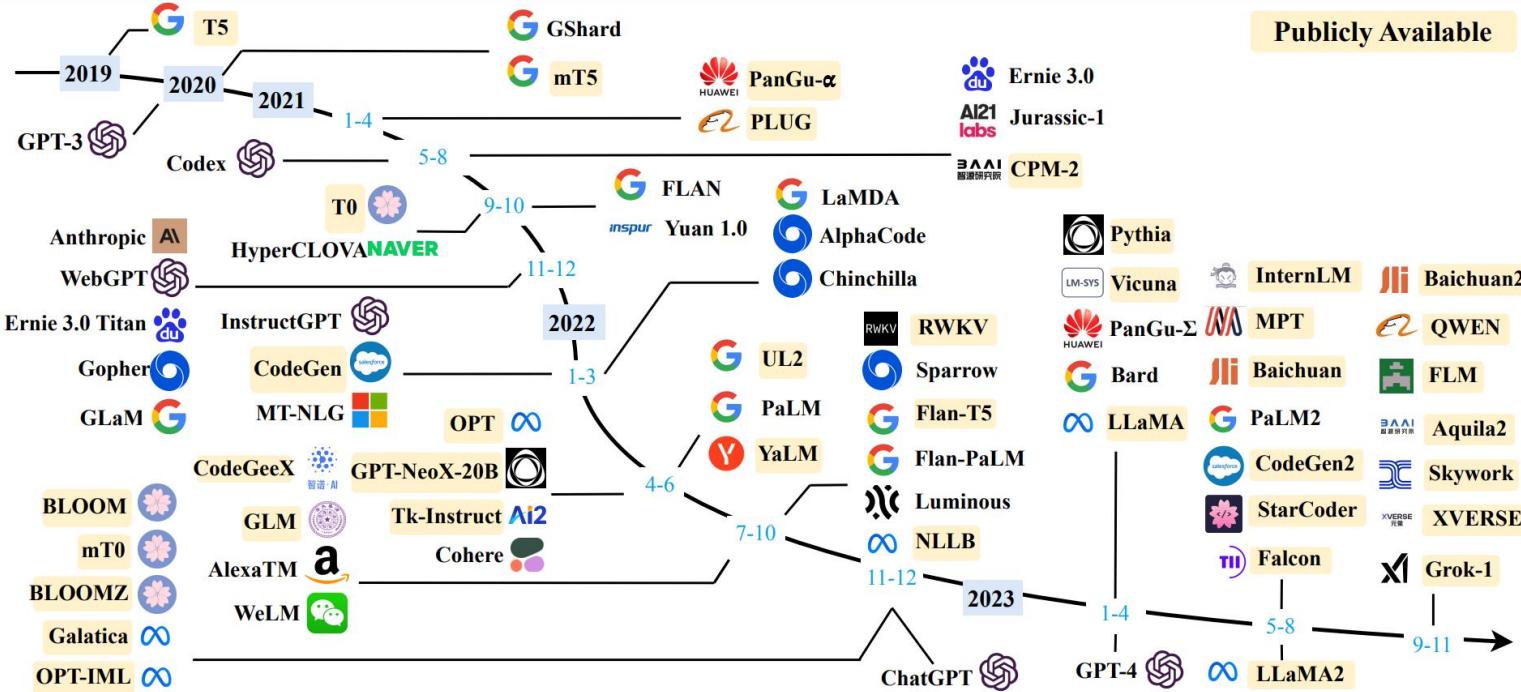


北京航空航天大學
BEIHANG UNIVERSITY

人工智能研究院
Institute of Artificial Intelligence

Intro to Large Language Models (LLMs)

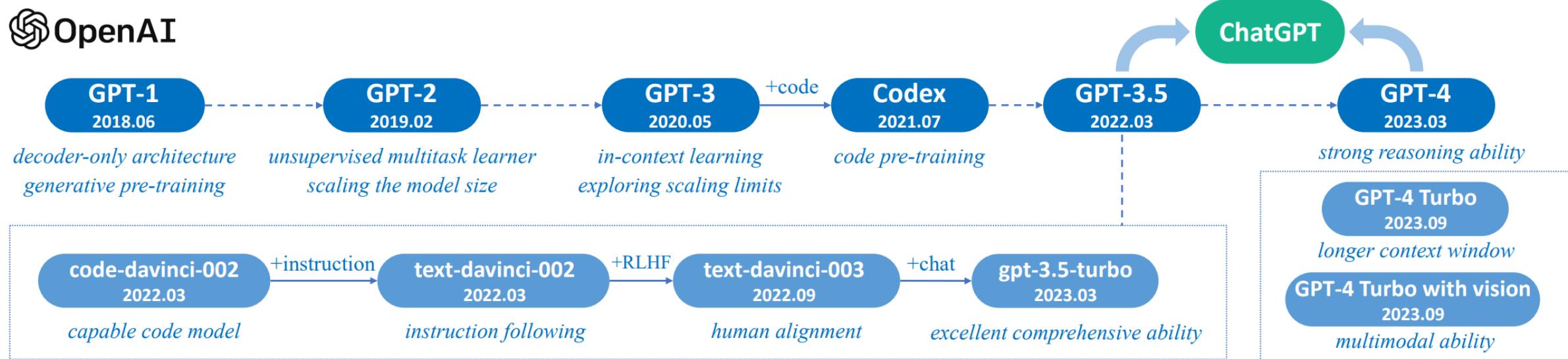
以GPT为代表的基础大模型技术在近年来发展迅猛



- ✓ 2018.06: GPT1.0 [117M Param]
- ✓ Generative Pre-training + Finetuning
- ✓ 2019.02: GPT2.0 [1.5B Param]
- ✓ Unsupervised Multitask Learners
- ✓ 2020.05: GPT3.0 [175B Param]
- ✓ Few-Shot Learner
- ✓ ... , ...
- ✓ 2022.11: ChatGPT(GPT3.5) [200B Param]
- ✓ Finetuning on GPT3.0 & 4096 Tokens
- ✓ 2023.03: GPT4.0 [? Param]
- ✓ 32768 Tokens

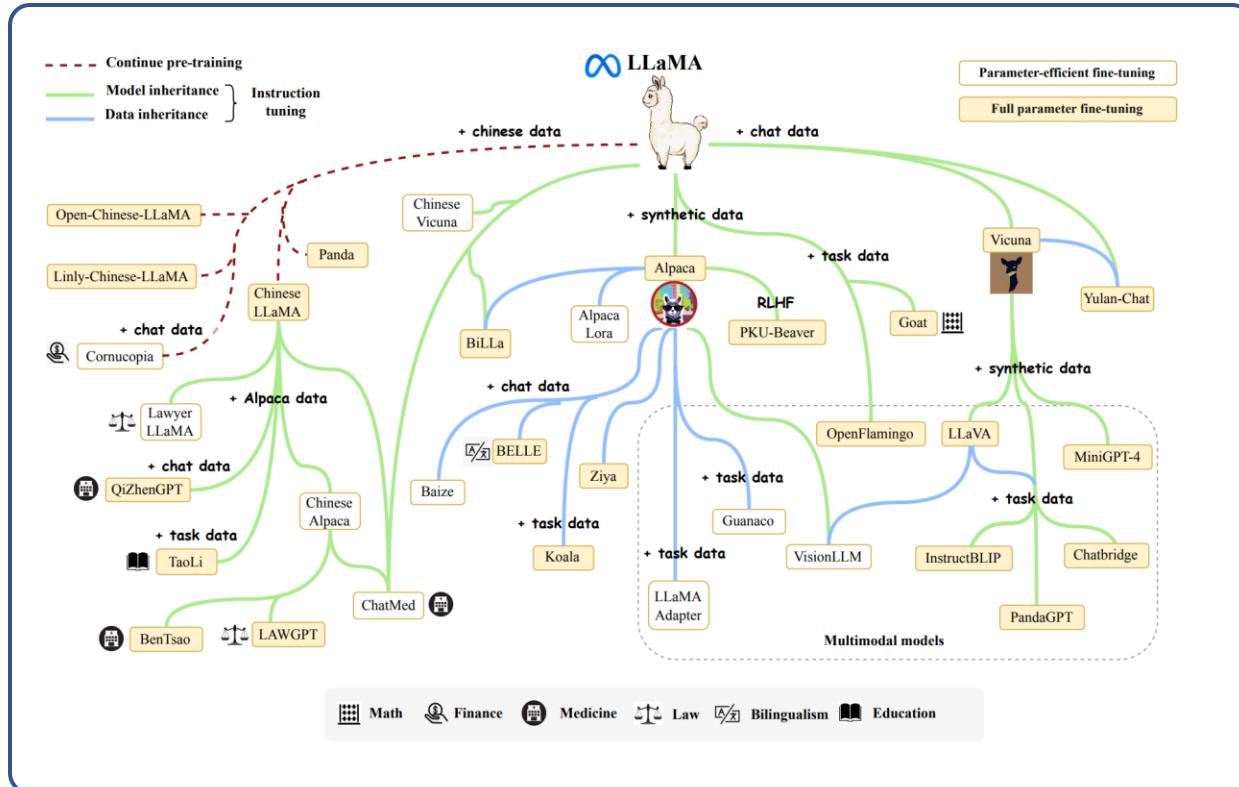
基础大模型的发展趋势: 从纯语言模型发展到多模态大模型

OpenAI GPT-4已经具备处理 图像 模态的能力



基础大模型的发展趋势: 从纯语言模型发展到多模态大模型

开源领域: 基于大语言架构 LLaMa 衍生出了众多的多模态大模型 具备对 图像 视频 音频 等模态的处理能力



X-InstructBLIP: A Framework for aligning X-Modal instruction-aware representations to LLMs and Emergent Cross-modal Reasoning

Artemis Panagopoulou, Le Xue*, Ning Yu*, Junnan Li, Dongxu Li, Shafiq Joty, Ran Xu, Silvio Savarese, Caiming Xiong, and Juan Carlos Niebles
arXiv 1234.56789 | Open in Colab *equal mentorship contribution.

Overview

LLaVA: Large Language and Vision Assistant

Visual instruction tuning towards large language and vision models with GPT-4 level capabilities.

[!] [LLaVA-NeXT Blog](#) | [Project Page](#) | [Demo](#) | [Data](#) | [Model Zoo](#)

Community Contributions: [llama.cpp](#) | [Colab](#) | [Space](#) | [Replicate](#) | [AutoGen](#) | [BakLLaVA](#)

OpenFlamingo

pypi package 2.0.1

[Paper](#) | Blog posts: [1](#), [2](#) | [Demo](#)



北京航空航天大學
BEIHANG UNIVERSITY

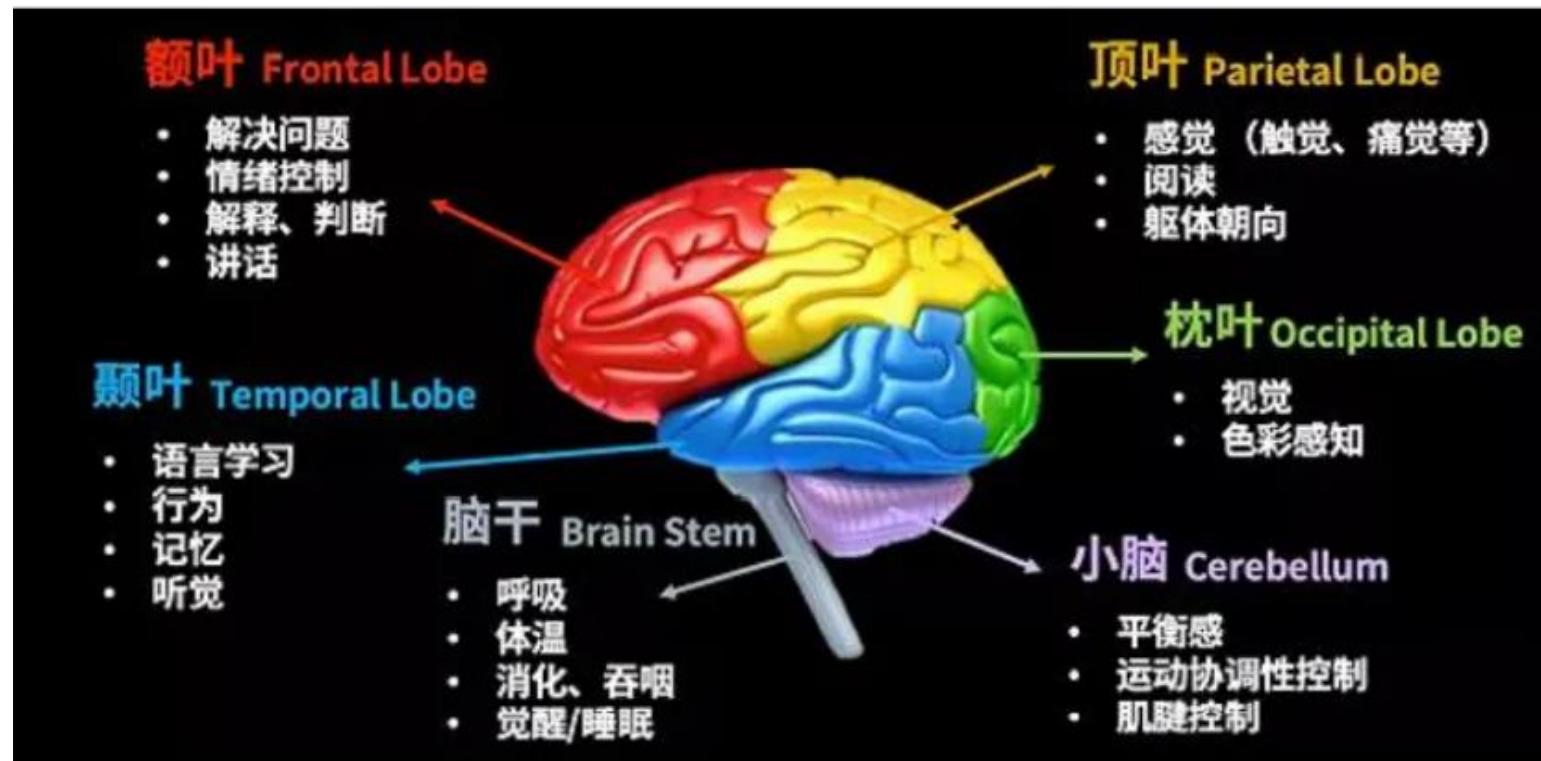
人工智能研究院
Institute of Artificial Intelligence

Intro to Vision-Language Pre-training (VLP)

为什么需要关注多模态预训练问题？从人脑机制启发

人脑“视觉概念与语言概念一致性”的认知机制是我们采取多模态预训练模型的可靠性依据之一

1. 人类约有70%的信息依靠视觉获得，剩余约20%~30%的信息依靠听觉和触觉
2. 互联网提供了90%的图文大数据，而文本数据只占了10%



Vision-Language (VL) 发展历史

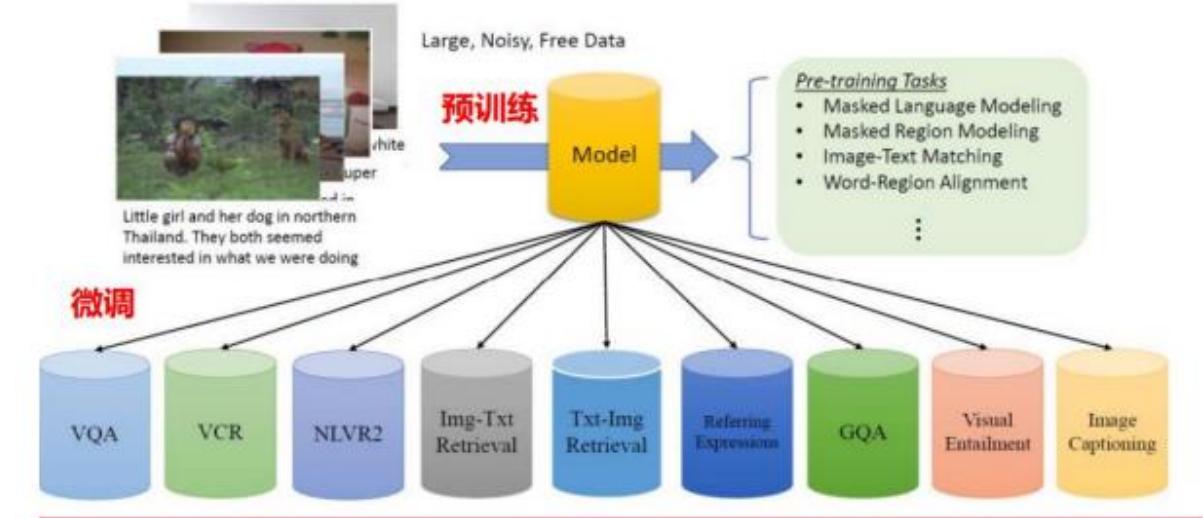
VL学习的发展分为三个阶段

1. 2014-2018年：专门的模型被设计用于不同的任务
2. 2019-2021年 (BERT诞生) : 通过使用高质量标签的VL数据集进行预训练，模型能够学习视觉和语言的联合表征
 - ViLBERT
 - UNITER



VL预训练模型 (VL-PTM)

- 2017年-2018年, NLP-PTMs → PTMs+微调模式→SOTA性能
- 2019年 VL-PTMs 开启研究工作



Vision-Language (VL) 发展历史

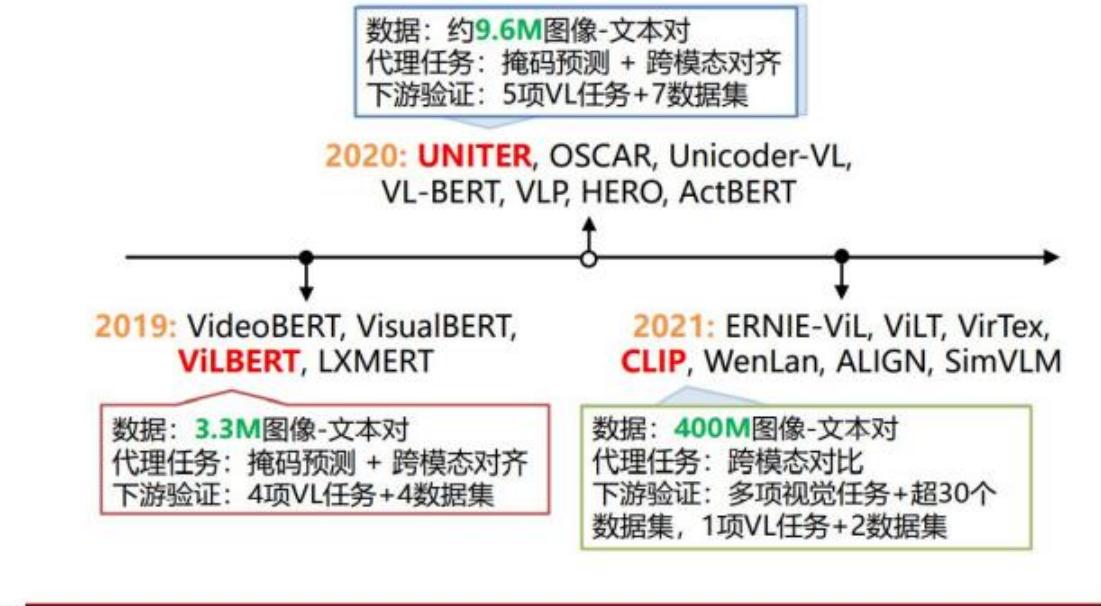
VL学习的发展分为三个阶段

3. 2021年-今：随着CLIP的出现，期望在更大的弱标签数据集上预训练VL模型，并通过VL预训练获得性能强大的基于零样本或少样本的视觉模型

- CLIP及改进工作（纯编码器架构）
- BLIP系列工作（编码-解码器架构）

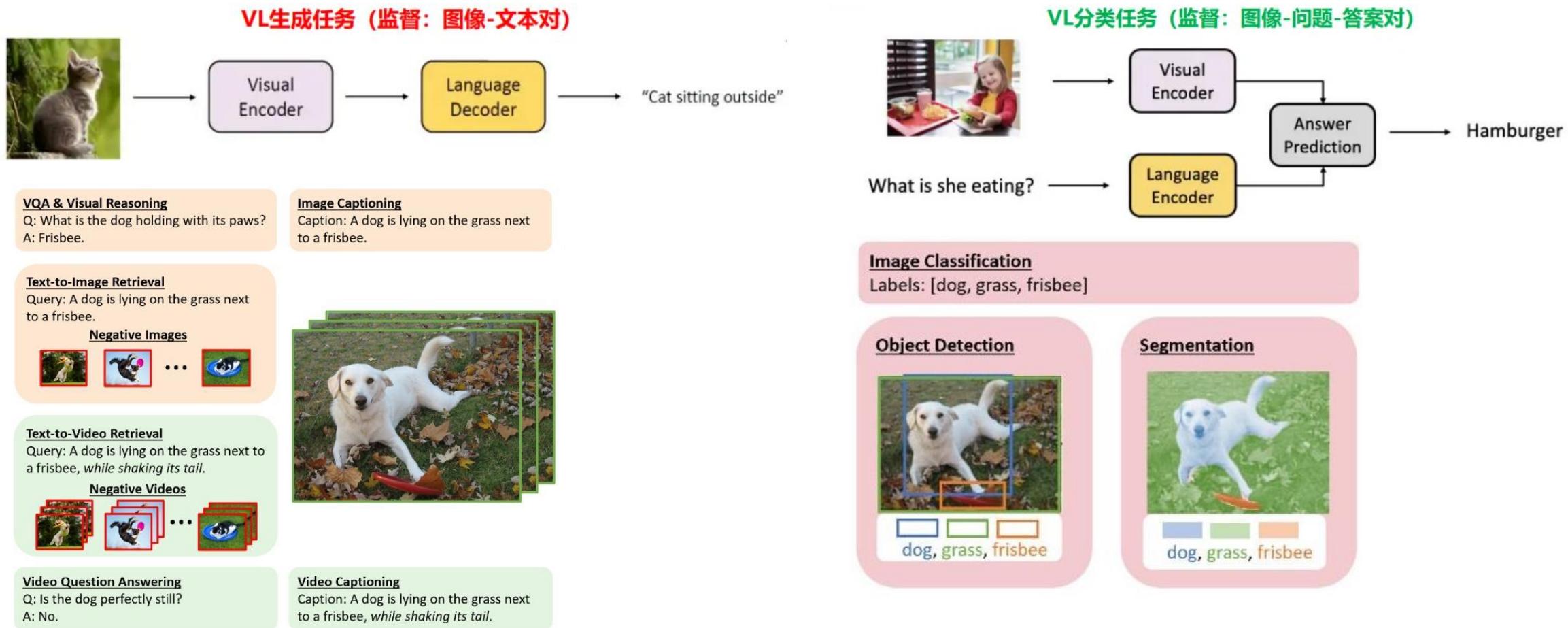


VL-PTMs的演进 (2019-2021)



多模态任务 / 下游任务

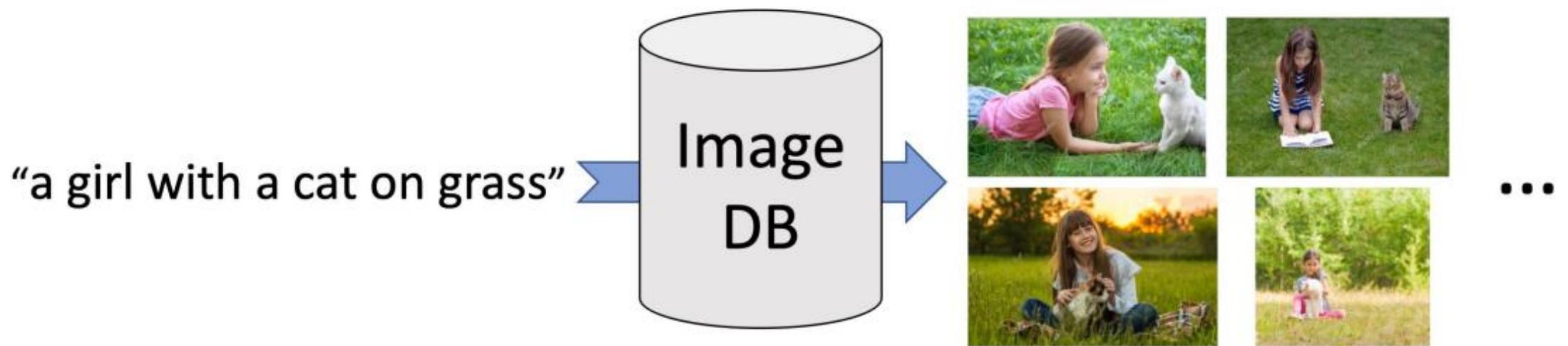
VL任务构成了多模态领域各种任务的基础 包括生成式任务和判别式任务



多模态任务 / 下游任务

一些常见的下游任务：

1. Text-Image / Image-Text Retrieval



多模态任务 / 下游任务

一些常见的下游任务：

2. Captioning & Visual Question Answering



What color are her eyes?
What is the mustache made of?



How many slices of pizza are there?
Is this a vegetarian pizza?



Is this person expecting company?
What is just under the tree?



Does it appear to be rainy?
Does this person have 20/20 vision?

多模态任务 / 下游任务

一些常见的下游任务：

3. Visual Entailment

视觉蕴涵是最近提出的多模态推理任务，其目标是预测一段文本与一幅图像的逻辑关系（蕴含/中立/矛盾）



Premise

+

- Two woman are holding packages.
- The sisters are hugging goodbye while holding to go packages after just eating lunch.
- The men are fighting outside a deli.

=

- Entailment
- Neutral
- Contradiction

Hypothesis

Answer

多模态任务 / 下游任务

一些常见的下游任务：

4. Natural Language for Visual Reasoning



The left image contains twice the number of dogs as the right image, and at least two dogs in total are standing.

true



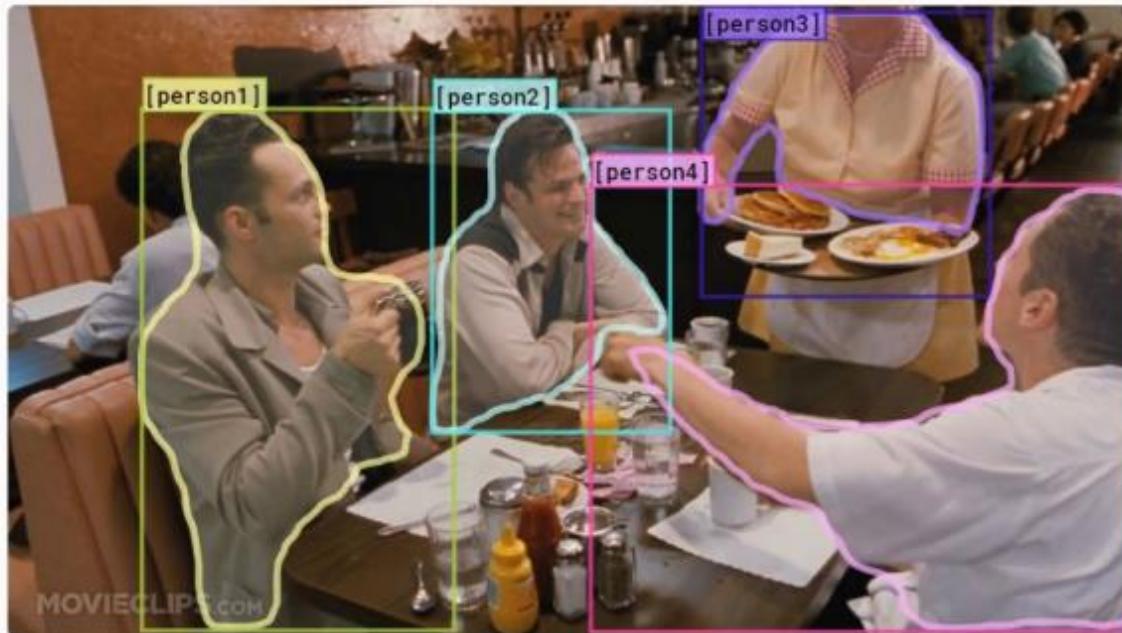
One image shows exactly two brown acorns in back-to-back caps on green foliage.

false

多模态任务 / 下游任务

一些常见的下游任务：

5. Visual Commonsense Reasoning 回答问题+推理为什么这么回答



Why is [person4] pointing at [person1]?

- a) He is telling [person3] that [person1] ordered the pancakes.
- b) He just told a joke.
- c) He is feeling accusatory towards [person1].
- d) He is giving [person1] directions.

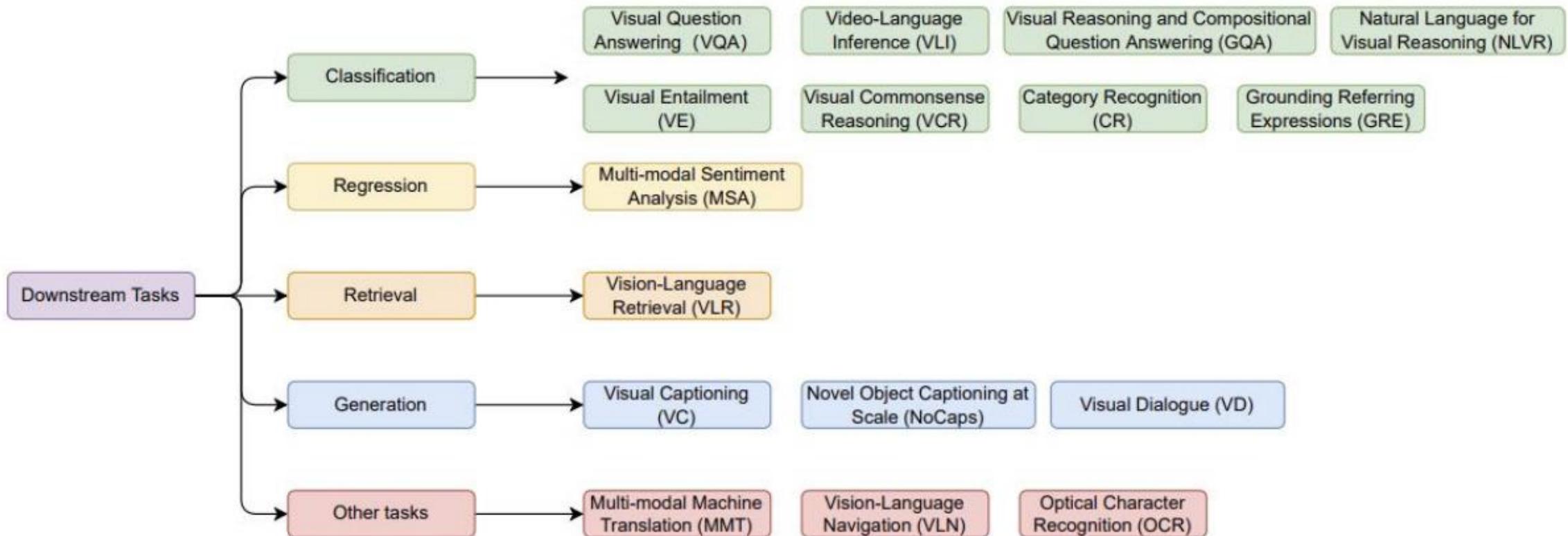
I choose (a) because:

- a) [person1] has the pancakes in front of him.
- b) [person4] is taking everyone's order and asked for clarification.
- c) [person3] is looking at the pancakes and both she and [person2] are smiling slightly.
- d) [person3] is delivering food to the table, and she might not know whose order is whose.

多模态任务 / 下游任务

一些常见的下游任务：

6. More...



VLP的常用模型架构

用于 Feature Extraction

1. 文本模态的特征提取

- LSTM
- BERT

2. 图像模态的特征提取

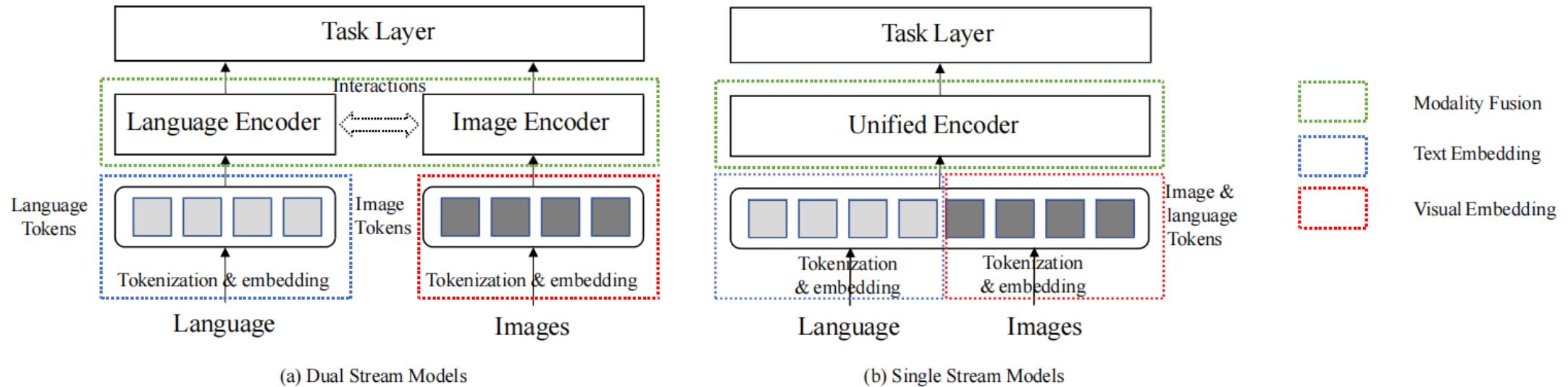
- CNN: Feature Maps
- ROI: Detected Objects
- ViT: 16*16 patch
- Pixel

3. 视频模态的特征提取

- 先抽取帧图片特征，再把时序图片特征变成定长特征
- temporal maxpooling / mean pooling

VLP的常用模型架构

Model Architecture

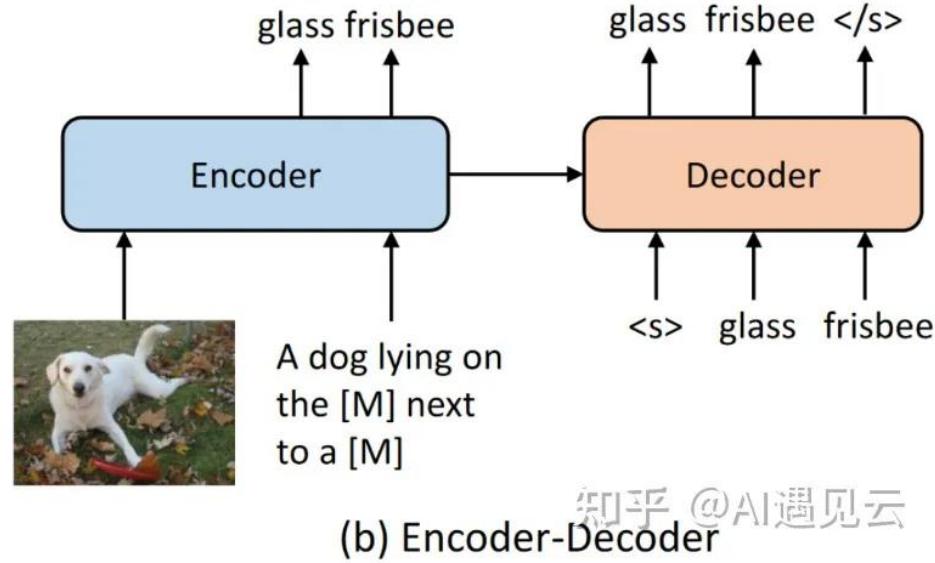
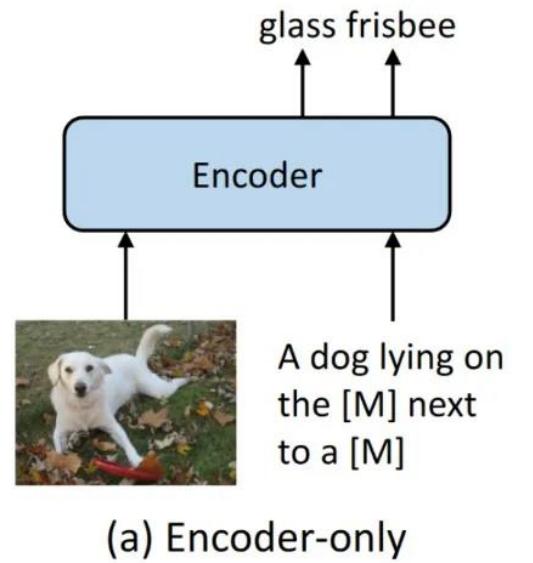


1. Single-Stream VS. Dual-Stream

- 单流模式指的是将视觉和文本编码特征组合在一起，然后输入单个 Transformer 块中，通过合并注意力来融合多模态输入，通常也被叫做 merged attention。参数效率更高。
- 双流模式是指视觉和文本编码特征没有组合在一起，而是独立输入到两个不同的 Transformer块，不共享参数，而是通过交叉注意力实现跨模态交互，因此也被叫做 co-attention。

VLP的常用模型架构

Model Architecture



2. Encoder-only VS. Encoder-decoder

- 大多数 VLP 模型仅包含编码器，其中跨模态表示直接输入到基于 MLP 的输出层来生成最终输出。这种设计自然适合 VL 理解任务，例如 VQA 和视觉推理
- 近期衍生除了一些基于 Transformer 的编码器-解码器架构，跨模态表示首先被送入解码器，然后送入输出层。在这些模型中，解码器同时关注编码器表示和先前生成的标记，以自回归地方式产生输出。可以实现各种图像-文本任务的统一和 VLP 模型的零样本/少样本学习，并且也很自然地适合生成类的任务。

预训练形式1. Masked Language Modeling (MLM)

$$\mathcal{L}_{\text{MLM}}(\theta) = -E_{(\mathbf{w}, \mathbf{v}) \sim D} \log P_\theta (\mathbf{w}_m \mid \mathbf{w}_{\setminus m}, \mathbf{v})$$



- Image Regions: $\mathbf{v} = \{v_1, \dots, v_K\}$
- Sentence Tokens: $\mathbf{w} = \{w_1, \dots, w_T\}$
- Masking Indices: $\mathbf{m} \in \mathbb{N}^M$

Masked Language Modeling (MLM)

- 十分straightforward的一种思路，把图像块处理为一种特殊的token
- 迁移自LLMs中的Masked-Language-Modeling目标，结合图像和文本模态进行mask预测
- 最大化 mask内预测为正确token的期望

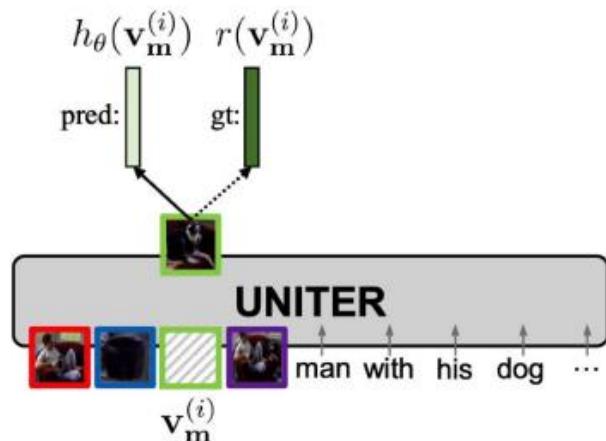
预训练形式2. Masked Vision Modeling (MVM)

- 模仿MLM，只不过是对图片或者图片patch的 token 进行随机mask
- 使需要在给定剩余的视觉特征和所有文本特征的情况下重建被 mask 的视觉特征

2.1 Masked Region Feature Regression (MRFR)

$$f_{\theta} (\mathbf{v}_m \mid \mathbf{v}_{\setminus m}, \mathbf{w}) = \sum_{i=1}^M \left\| h_{\theta} \left(\mathbf{v}_m^{(i)} \right) - r \left(\mathbf{v}_m^{(i)} \right) \right\|_2^2$$

- Image Regions: $\mathbf{v} = \{v_1, \dots, v_K\}$
- Sentence Tokens: $\mathbf{w} = \{w_1, \dots, w_T\}$
- Masking Indices: $\mathbf{m} \in \mathbb{N}^M$



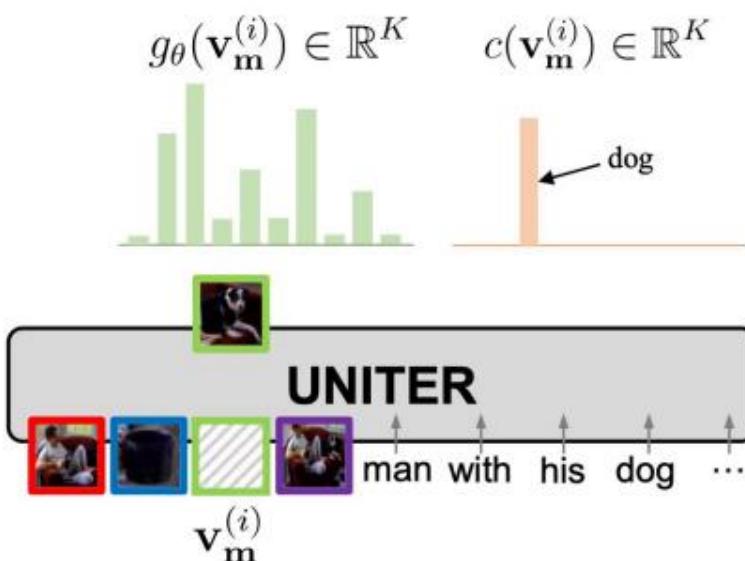
- 将 mask 特征的模型输出转换为与原始视觉特征相同维度的向量，并在原始视觉特征和向量之间应用 L2 回归

预训练形式2. Masked Vision Modeling (MVM)

- 模仿MLM，只不过是对图片或者图片patch的 token 进行随机mask
- 使需要在给定剩余的视觉特征和所有文本特征的情况下重建被 mask 的视觉特征

2.2 Masked Region Classification (MRC)

$$f_{\theta}(\mathbf{v}_m \mid \mathbf{v}_{\setminus m}, \mathbf{w}) = \sum_{i=1}^M \text{CE}\left(c\left(\mathbf{v}_m^{(i)}\right), g_{\theta}\left(\mathbf{v}_m^{(i)}\right)\right)$$



- Image Regions: $\mathbf{v} = \{v_1, \dots, v_K\}$
- Sentence Tokens: $\mathbf{w} = \{w_1, \dots, w_T\}$
- Masking Indices: $\mathbf{m} \in \mathbb{N}^M$

- 学习预测 mask 特征的对象语义类。VLP 模型首先将 mask 特征的输出馈送到 FC 层以预测对象类别的分数，然后进一步通过 softmax 函数将其转换为预测归一化分布

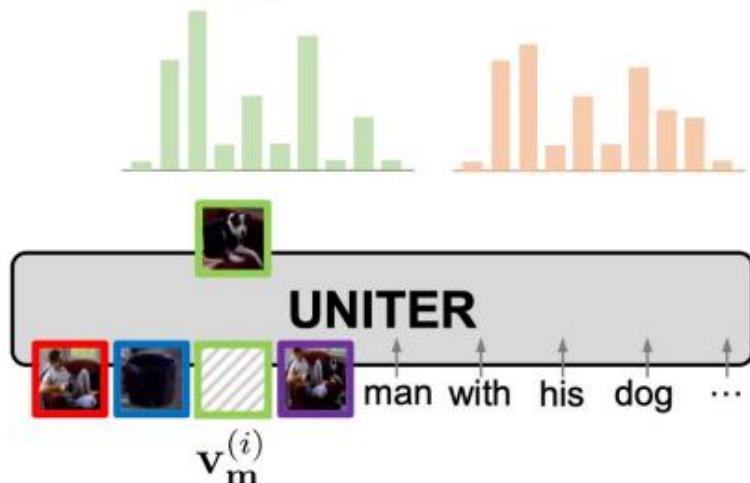
预训练形式2. Masked Vision Modeling (MVM)

- 模仿MLM，只不过是对图片或者图片patch的 token 进行随机mask
- 使需要在给定剩余的视觉特征和所有文本特征的情况下重建被 mask 的视觉特征

2.3 Masked Region Classification – KL Divergence (MRC-kl)

$$f_{\theta} (\mathbf{v}_m \mid \mathbf{v}_{\setminus m}, \mathbf{w}) = \sum_{i=1}^M D_{KL} \left(\tilde{c} \left(\mathbf{v}_m^{(i)} \right) \| g_{\theta} \left(\mathbf{v}_m^{(i)} \right) \right)$$

$$g_{\theta}(\mathbf{v}_m^{(i)}) \in \mathbb{R}^K \quad \tilde{c}(\mathbf{v}_m^{(i)}) \in \mathbb{R}^K$$



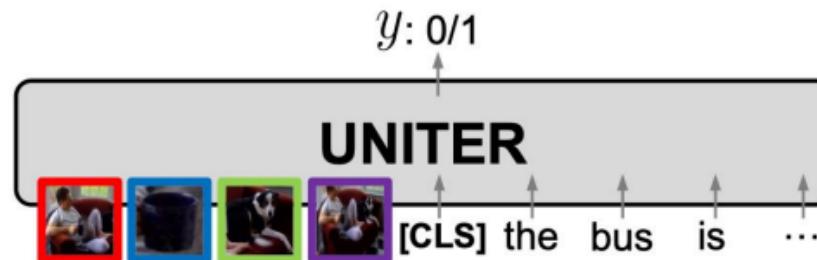
- Image Regions: $\mathbf{v} = \{v_1, \dots, v_K\}$
- Sentence Tokens: $\mathbf{w} = \{w_1, \dots, w_T\}$
- Masking Indices: $\mathbf{m} \in \mathbb{N}^M$

- 与MRC不同，在这里，不使用绝对的类语义标签作为ground truth，而是预测一个 distribution over semantic classes (soft label)，使用KL散度计算损失

预训练形式3. Image-Text Matching (ITM)

- 判断当前 `text-image` pair是不是匹配：将图片的`[IMG]` token和文本的`[CLS]` token进行匹配

$$\mathcal{L}_{\text{ITM}}(\theta) = -E_{(\mathbf{w}, \mathbf{v}) \sim D} [y \log s_\theta(\mathbf{w}, \mathbf{v}) + (1 - y) \log (1 - s_\theta(\mathbf{w}, \mathbf{v}))]$$



最常见形式：Vision-Language Contrastive Learning

- VLP 模型将两种模式的融合表示提供给 FC 层和 sigmoid 函数以预测 0 到 1 之间的分数，其中 0 表示视觉和语言不匹配，1 表示视觉和语言匹配。
- 在训练期间，VLP 模型在每一步从数据集中采样正对或负对。负对是通过用从其他样本中随机选择的替换成对样本中的视觉或文本来创建的。

A summary

VLP的特征提取：文本模态 / 图像模态 / 视频模态

VLP的架构区分 (模态特征是否进行融合处理) Single-Stream VS. Dual-Stream

VLP的架构区分 (是否包含编码器解码器) Encoder-only VS. Encoder-decoder

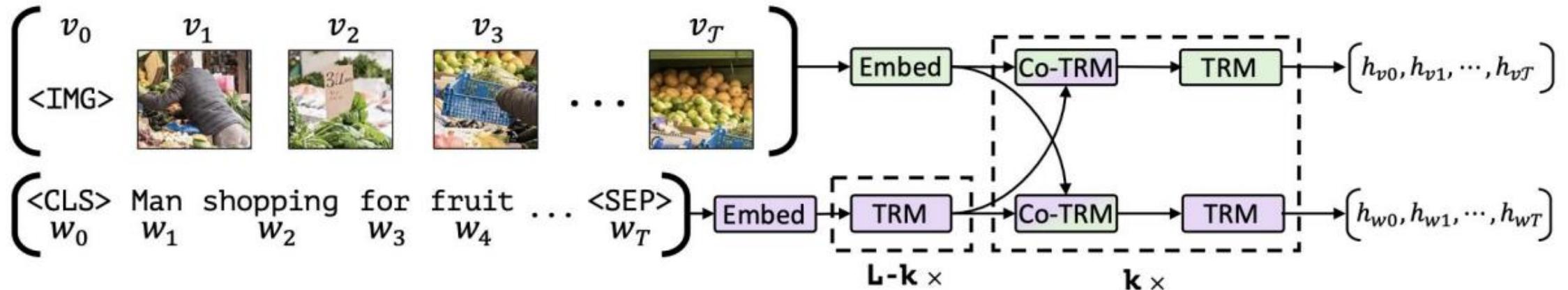
预训练形式1. Masked Language Modeling (MLM)

预训练形式2. Masked Vision Modeling (MVM)

预训练形式3. Image-Text Matching (ITM)

代表性的早期工作

ViLBERT



- dual-stream
- Image Features: detected objects (RoI) + 5-d spatial location vector
5-d vector: region position (normalized top-left and bottom-right coordinates)
and the fraction of image area covered (区域坐标+区域面积)
- Co-TRM: 跨模态attention

代表性的早期工作

ViLBERT

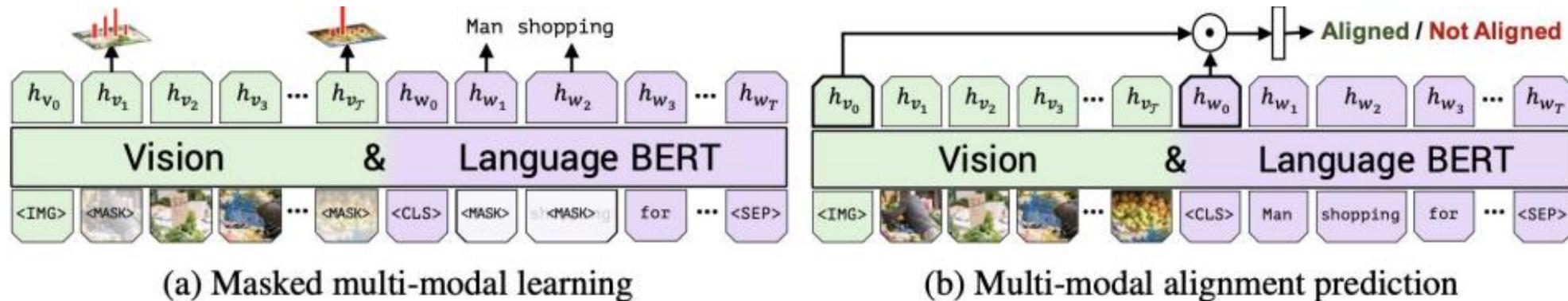
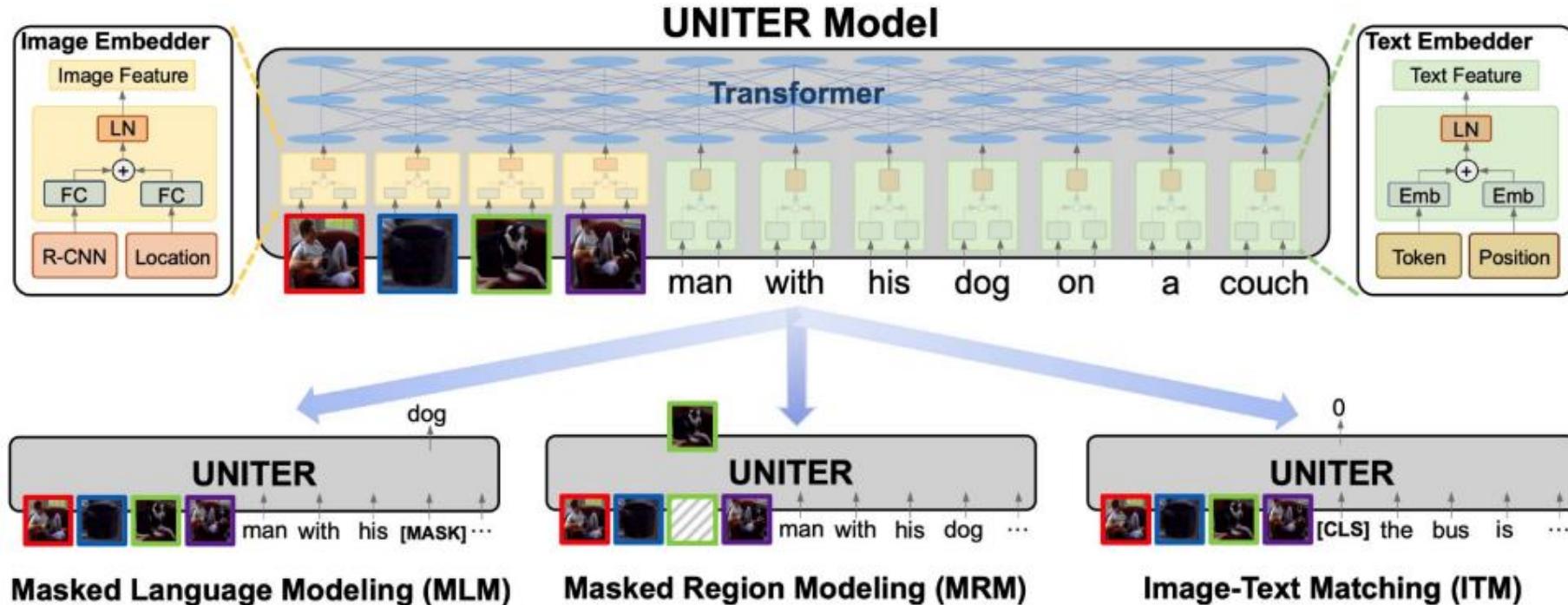


Figure 3: We train ViLBERT on the Conceptual Captions [24] dataset under two training tasks to learn visual grounding. In masked multi-modal learning, the model must reconstruct **image region categories** or words for masked inputs given the observed inputs. In multi-modal alignment prediction, the model must predict whether or not the caption describes the image content.

- Soft-label MVM: the model predicts a distribution over semantic classes for the corresponding image region; KL divergence
- ITM: Predict Aligned / Not Aligned

代表性的早期工作

UNITER

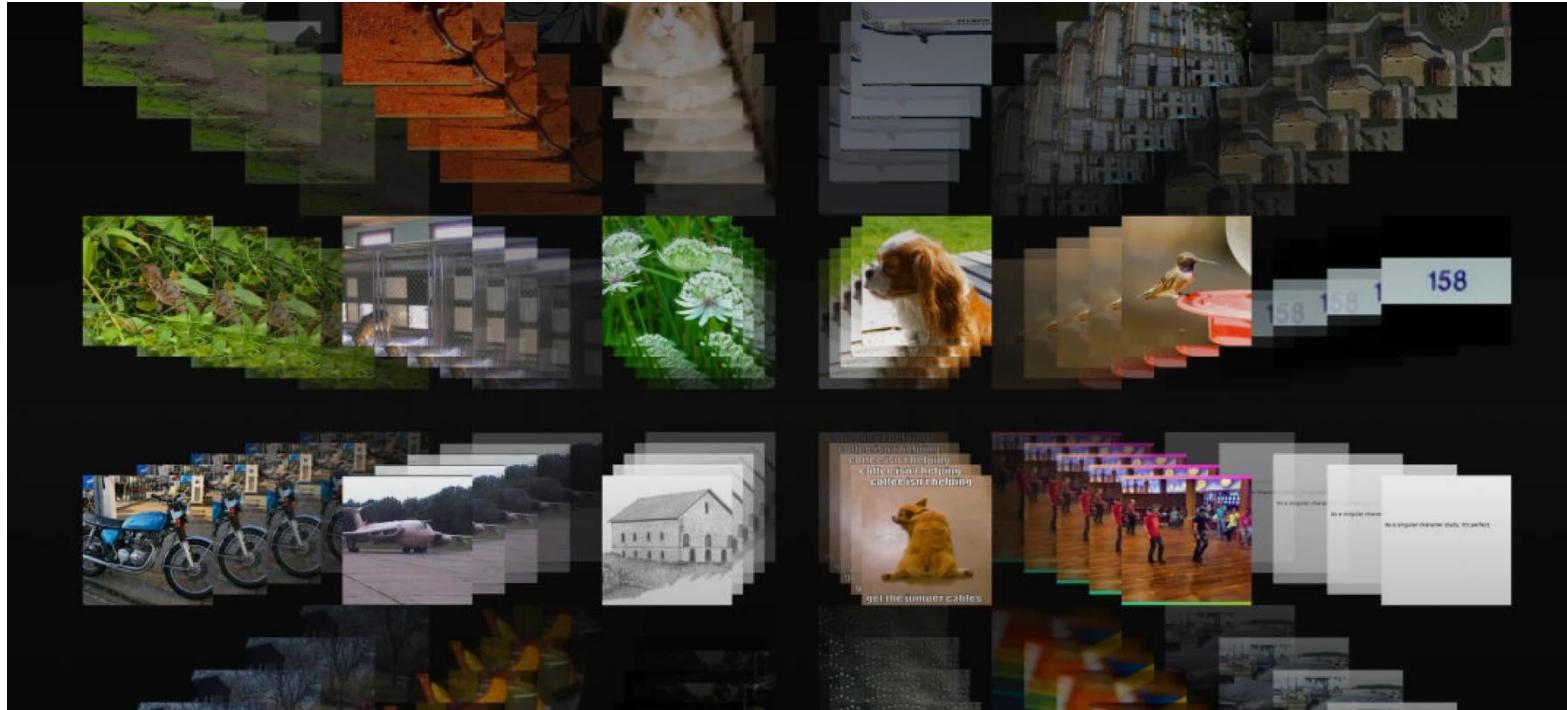


- single-stream
- Image Input: detected objects (RoI)
- Text Input: tokens
- Pretrained on MLM / MVM / ITM

双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

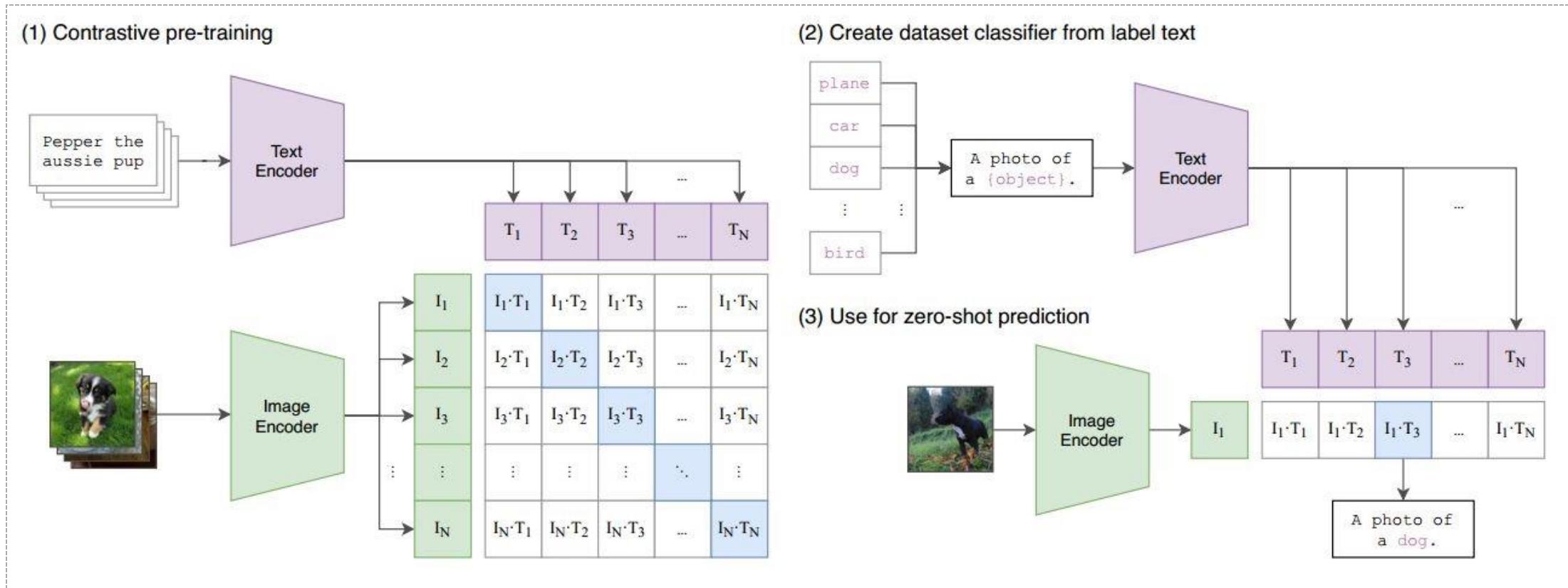
带领视觉语言预训练技术迈向新的时代



- Motivation: 当前的计算机视觉系统被训练成一种分类系统，严重影响了其在未见类别上的泛化性（需要用额外的有标注数据）。因此直接从嘈杂的原始文本中学习是一个更有前景的可替代方法，其能带来大量广泛的有监督数据。

双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training



- dual-stream
- train on large weakly paired text-image examples (WIT数据集：包含400M互联网采集的图文对)

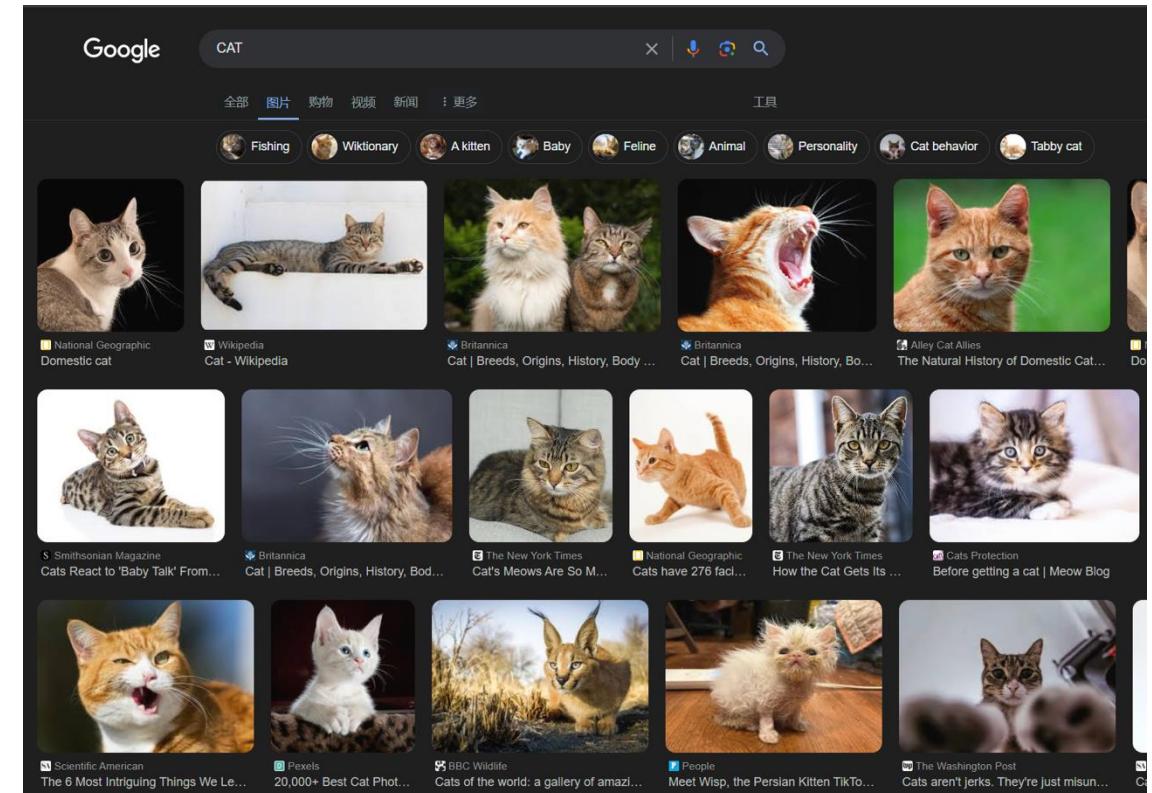
双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- Dataset: 基于互联网采集一个足够大的图像-文本对数据集

在此以前，用于VLP的大规模数据集主要有3个，MS-COCO、Visual Genome、YFCC100M。前两者是人工标注的，质量高但是数据量小，大约只有0.1M。YFCC100M大约有100M图片，但是质量堪忧，有些标题只有无意义的名字，过滤后，这个数据集缩小至15M，大约和ImageNet差不多大。

考虑到现有数据集太小，OpenAI自己构建了一个400M的数据集，使用500K个查询进行搜索，每个查询大约有20K个“图像一文本”对。该数据集被称为WIT (webimagetext)，数据量和GPT-2使用的差不多。



双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- 学习策略：多模态对比学习 属于ITM的一种

对比学习的目的是学习一个**嵌入函数**： $f: \chi \rightarrow E$ ，以使得相似样本在嵌入空间内的距离度量相近而不相似的远离。早期通常采用 triplet loss 训练：

$$L = \max(d(a, p) - d(a, n) + margin, 0)$$

多模态对比学习，则是针对不同模态的数据对，进行对比学习的操作（例如图像-文本对），对于给定的数据集中的文本和图像，我们分别利用映射函数将其映射到相同嵌入空间。在训练时，给定训练样本对，最大化匹配样本对的嵌入内积，最小化其他不匹配样本对的内积。实现这一途径的目标函数，我们称之为图文对比损失：image-text contrastive (ITC) loss

$$\mathcal{L}_{I \rightarrow T} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp(d(z_i^I, z_i^T)/\tau)}{\sum_{k=1}^N \exp(d(z_i^I, z_k^T)/\tau)},$$
$$\mathcal{L}_{T \rightarrow I} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp(d(z_i^T, z_i^I)/\tau)}{\sum_{k=1}^N \exp(d(z_i^T, z_k^I)/\tau)},$$
$$\mathcal{L}_{ITC} = \frac{1}{2}(\mathcal{L}_{I \rightarrow T} + \mathcal{L}_{T \rightarrow I}),$$

双流编码器架构：CLIP及相关改进工作

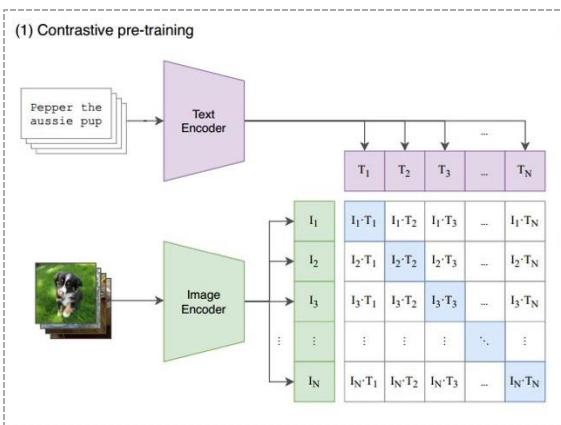
CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- Pretrained on ITM Task

$$\mathcal{L}_{ITC} = \frac{1}{2}(\mathcal{L}_{I \rightarrow T} + \mathcal{L}_{T \rightarrow I}),$$

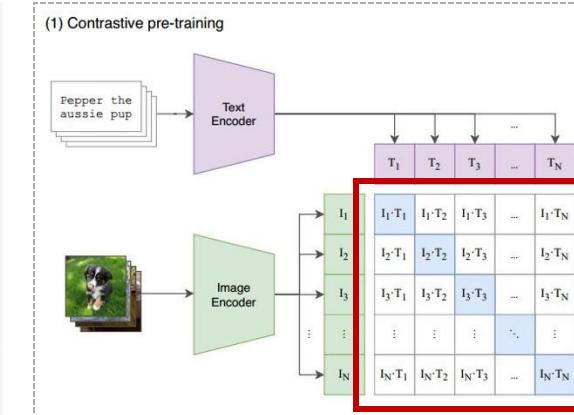
$$\mathcal{L}_{I \rightarrow T} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp(d(z_i^I, z_i^T)/\tau)}{\sum_{k=1}^N \exp(d(z_i^I, z_k^T)/\tau)},$$

$$\mathcal{L}_{T \rightarrow I} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp(d(z_i^T, z_i^I)/\tau)}{\sum_{k=1}^N \exp(d(z_i^T, z_k^I)/\tau)},$$



对图像和文本，分别应用两个Encoder，Image encoder可以是resnet或vit，而Text encoder采用transformer结构得到文本的embedding $[T_1, T_2, \dots, T_N]$ 以及图像embedding $[I_1, I_2, \dots, I_N]$ ，将其分别进行内积计算，得到一个内积矩阵

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, 1] - minibatch of aligned texts
# W_id_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]
# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)
# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```



内积矩阵

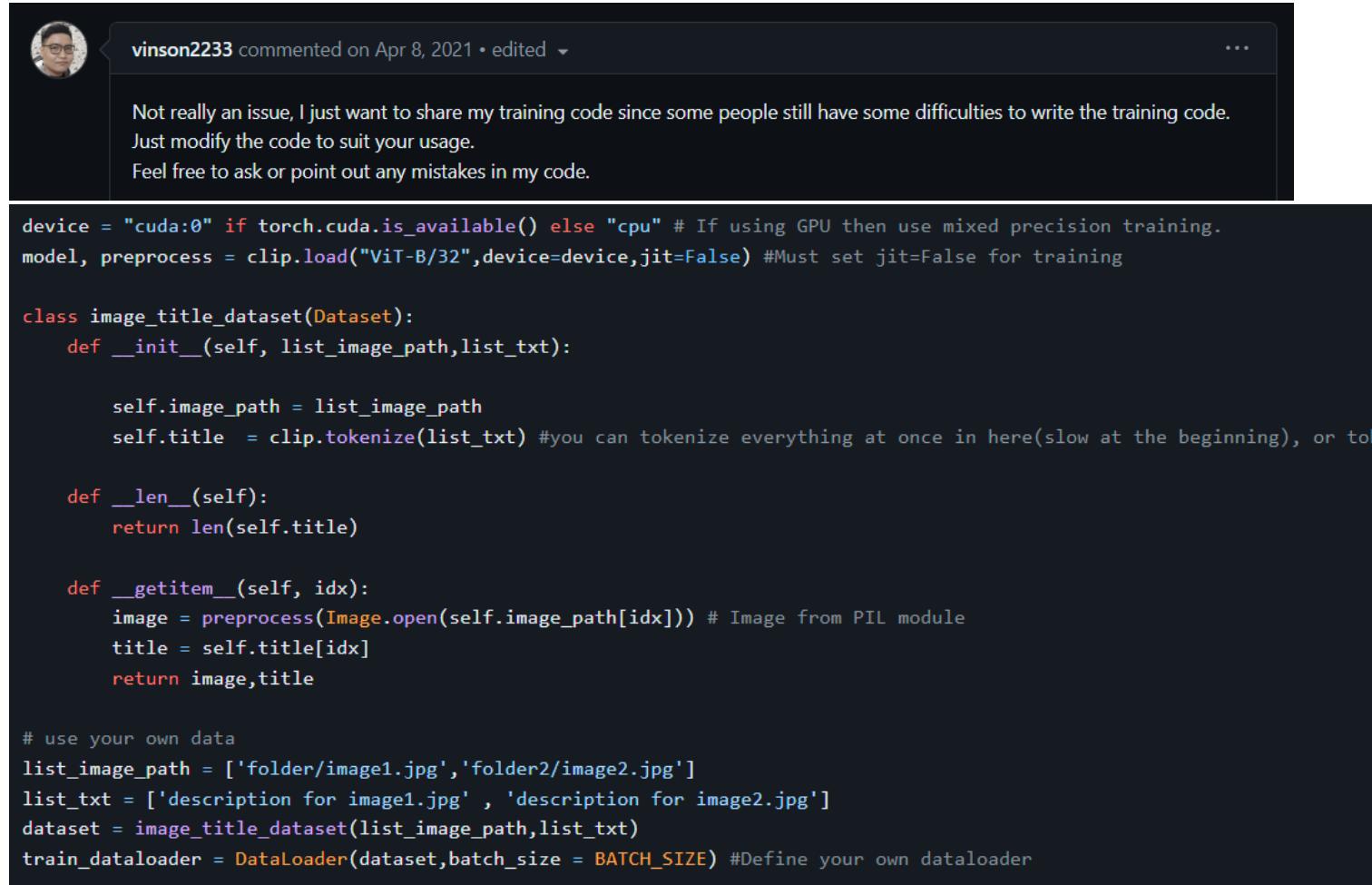
优化目标是最大化矩阵对角线上的元素，而最小化与不相关特征的内积（矩阵的其他元素）

损失函数：使用 对称的cross-entropy loss，对内积矩阵的每一行（同一张图片，不同文本标签）和每一列（同一个文本，不同的图片）分别求cross-entropy loss，将其加和作为总的损失

双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- 如何实现CLIP训练—以pytorch为例



vinson2233 commented on Apr 8, 2021 • edited

Not really an issue, I just want to share my training code since some people still have some difficulties to write the training code.
Just modify the code to suit your usage.
Feel free to ask or point out any mistakes in my code.

```
device = "cuda:0" if torch.cuda.is_available() else "cpu" # If using GPU then use mixed precision training.
model, preprocess = clip.load("ViT-B/32",device=device,jit=False) #Must set jit=False for training

class image_title_dataset(Dataset):
    def __init__(self, list_image_path,list_txt):

        self.image_path = list_image_path
        self.title = clip.tokenize(list_txt) #you can tokenize everything at once in here(slow at the beginning), or tok

    def __len__(self):
        return len(self.title)

    def __getitem__(self, idx):
        image = preprocess(Image.open(self.image_path[idx])) # Image from PIL module
        title = self.title[idx]
        return image,title

# use your own data
list_image_path = ['folder/image1.jpg','folder2/image2.jpg']
list_txt = ['description for image1.jpg' , 'description for image2.jpg']
dataset = image_title_dataset(list_image_path,list_txt)
train_dataloader = DataLoader(dataset,batch_size = BATCH_SIZE) #Define your own dataloader
```

双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- 如何实现CLIP训练—以pytorch为例

```
loss_img = nn.CrossEntropyLoss()
loss_txt = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=5e-5,betas=(0.9,0.98),eps=1e-6,weight_decay=0.2) #Params used from paper,
# add your own code to track the training progress.
for epoch in range(EPOCH):
    for batch in train_dataloader :
        optimizer.zero_grad()

        images,texts = batch

        images= images.to(device)
        texts = texts.to(device)

        logits_per_image, logits_per_text = model(images, texts)

        ground_truth = torch.arange(len(images),dtype=torch.long,device=device)

        total_loss = (loss_img(logits_per_image,ground_truth) + loss_txt(logits_per_text,ground_truth))/2
        total_loss.backward()
        if device == "cpu":
            optimizer.step()
        else :
            convert_models_to_fp32(model)
            optimizer.step()
            clip.model.convert_weights(model)
```

- 思考：为什么ground-truth是一个
arange张量？ $[0,1,2,3,\dots,\text{len}(\text{images})]$

双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- 如何实现CLIP训练—以pytorch为例

```
loss_img = nn.CrossEntropyLoss()
loss_txt = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=5e-5,betas=(0.9,0.98),eps=1e-6,weight_decay=0.2) #Params used from paper,
# add your own code to track the training progress.
for epoch in range(EPOCH):
    for batch in train_dataloader :
        optimizer.zero_grad()

        images,texts = batch

        images= images.to(device)
        texts = texts.to(device)

        logits_per_image, logits_per_text = model(images, texts)

        ground_truth = torch.arange(len(images),dtype=torch.long,device=device)

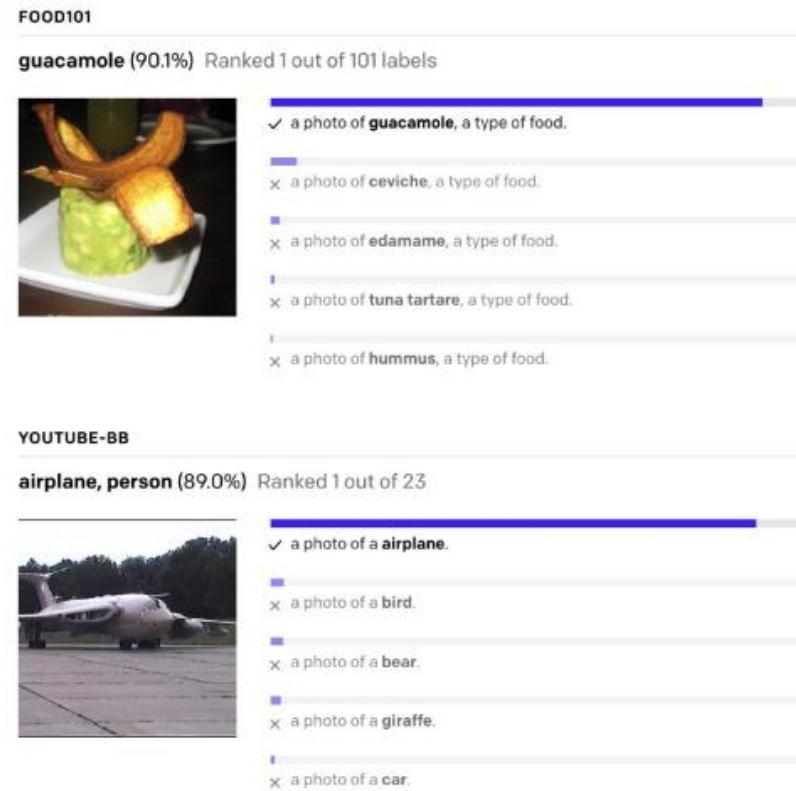
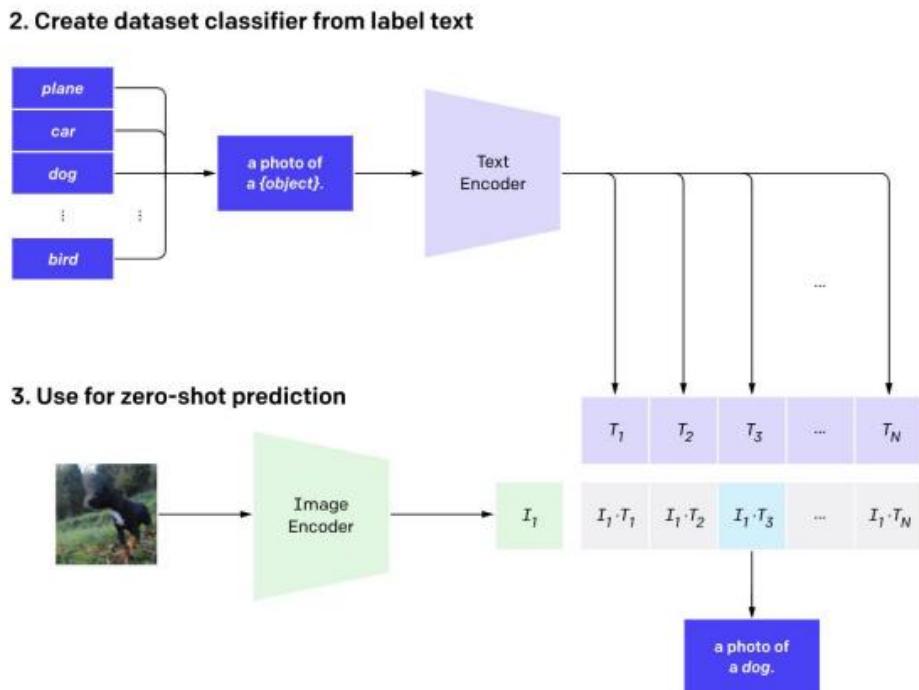
        total_loss = (loss_img(logits_per_image,ground_truth) + loss_txt(logits_per_text,ground_truth))/2
        total_loss.backward()
        if device == "cpu":
            optimizer.step()
        else :
            convert_models_to_fp32(model)
            optimizer.step()
            clip.model.convert_weights(model)
```

- 思考：为什么ground-truth是一个arange张量？ $[0,1,2,3,\dots,\text{len}(\text{images})]$
- 和以往的分类任务不同，假设我有[苹果， 苹果， 西瓜]的图像，那么交叉熵损失的输入label是 $[0,0,1]$
- 现在，对于每个图像，对应都有不同的文本模态，为了对齐两个模态，我们假设每个图像文本都是同一类的数据。那么，就需要在batch中创建一个arange张量代表虚拟的标签
- 使用对称的交叉熵，对齐文本到虚拟标签，以及图像到虚拟标签，从而使配对的图文对在特征上逼近

双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- CLIP出色的下游任务表现

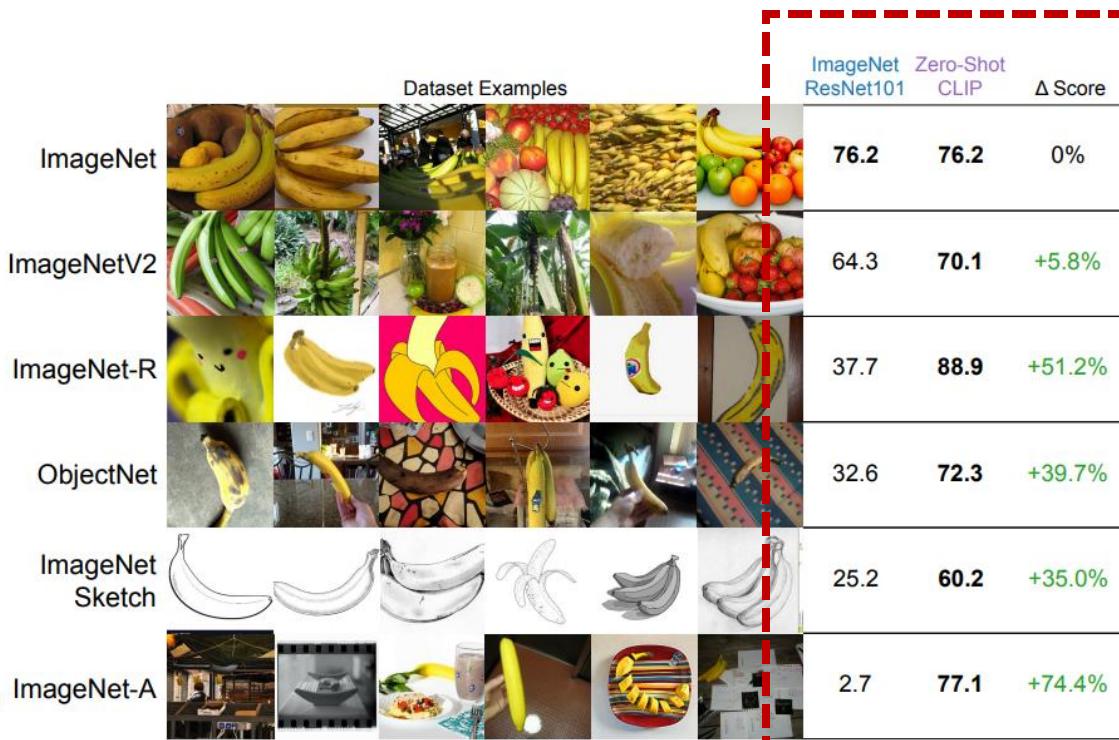
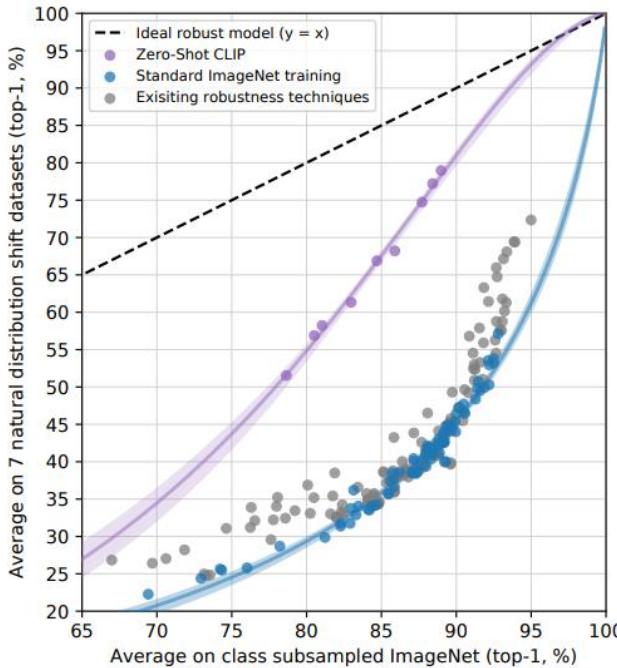


考虑大部分数据集通常是以单词作为标签，如“bird”、“plane”。因此采用**指示上下文的方式对标签进行扩展**：采用a photo of a <LABEL>. 作为推理阶段的文本输入。<LABEL>即为需要预测的zero-shot标签。将不同的标签扩展文本的embedding和待预测图像embedding求内积，得到最终预测的结果。而标签扩展可以是更详细的形式：A photo of a <LABEL>, a type of XXX 可以提升预测性能

双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- CLIP出色的下游任务表现
- 在zero-shot分类中能够准确预测各种不同的OOD样本



双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- CLIP出色的下游任务表现
- 应用CLIP进行few-shot和Linear probe分类

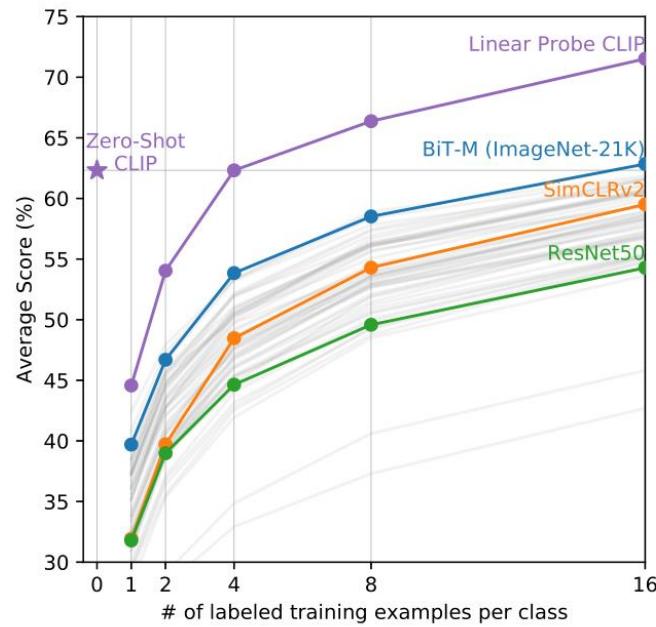
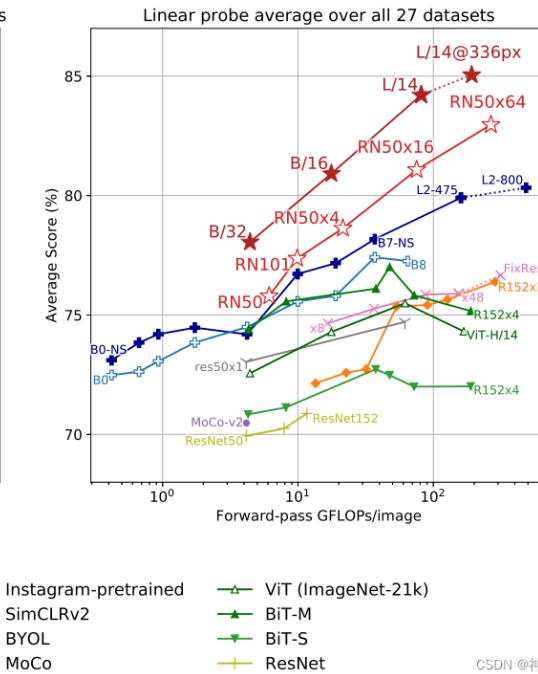
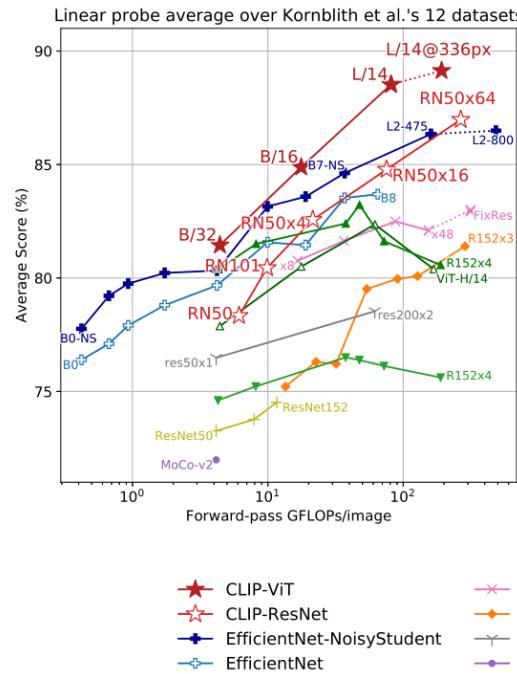


Figure 6. Zero-shot CLIP outperforms few-shot linear probes.

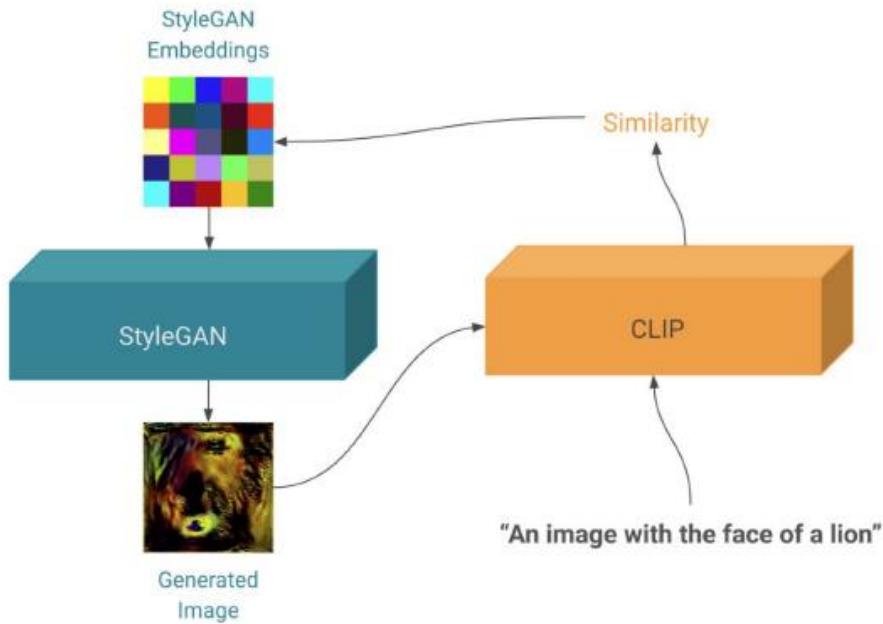


CSDN @神洛华

双流编码器架构：CLIP及相关改进工作

CLIP: Connecting text and images with Contrastive Language–Image Pre-training

- CLIP出色的下游任务表现
- Image Generation: CLIP + StyleGAN



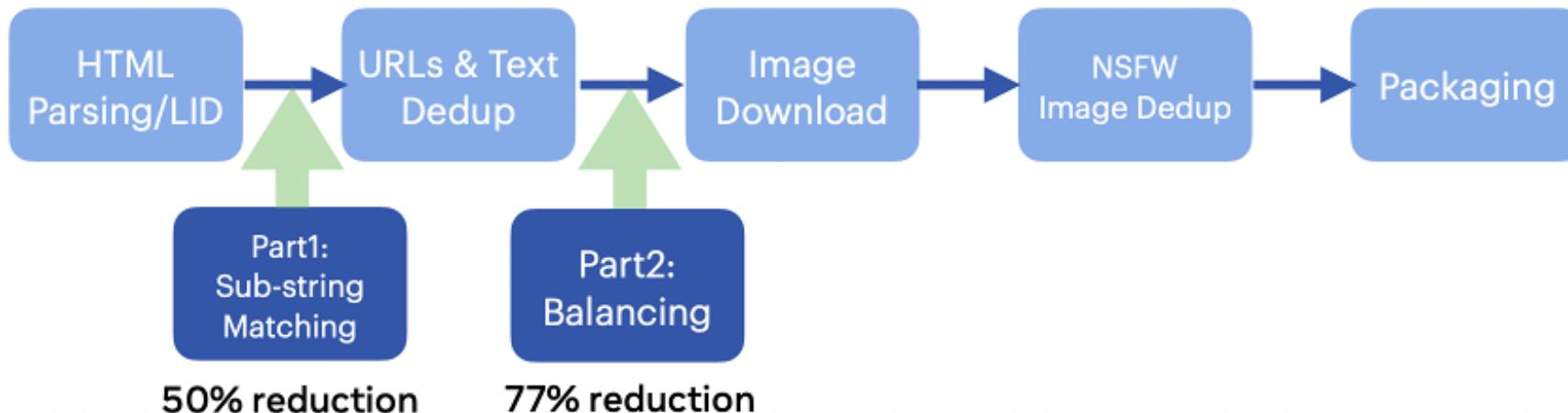
"An image with the face of Elon Musk with blonde hair"



双流编码器架构：CLIP及相关改进工作

MetaCLIP: Demystifying CLIP Data

- 证明CLIP的数据质量对模型表现至关重要
- OpenAI并未提供训练集和数据筛选算法，本文提出了CLIP可能采用的数据算法，简化和产生高质量的训练数据
- 创建元数据：使用WordNet和Wikipedia的高质量视觉概念（visual concept）
- 数据筛选算法：包括元数据字符串匹配和平衡数据分布，使得1.6B的原始数据被下采样成400M的训练数据，降低头部分布的冗余无效信息和头部数据的噪声（比如untitled photo），所有长尾分布的视觉概念全部保留



双流编码器架构：CLIP及相关改进工作

MetaCLIP: Demystifying CLIP Data

- 证明CLIP的数据质量对模型表现至关重要
- OpenAI并未提供训练集和数据筛选算法，本文提出了CLIP可能采用的数据算法，简化和产生高质量的训练数据
- 创建元数据：使用WordNet和Wikipedia的高质量视觉概念（visual concept）
- 数据筛选算法：包括元数据字符串匹配和平衡数据分布，使得1.6B的原始数据被下采样成400M的训练数据，降低头部分布的冗余无效信息和头部数据的噪声（比如untitled photo），所有长尾分布的视觉概念全部保留

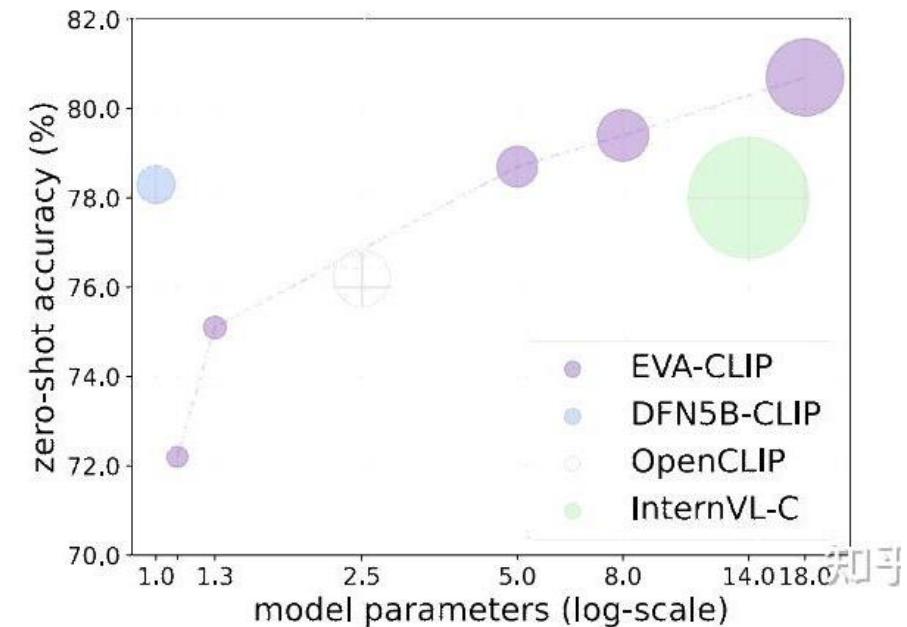
Average	ImageNet	Food-101	CIFAR10	CIFAR100	CUB	SUN397	Cars	Aircraft	DTD	Pets	Caltech-101	Flowers	MNIST	FER-2013	STL-10	EuroSAT	RESCISC45	GTSRB	KITTI	Country211	PCAM	UCF101	Kinetics700	CLEVR	HatefulMemes	SST2	
ViT-B/32																											
CLIP, our eval.	56.6	63.4	83.7	89.8	65.1	53.7	62.0	59.7	19.6	44.0	87.2	87.4	66.9	48.2	46.6	97.1	44.9	61.0	32.6	28.7	17.2	62.5	63.9	48.0	23.6	56.4	58.6
OpenCLIP, our eval.	57.6	62.9	80.7	90.7	70.6	61.2	66.4	79.2	16.7	54.5	86.5	90.7	66.1	37.4	48.2	95.6	52.2	58.0	42.0	38.0	14.8	50.1	63.0	42.8	22.5	53.3	52.3
MetaCLIP	58.2	65.5	80.6	91.3	70.2	63.4	63.0	70.7	26.8	52.8	88.7	91.9	68.5	41.5	35.9	95.4	52.6	64.2	35.8	30.7	17.2	55.5	66.1	45.4	30.6	56.4	53.4
ViT-B/16																											
CLIP, our eval.	59.6	68.3	88.8	90.8	68.2	55.6	64.0	64.6	24.0	45.1	88.9	89.1	69.4	51.8	53.0	98.2	54.8	65.5	43.3	21.7	22.8	56.3	68.5	52.3	25.5	58.7	60.5
OpenCLIP, our eval.	60.4	67.0	85.8	91.7	71.4	65.3	69.2	83.6	17.4	51.0	89.2	90.8	66.5	66.3	46.1	97.0	52.2	65.7	43.5	23.7	18.1	51.7	67.0	46.2	33.9	54.5	54.4
MetaCLIP	61.1	70.8	86.8	90.1	66.5	70.8	66.6	74.1	27.9	55.9	90.4	93.8	72.3	47.8	44.6	97.2	55.4	68.8	43.8	33.4	22.6	52.9	68.0	49.5	22.8	54.8	60.6
ViT-L/14																											
CLIP, our eval.	65.7	75.5	93.0	95.6	78.3	63.3	66.8	77.8	31.3	55.3	93.6	93.3	79.3	76.4	56.9	99.4	61.9	70.9	50.6	19.2	31.9	50.1	75.7	60.2	22.3	59.7	68.9
OpenCLIP, our eval.	64.5	72.7	90.0	94.7	78.0	73.9	72.4	89.5	24.7	60.2	91.6	93.6	73.0	76.1	54.3	98.1	63.9	69.6	49.9	16.0	23.0	51.7	71.5	51.6	25.4	55.3	56.0
MetaCLIP	67.1	76.2	90.7	95.5	77.4	75.9	70.5	84.7	40.4	62.0	93.7	94.4	76.4	61.7	46.5	99.3	59.7	71.9	47.5	29.9	30.9	70.1	75.5	57.1	35.1	56.6	65.6

双流编码器架构：CLIP及相关改进工作

EVA-CLIP-18B: Scaling CLIP to 18 Billion Parameters

- 用更大规模的数据和模型架构训练CLIP
- 原始的CLIP最大权重为ViT-L/14-336px，大约500M的参数量，模型越大是否表现越好？

- 模型规模：180亿参数量，迄今为止最强大的开源CLIP模型
- 训练样本量：该模型在观看了60亿个训练样本后取得显著性能。
- 性能表现：在27个广泛认可的图像分类基准测试中平均实现了80.7%的零样本顶级准确率。
- 性能饱和：**EVA-CLIP-18B未显示出性能饱和的迹象，意味着有进一步提高其性能的潜力，尤其在视觉模型的扩展方面。



双流编码-解码器架构：BLIP系列

BLIP: Bootstrapping Language-Image Pretraining

Bootstrapping (improve your situation by your own efforts) — 在线的数据标注和清理，把处理好的数据继续用来迭代原模型

salesforce在2022年提出的多模态框架，是**理解和生成的统一**，引入了跨模态的编码器和解码器，实现了跨模态信息流动，在多项视觉和语言任务取得SOTA。在AIGC中通常用来给图像生成prompt，好的prompt对交叉注意力的微调非常关键，例如ControlNet中的Automatic Prompt就是BLIP生成的。

Image Description Generation

Try the provided image or upload your own:
Drag and drop file here
Limit 200MB per file

Image

Generate

Description

the golden gate bridge is illuminated with bright lights in blue twilight sky

Zero-shot Classification

Image

Categories

category 1	category 2
merlion	elephant
category 3	category 4
giraffe	fountain
category 5	category 6
marina bay	singapore river

Submit

Prediction

Category	Probability (%)
merlion	60.78%
elephant	0.00%
giraffe	0.00%
fountain	8.77%
marina bay	28.06%
singapore river	2.38%

Text Localization

Person playing with dog on the beach.

Submit

Image

GradCam

person playing with dog

Visual Question Answering

Image

Question

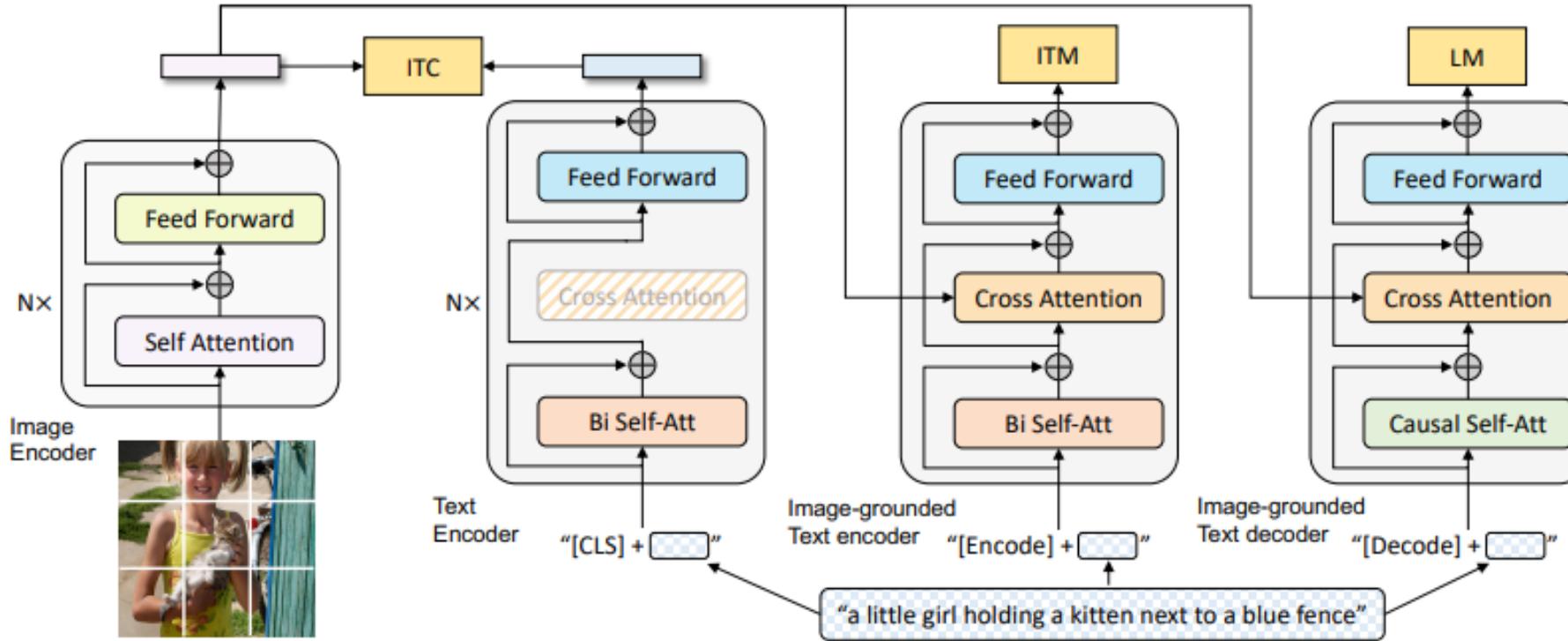
Input your question!
What is the dog's breed?

Answer

golden retriever

双流编码-解码器架构：BLIP系列

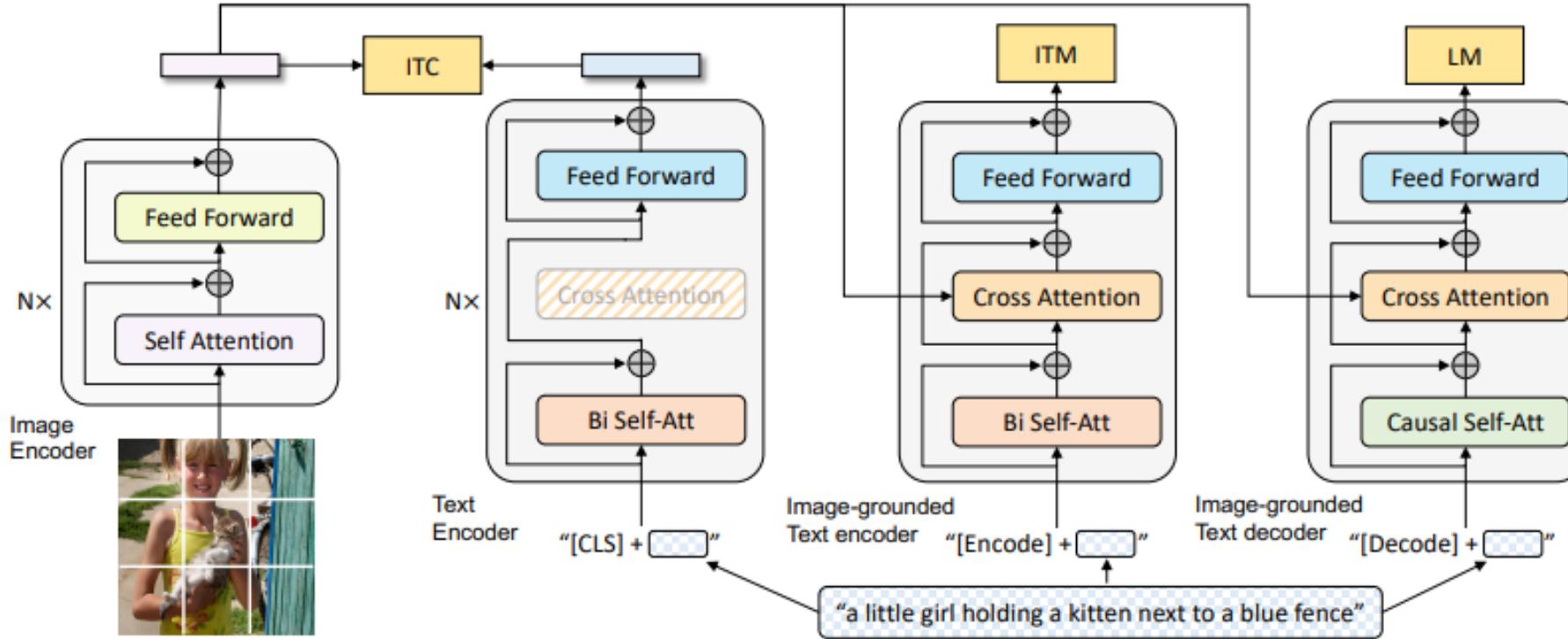
BLIP: Bootstrapping Language-Image Pretraining



- BLIP引入了编码器-解码器的多模态混合结构MED (Multimodal mixture of Encoder-Decoder)
- 能够有效地进行多任务预学习和迁移学习。MED包括两个单模态编码器 (Image Encoder, Text Encoder) , 一个以图像为基础的编码器 (image-grounded text encoder) 和一个以图像为基础的解码器 (image-grounded text decoder)

双流编码-解码器架构：BLIP系列

BLIP: Bootstrapping Language-Image Pretraining

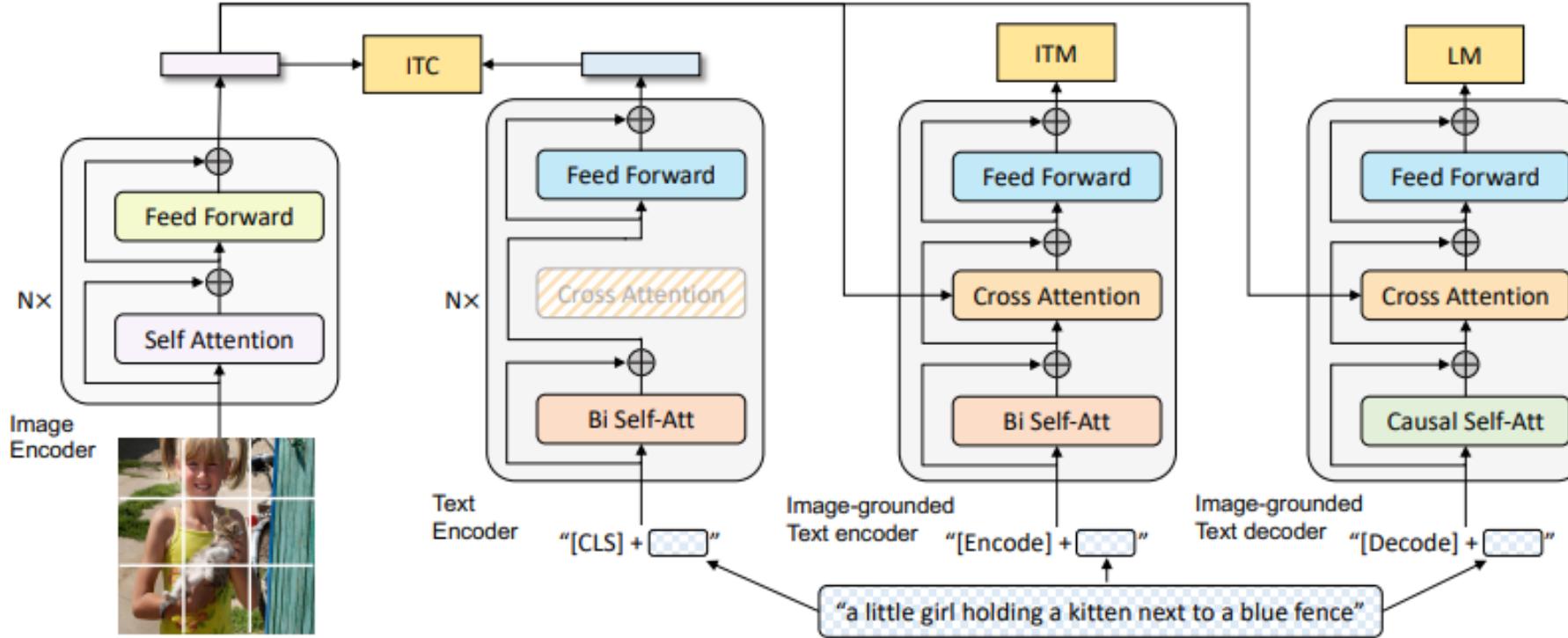


- **训练目标：通过三个损失函数联合进行预训练：**

1. 图像-文本对比损失 ITC (Image-Text Contrastive Loss)：针对图像编码器和文本编码器，通过正负图文对的对比学习，来对齐图像和文本的潜在特征空间。

双流编码-解码器架构：BLIP系列

BLIP: Bootstrapping Language-Image Pretraining

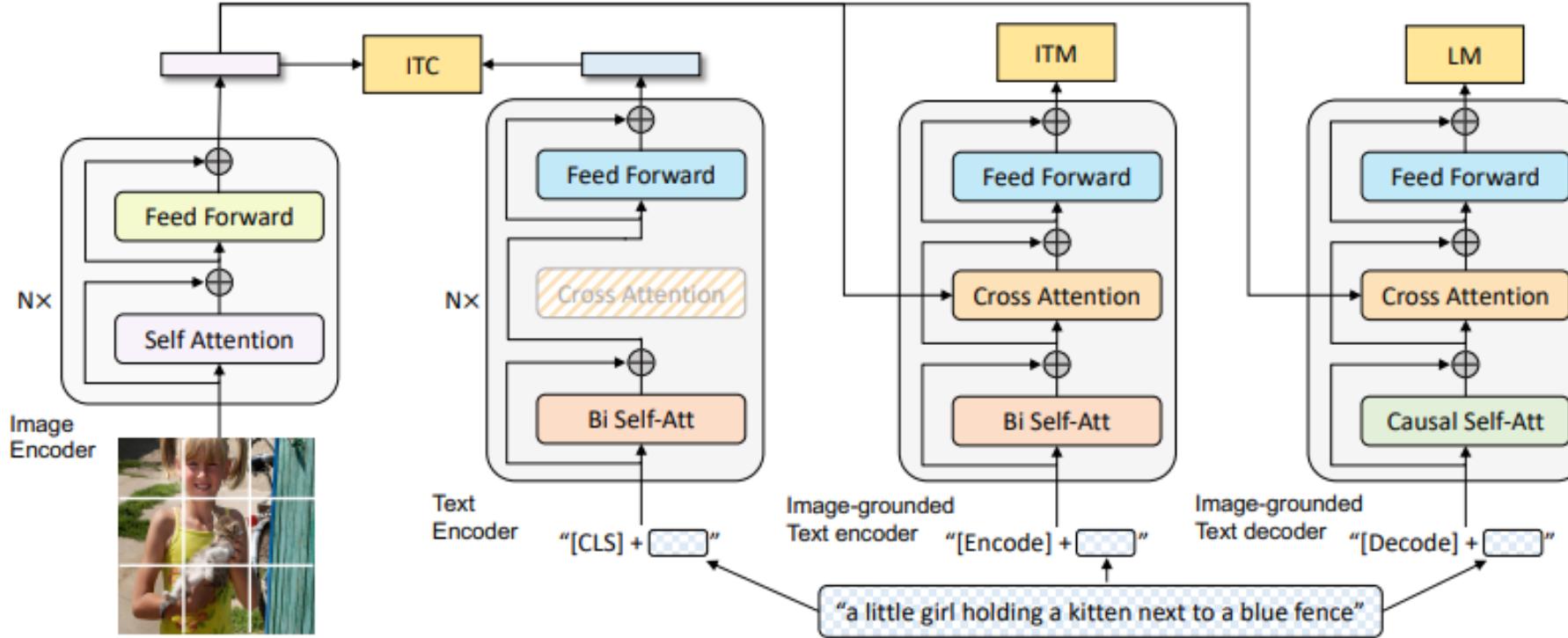


- **训练目标：通过三个损失函数联合进行预训练：**

2. 图像-文本匹配损失 ITM (Image-Text Matching Loss) : 针对以图像为基础的文本编码器，通过对图文匹配性进行二分类，建模图文多模态信息的相关性。

双流编码-解码器架构：BLIP系列

BLIP: Bootstrapping Language-Image Pretraining

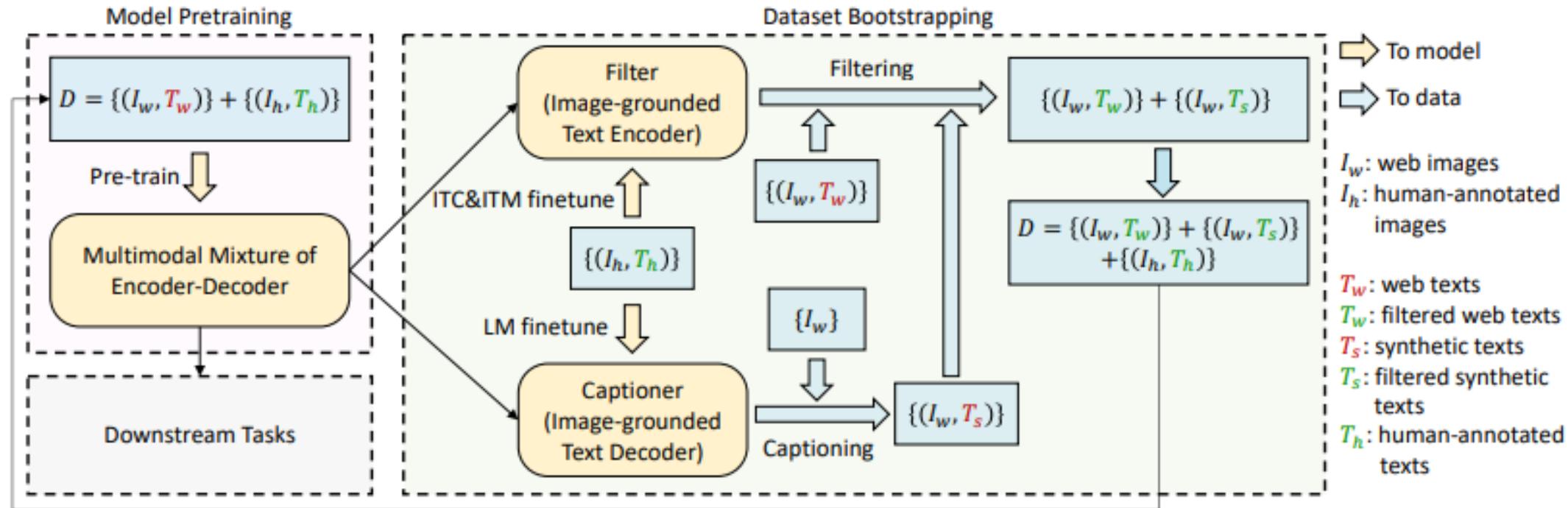


- **训练目标：通过三个损失函数联合进行预训练：**

3. 语言建模损失 LM (Language Modeling Loss) : 针对以图像为基础的文本解码器，通过交叉熵损失进行优化，训练模型以自回归的方式生成目标caption。

双流编码-解码器架构：BLIP系列

BLIP: Bootstrapping Language-Image Pretraining

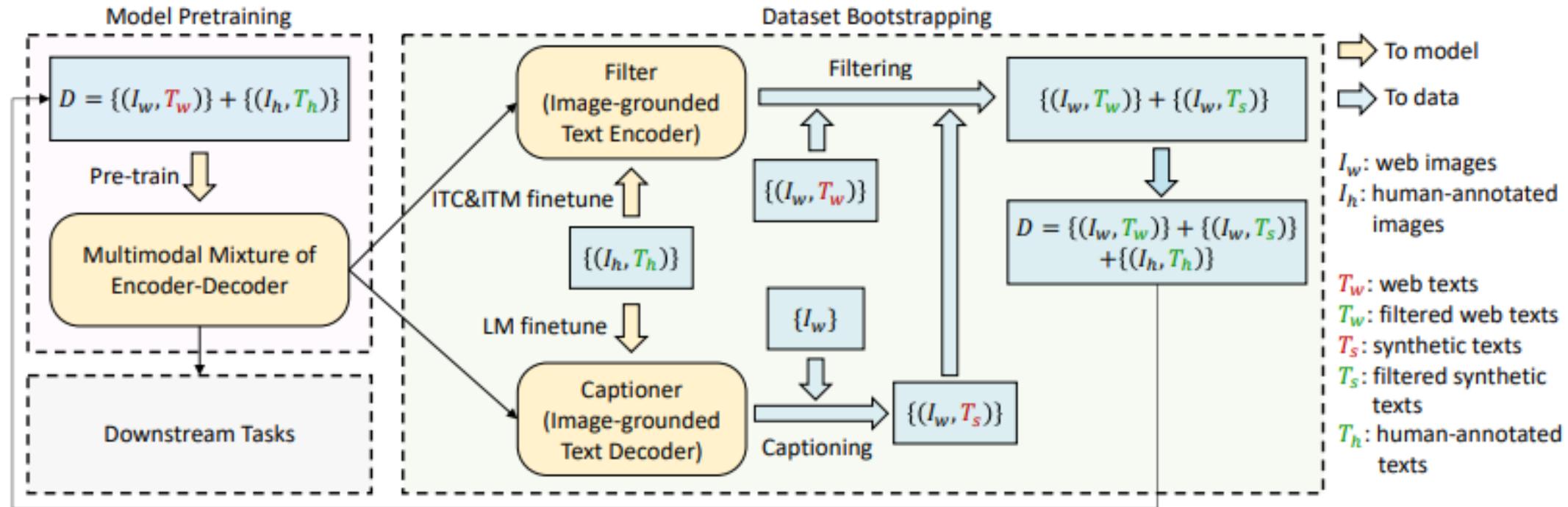


- 训练方法：Bootstrapping的方式获取更好的训练样本

网络上获得的图文对通常包含许多不准确甚至错误的信息，为了有效利用这种形态的数据，BLIP提出caption生成和过滤模块CapFilt (Captioning and Filtering)，首先从噪声图文对中学习，然后生成和过滤产生新的数据集，再去迭代优化原模型。

双流编码-解码器架构：BLIP系列

BLIP: Bootstrapping Language-Image Pretraining



- 训练方法：Bootstrapping的方式在训练的同时获取更好的训练样本

CapFilt包含两个模块：一个是captioner，给网络图像生成caption，另一个是Filter，过滤原始网络文本和合成文本中的噪声caption。这一过程十分类似于GPT中引入的Reward model，通过给予反馈不断提升模型的性能



北京航空航天大學
BEIHANG UNIVERSITY

人工智能研究院
Institute of Artificial Intelligence

Intro to Vision Large-Language Models (VLLMs)

VLLMs 和 VLP 的区别与关联

区别：

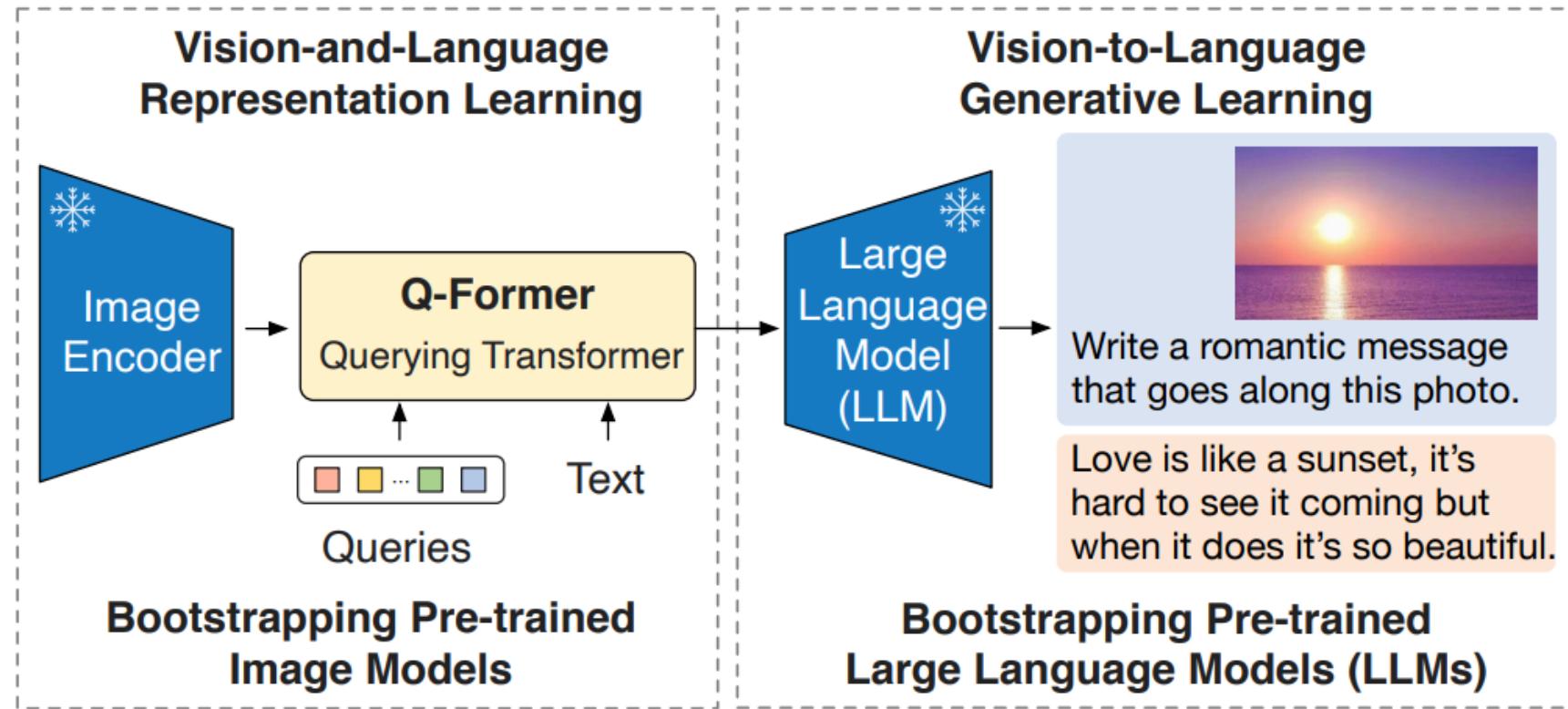
- VLP：专注于如何通过结合视觉和语言信息来预训练模型。以生成可以广泛应用于多种下游任务的通用特征和表示。
- VLLMs：更广泛地指代任何结合视觉和语言能力的大型模型，不限于预训练策略。
VLLMs可以是专门为特定任务设计或优化的，也可以是基于VLP技术进一步发展的模型。

关联：

- VLLM 可能包括VLP作为其组成之一，更具体而言，VLP是实现VLLMs的一种技术途径和重要组成模块（常作为基础的特征提取 / 解码器模块）
- 随着研究的深入和技术的发展，VLP可以被视为是VLLMs发展的一个分支或初级阶段，VLLMs则代表了在视觉语言模型领域内更广泛和深入应用的发展方向。

使用Q-former连接图像编码器和LLMs: BLIP-2

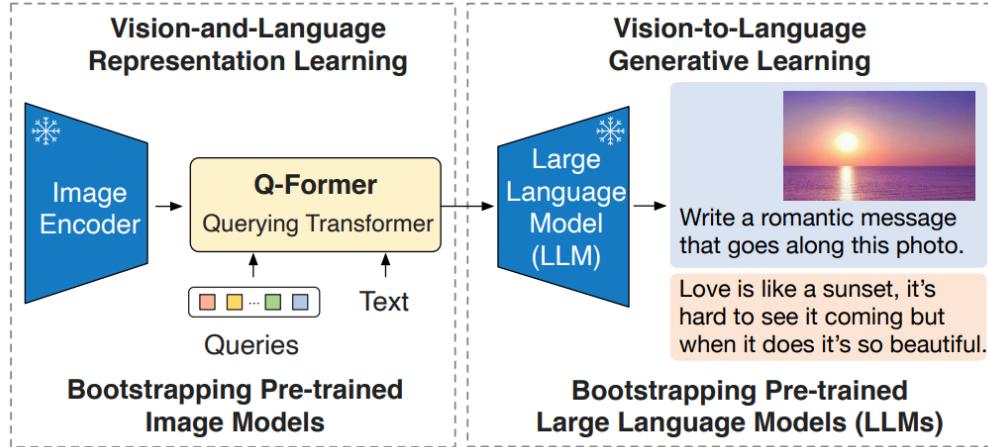
BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and LLMs



Salesforce在2023年提出 BLIP-2，通过利用预训练的视觉模型和语言模型来提升多模态效果和降低训练成本，预训练的视觉模型能够提供高质量的视觉表征，预训练的语言模型则提供了强大的语言生成能力。

使用Q-former连接图像编码器和LLMs: BLIP-2

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and LLMs



```
# pretrained image encoder
self.vision_model = Blip2VisionModel(config.vision_config)
# input query
self.query_tokens = nn.Parameter(torch.zeros(1, config.num_query_tokens, config.qformer_config.hidden_size))
self.qformer = Blip2QFormerModel(config.qformer_config)

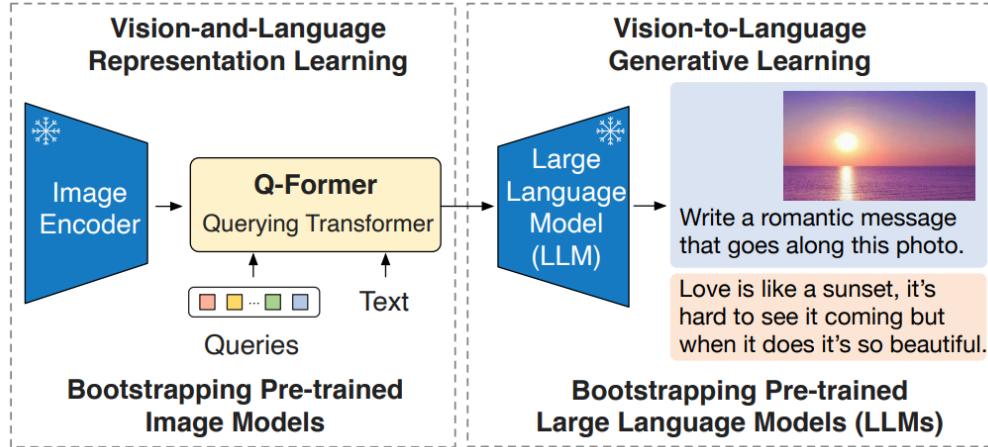
self.language_projection = nn.Linear(config.qformer_config.hidden_size, config.text_config.hidden_size)
if config.use_decoder_only_language_model:
    language_model = AutoModelForCausalLM.from_config(config.text_config)
else:
    language_model = AutoModelForSeq2SeqLM.from_config(config.text_config)
# pretrained Large Language model
self.language_model = language_model
```

模型架构: BLIP-2 由预训练的Image Encoder, 预训练的Large Language Model, 和一个可学习的 Q-Former 组成

- Image Encoder: 负责从输入图片中提取视觉特征, 本文试验了两种网络结构, CLIP 训练的 ViT-L/14 和 EVA-CLIP 训练的 ViT-g/14
- Large Language Model: 负责文本生成, 本文试验了 decoder-based LLM and encoder-decoder-based LLM
- **Q-Former**: 负责弥合视觉和语言两种模态的差距, 由 Image Transformer 和 Text Transformer 两个子模块构成, 它们**共享相同自注意力层**。

使用Q-former连接图像编码器和LLMs: BLIP-2

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and LLMs



```
# pretrained image encoder
self.vision_model = Blip2VisionModel(config.vision_config)
# input query
self.query_tokens = nn.Parameter(torch.zeros(1, config.num_query_tokens, config.qformer_config.hidden_size))
self.qformer = Blip2QFormerModel(config.qformer_config)

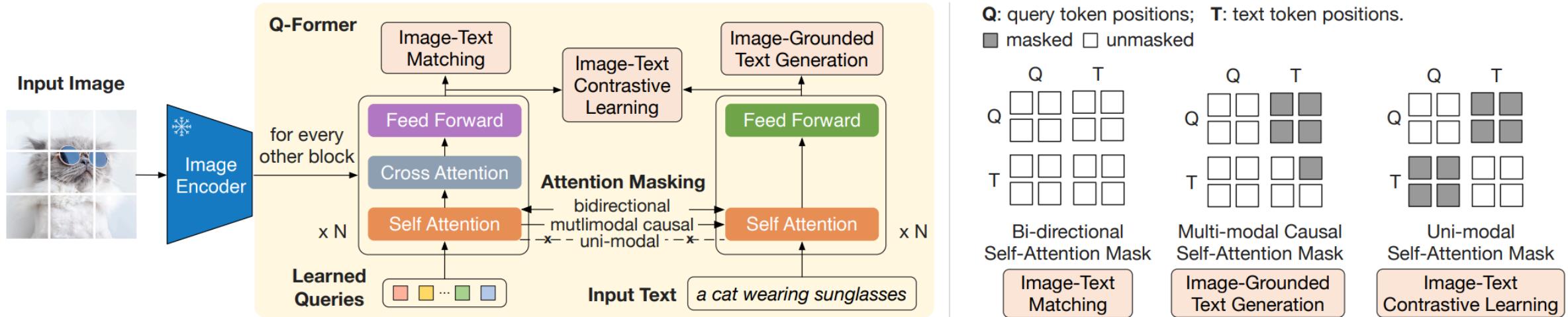
self.language_projection = nn.Linear(config.qformer_config.hidden_size, config.text_config.hidden_size)
if config.use_decoder_only_language_model:
    language_model = AutoModelForCausalLM.from_config(config.text_config)
else:
    language_model = AutoModelForSeq2SeqLM.from_config(config.text_config)
# pretrained Large Language model
self.language_model = language_model
```

模型架构: BLIP-2 由预训练的Image Encoder, 预训练的Large Language Model, 和一个可学习的 Q-Former 组成

- **Q-Former:** 负责弥合视觉和语言两种模态的差距, 由**Image Transformer**和**Text Transformer**两个子模块构成
- **Image Transformer**通过与**图像编码器**进行交互提取视觉特征, 它的输入是**可学习的 Query**, 这些Query通过**自注意力层**相互交互, 并通过**交叉注意力层与冻结的图像特征交互**, 还可以通过**共享的自注意力层与文本进行交互**。
- **Text Transformer**作为文本编码器和解码器, 它的自注意力层与**Image Transformer**共享, 根据预训练任务, 应用不同的自注意力掩码来控制Query和文本的交互方式。

使用Q-former连接图像编码器和LLMs: BLIP-2

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and LLMs

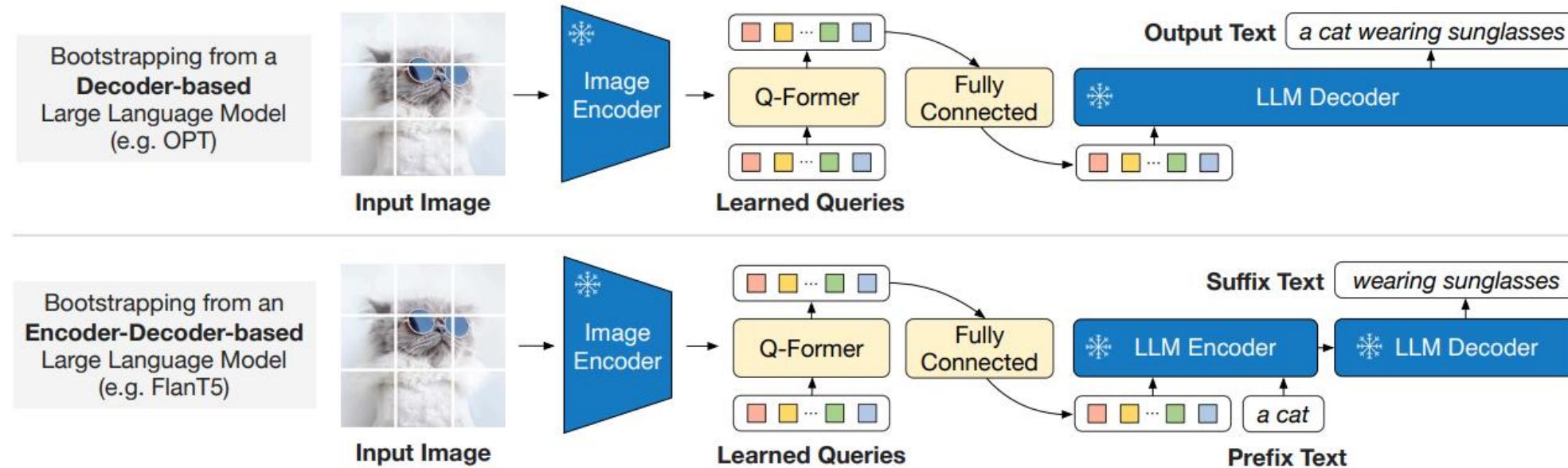


训练范式：两阶段训练Q-former—表示学习阶段和生成学习阶段

- 将 Q-Former 连接到冻结的 Image Encoder，训练集为图像-文本对，通过联合优化三个预训练目标，在Query和Text之间分别采用不同的注意力掩码策略，从而控制Image Transformer和Text Transformer的交互方式。
- 目标1：ITC(Image-Text Contrastive Learning) 对齐图像嵌入和文本嵌入
- 目标2：ITG(Image-grounded Text Generation) 给定输入图像作为条件的情况下，训练 Q-Former 生成文本。
- 目标3：ITM(Image-Text Matching) 预测图像-文本对是正匹配还是负匹配，学习图像和文本表示之间的细粒度对齐

使用Q-former连接图像编码器和LLMs: BLIP-2

BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and LLMs

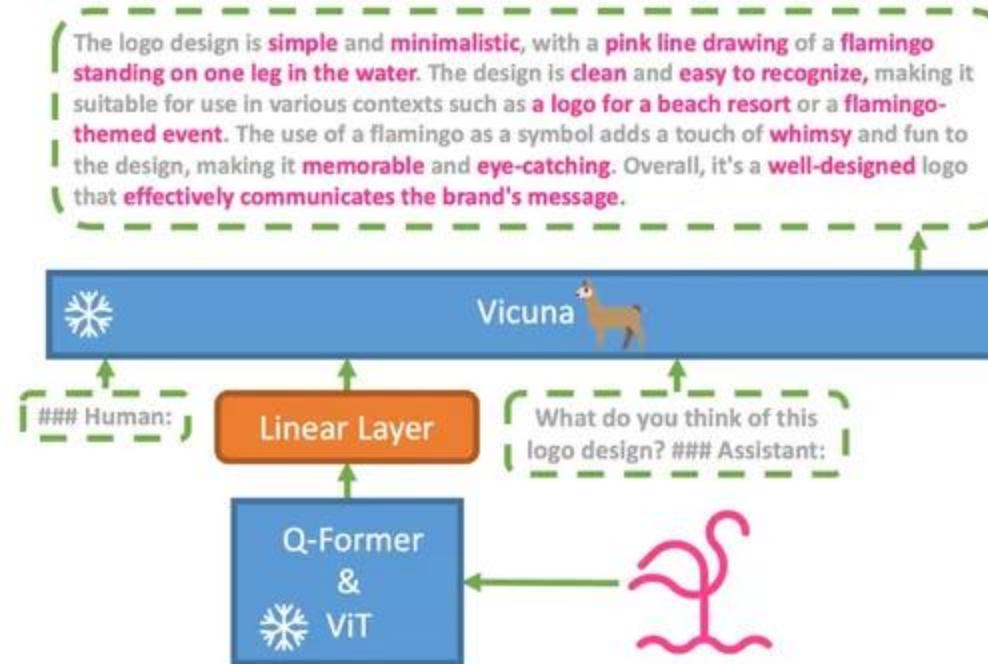


训练范式: 两阶段训练Q-former—表示学习阶段和生成学习阶段

- 将 Q-Former 连接到冻结的 LLM，以利用 LLM 的语言生成能力。使用全连接层将输出的 Query 嵌入线性投影到与 LLM 的文本嵌入相同的维度，然后将投影的 Query 嵌入添加到输入文本嵌入前面
- 由于 Q-Former 已经过预训练，可以提取包含语言信息的视觉表示，因此它可以有效地充当信息瓶颈，将最有用的信息提供给 LLM，同时删除不相关的视觉信息，减轻了 LLM 学习视觉语言对齐的负担。

使用投影层链接视觉模型和LLMs: MiniGPT4系列

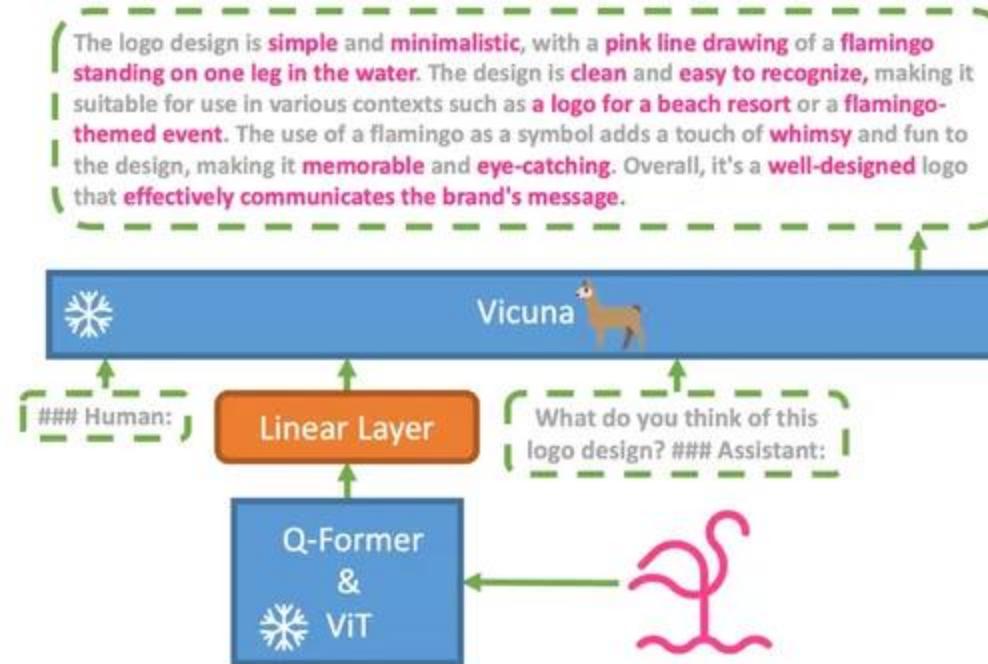
MiniGPT-4: Enhancing Vision-language Understanding with Advanced Large Language Models



- 和BLIP-2的思想类似， MiniGPT4认为增强多模态模型的生成能力一定要利用**大语言模型的知识**。因此也是采用**“对齐”** 的思路来训练一个投影层将预训练好的LLM和视觉模型链接在一起。这也是当前VLLMs的主要实现途径
- 简单概括：通过投影层来将图片的特征转化为LLM能够理解的token，剩下的就和LLM做的一样了（进行视觉语言的自回归训练），这也是猜测GPT-4V所可能采用的范式

使用投影层链接视觉模型和LLMs: MiniGPT4系列

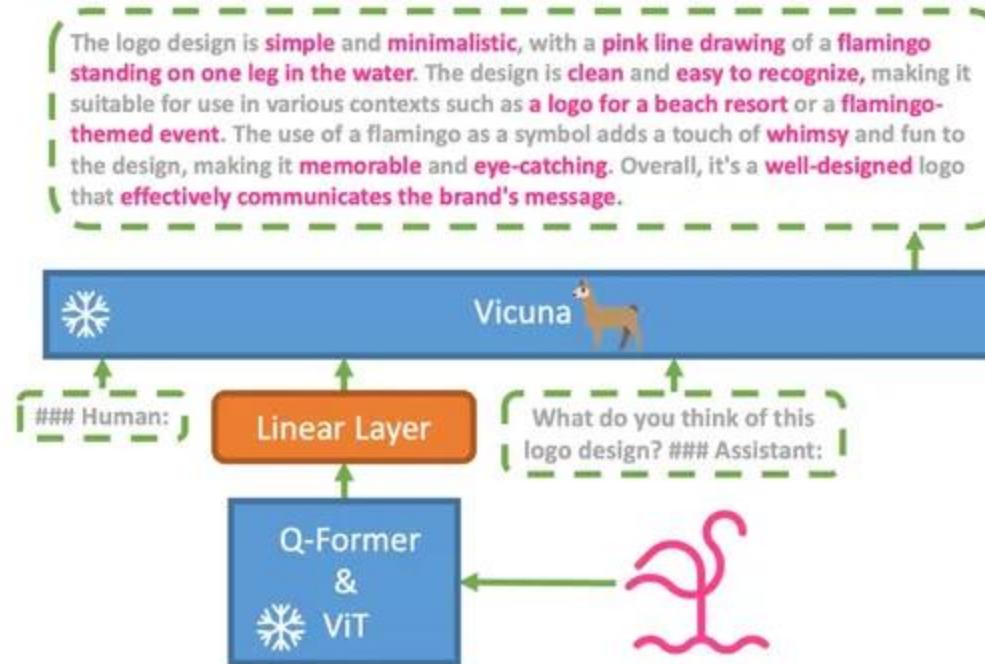
MiniGPT-4: Enhancing Vision-language Understanding with Advanced Large Language Models



- image encode采用的是预训练的ViT和QFormer
- LLM采用的是Vicunna
- 对齐层采用的是简单的线性层

使用投影层链接视觉模型和LLMs: MiniGPT4系列

MiniGPT-4: Enhancing Vision-language Understanding with Advanced Large Language Models



- **训练范式: 两阶段训练 (多模态大多用多阶段训练的思路)**
- 第一阶段: 使用大规模图文对数据, 优化投影层的参数, 将image encoder提取的图片特征转变成LLM能够理解的形式
- 第二阶段: 指令集数据集微调, 优化模型生成文本的质量

使用投影层链接视觉模型和LLMs: MiniGPT4系列

MiniGPT-4: Enhancing Vision-language Understanding with Advanced Large Language Models

- 如何制作指令集数据：取决于希望模型解决什么类型的任务
- 1. Conceptual Caption数据集随机挑选5000图片。用stage1的模型来生成详细的图片描述。所用的prompt如下：
####Human: <ImageFeature>Describe this image in detail. Give as many details as possible. Say everything you see. ####Assistant:
- 为了避免生成caption答非所问和语法错误的情况，作者对生成的每一个图文对进行了人工check，以此保证数据集的质量。清洗后获得了总计3500个图文对。

虽然作者主要构建了image caption的数据。我们训练自己的多模态模型的时候，可以根据需要构建自己的instruction数据集。

【例如：open set检测】

prompt: ####Human: <ImageFeature> . 请返回图中所有任务的坐标，返回
格式以json输出，模版如：{person1: [xmin, ymin, xmax, ymax]},其中坐标做了归一化处理
####Assistant:

label: {person1: [0.14, 0.24, 0.45, 0.65], person2: [0.11, 0.]...}

【例如：表格信息抽取】

prompt: ####Human: <ImageFeature> . 请抽取表格中的所有3元组

####Assistant:

label: [(分类1, 分类2, value), (分类1, 分类2, value), ...] 总结来说就是把任务的统一建模成sequence-to-sequence的架构。这样就能联合起来一起训练。相当于同一种统一的架构做多任务学习。

Visual Instruction Tuning

LLaVA

Haotian Liu^{1*}, Chunyuan Li^{2*}, Qingyang Wu³, Yong Jae Lee¹

¹University of Wisconsin-Madison ²Microsoft Research ³Columbia University
<https://llava-vl.github.io>

Improved Baselines with Visual Instruction Tuning

LLaVA-1.5

Haotian Liu¹ Chunyuan Li² Yuheng Li¹ Yong Jae Lee¹
¹University of Wisconsin-Madison ²Microsoft Research, Redmond
<https://llava-vl.github.io>

LLAVA-PLUS: LEARNING TO USE TOOLS FOR CREATING MULTIMODAL AGENTS

LLaVA-PLUS

Shilong Liu^{♣*}, Hao Cheng[♦], Haotian Liu^{◇*}, Hao Zhang^{♡*}, Feng Li^{♡*},
Tianhe Ren[◆], Xueyan Zou^{◇*}, Jianwei Yang[♦], Hang Su[♣], Jun Zhu[♣],
Lei Zhang[◆], Jianfeng Gao[♦], Chunyuan Li[♦] 

[♣]Dept. of Comp. Sci. & Tech., Institute for AI, BNRIst, Tsinghua University

[♦]Microsoft Research, Redmond

[◇]University of Wisconsin-Madison [♡]HKUST [◆]IDEA Research

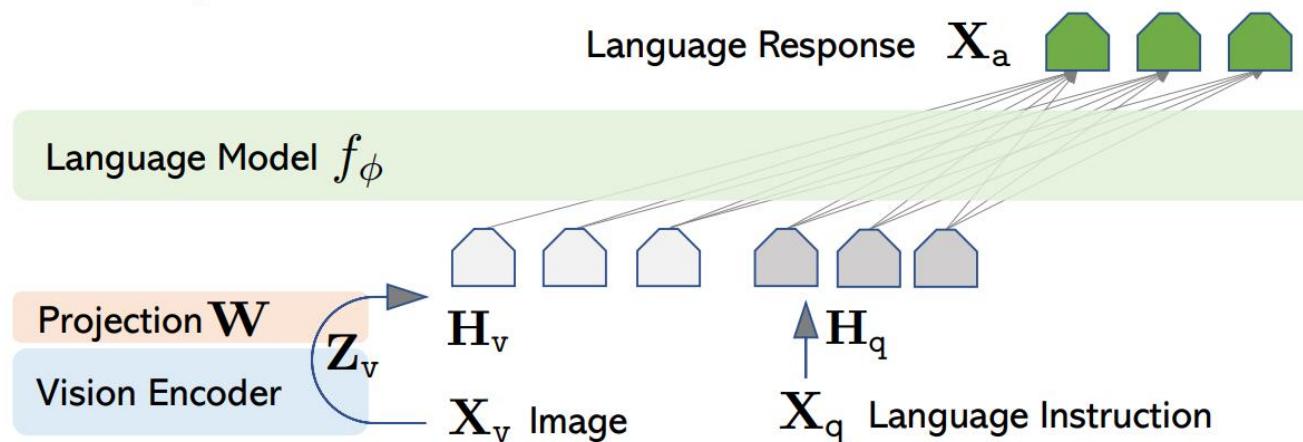
*Work performed during an internship at Microsoft  Project Lead

- LLaVA(Large Language and Vision Assistant), 即大型语言和视觉助手，是一个端到端训练的大型多模态模型，将视觉编码器和大语言模型连接起来实现通用的视觉和语言理解。当前开源社区中应用较广的VLLMs

视觉指令微调：LLaVa系列

LLaVa: Visual Instruction Tuning

- LLaVa的主要贡献
- **多模态指令跟随数据集：**使用 ChatGPT/GPT-4 将图像文本对转换为适当的指令遵循数据格式，生成三种类型的指令跟随数据：对话、详细描述、复杂推理。总共收集了 158K 个语言图像指令跟随数据样本，其中对话样本 58K，详细描述样本 23K，复杂推理样本 77K。
- 将 CLIP 的视觉编码器与LLaMA语言解码器连接起来，并在生成的视觉语言指令跟随数据集上对它们进行端到端的微调。
- 开源：向公众发布生成的多模态指令数据、用于数据生成和模型训练的代码库、模型权重文件和可视化聊天demo。



A summary

- 包括LLaVa在内，目前主流的VLLMs大多属于遵循指令的大型多模态模型（Instruction-following LMM），常见架构包括用于编码视觉特征的**预训练视觉主干网络**、用于理解用户指令并产生响应的**预训练大语言模型 (LLM)**，以及用于将视觉编码器输出与语言模型对齐的**视觉语言跨模态连接器**。训练此类模型通常遵循两阶段的方法：
 - 视觉语言对齐预训练阶段：**利用图像文本对将视觉特征与语言模型的词嵌入空间对齐。
 - 视觉指令调整阶段：**根据视觉指令调整模型，使模型能够遵循用户的不同指令要求。

Discovering unusual content



Describe the image.

The image shows a cactus plant standing in the middle of a frozen lake. The cactus is surrounded by large ice crystals that have formed on the surface of the water. In the background, there is a range of mountains with snow-capped peaks. The sky is clear and blue, with a few fluffy clouds floating in the distance. The overall mood of the image is peaceful and serene, with the cactus standing tall and proud in the center of the frozen lake.

Can this happen in real world?

No,





北京航空航天大學
BEIHANG UNIVERSITY

人工智能研究院
Institute of Artificial Intelligence

Intro to Other Multimodal Large-Language Models



AN EMBODIED GENERALIST AGENT IN 3D WORLD

**Jiangyong Huang^{1,2*}, Silong Yong^{1,3*}, Xiaojian Ma^{1*}, Xiongkun Linghu^{1*}, Puhaao Li^{1,4},
Yan Wang¹, Qing Li¹, Song-Chun Zhu^{1,2,4}, Baoxiong Jia¹, Siyuan Huang¹**

¹Beijing Institute for General Artificial Intelligence (BIGAI)

²Peking University ³Carnegie Mellon University ⁴Tsinghua University

<https://embodied-generalist.github.io>

(ICLR2024)

首个3D 具身 Agents



Motivation

Describe object.

There is a rectangular brown table. It is in the center of the room.

Leo

Where is the bathtub located in the bathroom?

To right of toilet

Leo

Is the door closed or open to my left side?

Closed

Leo

- **任务：构建具备在3D物理世界中感知、推理、规划、行动的Embodied Agents**
- **相关工作：**
- 目前的Embodied Agents框架（基于LLM和VLM进行任务规划、环境感知，并对齐下层的行动控制）取得了巨大进步，但他们的能力都局限于2D领域内，限制了其在3D真实场景下的应用，因此本文解决的核心问题是：**如何让Embodied Agents全面的理解3D环境并具备与真实3D世界交互的能力？**
- **挑战：合适的数据集 / 统一的模型框架 / 设计合理的优化目标**



Method detail

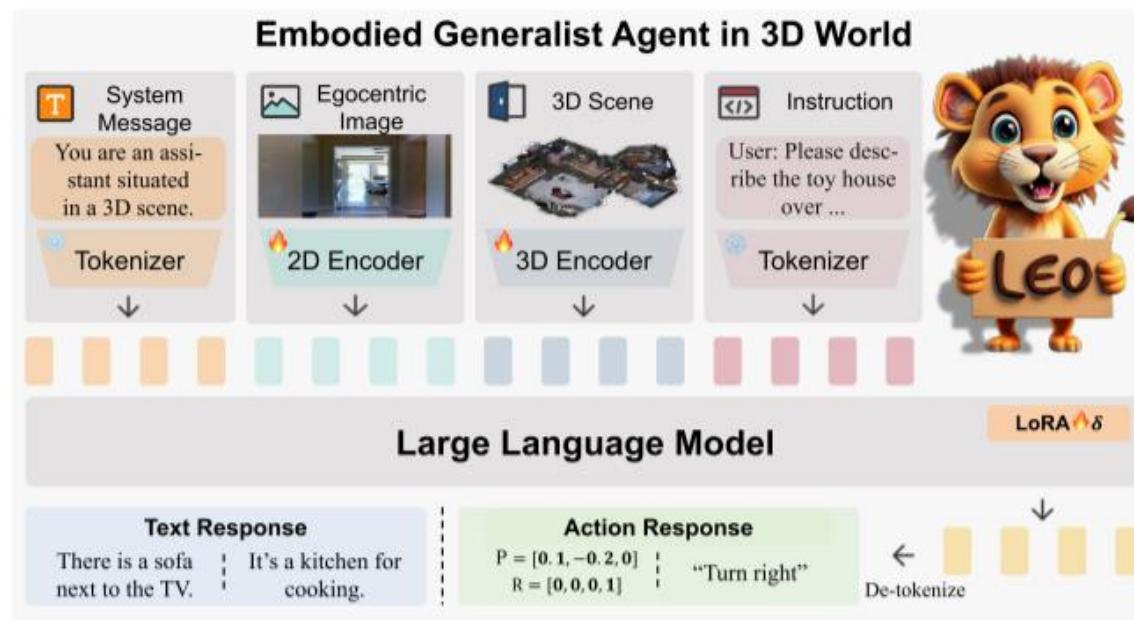
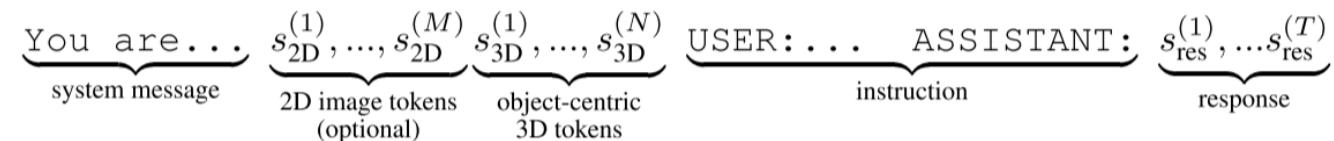
- 挑战：合适的数据集 / 统一的模型框架 / 设计合理的优化目标
- Overview: 将物体中心图像、3D点云场景、文本作为输入，将各种3D任务指定为自回归序列进行学习





Method detail

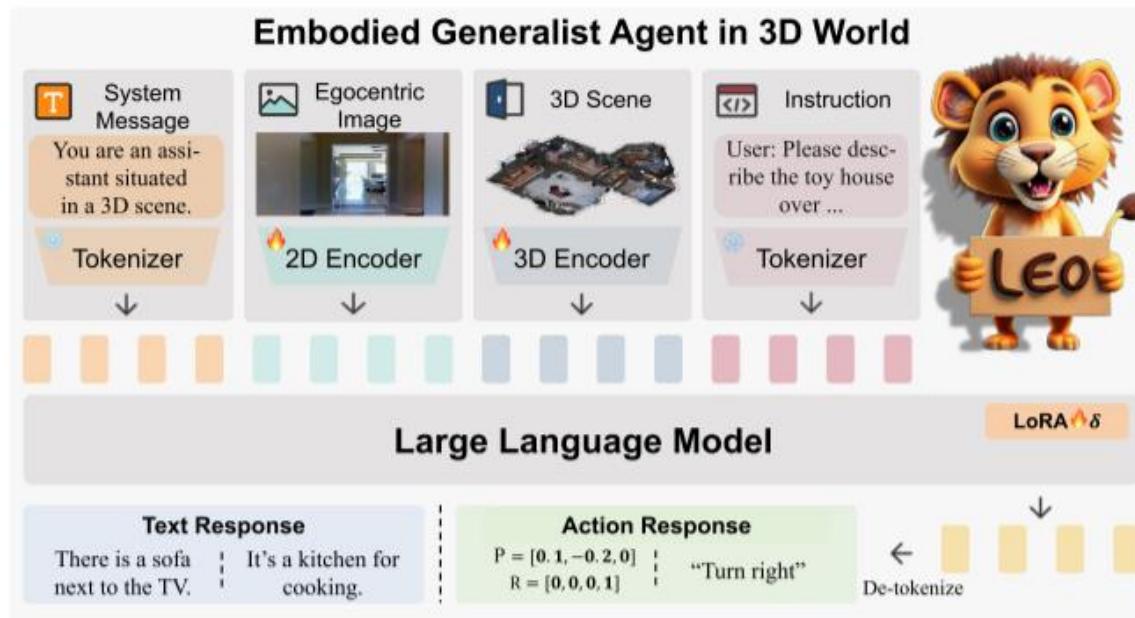
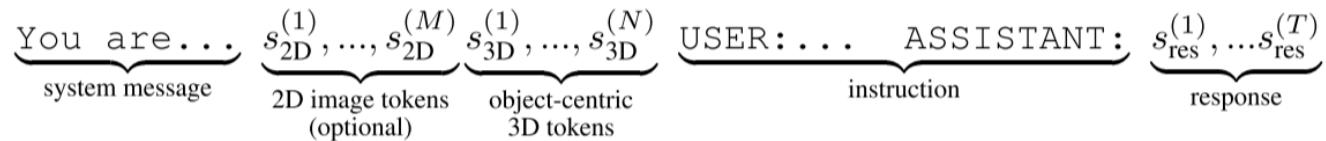
- **Model:** LEO的设计原则有两个方面，1. 能够处理自我为中心的2D信息、全局3D信息、以及文本指令，并具有文本输出和具身动作指令的统一架构 2. 应该利用预训练LLM作为先验
- 因此，将不同模态的数据转化为tokens序列，使用LLM进行处理，并采用类似GPT的自回归建模





Method detail

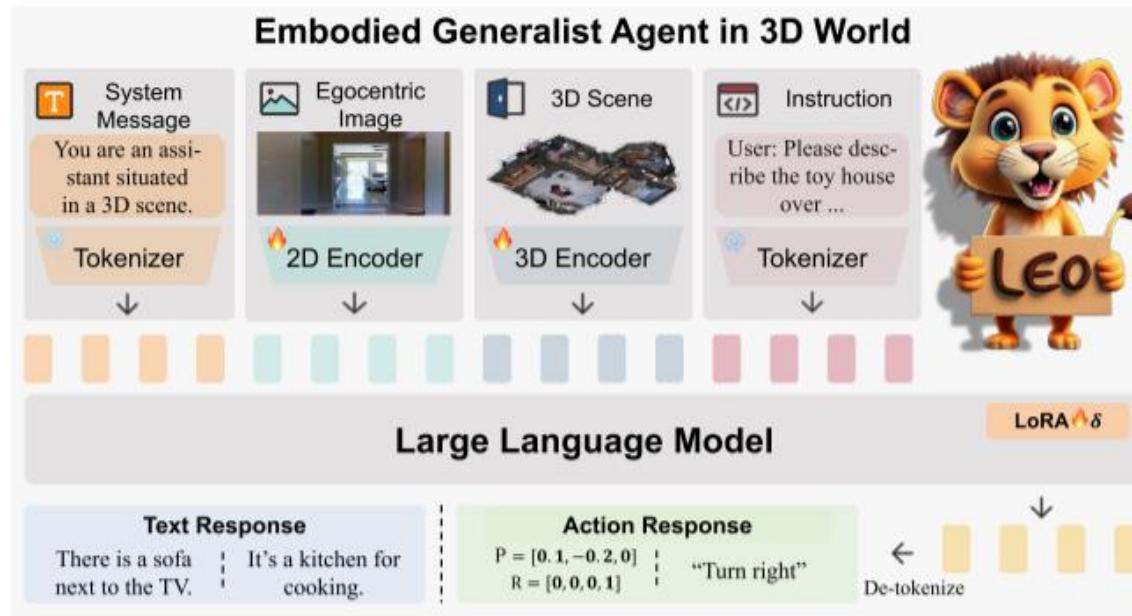
- **Tokenization (编码) :** 使用2D VLM和3D VLM编码多模态的数据，使用SentencePiece tokenizer进行文本编码。对于embodied action 指令，将连续动作离散化并映射到SentencePiece tokenizer中最少被使用的tokens，所有的数据被tokenization后，连接为式子中的形式





Method detail

- **Training:** 使用Vicuna-7B作为预训练的LLM处理token序列，使用LoRA的方式微调（仅使用少量额外的可学习参数，并引入冻结的LLM）
- **LoRA:** 低秩适应(low-rank adaptation), (Aghajanyan et. al.) 预训练大模型通常具有极小的内在维度，即存在一个极低维度的参数，微调它和在全参数空间中微调能起到相同的效果。这也是大模型具备few-shot能力的原因之一
- **训练目标:** 前缀语言建模，即给定前缀，预测后缀（回答或动作指令的输出）的token

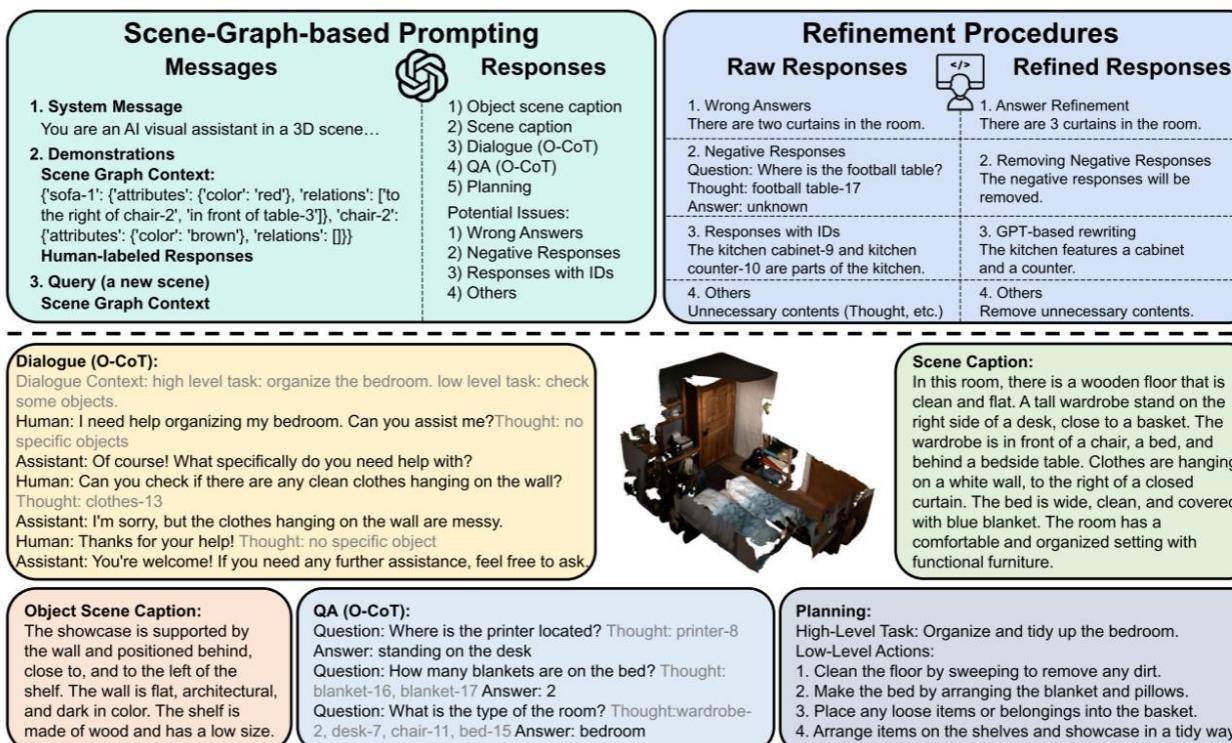


$$\mathcal{L}(\theta, \mathcal{B}) = - \sum_{b=1}^{|\mathcal{B}|} \sum_{t=1}^T \log p_\theta(s_{\text{res}}^{(b,t)} | s_{\text{res}}^{(b,<t)}, s_{\text{prefix}}^{(b,1)}, \dots, s_{\text{prefix}}^{(b,L)}),$$



Method detail

- Datasets:** 使用LLM辅助生成数据，训练数据分为两组（LEO-align和LEO-instruct），分别用于LEO的两阶段训练，即3D视觉-语言对齐和3D世界任务的Instruction Tuning
- 辅助数据生成的过程，首先生成带有3D场景的信息，输入包括系统信息，场景上下文关系，输出包括caption/对话/QA/规划等形式，接着进行人工纠错和修改提升数据质量





Experiment

在3D理解和具身推理、机器人操作、物体跟踪任务取得SoTA

Table 2: Quantitative comparison with state-of-the-art models on 3D VL understanding and embodied reasoning tasks. “C” stands for “CIDEr”, “B-4” for “BLEU-4”, “M” for “METEOR”, “R” for “ROUGE”, “Sim” for sentence similarity, and “EM@1” for top-1 exact match. The n-gram metrics for Scan2Cap are governed by IoU@0.5. [†] indicates answering questions via prompting GPT-3 with the generated scene caption.

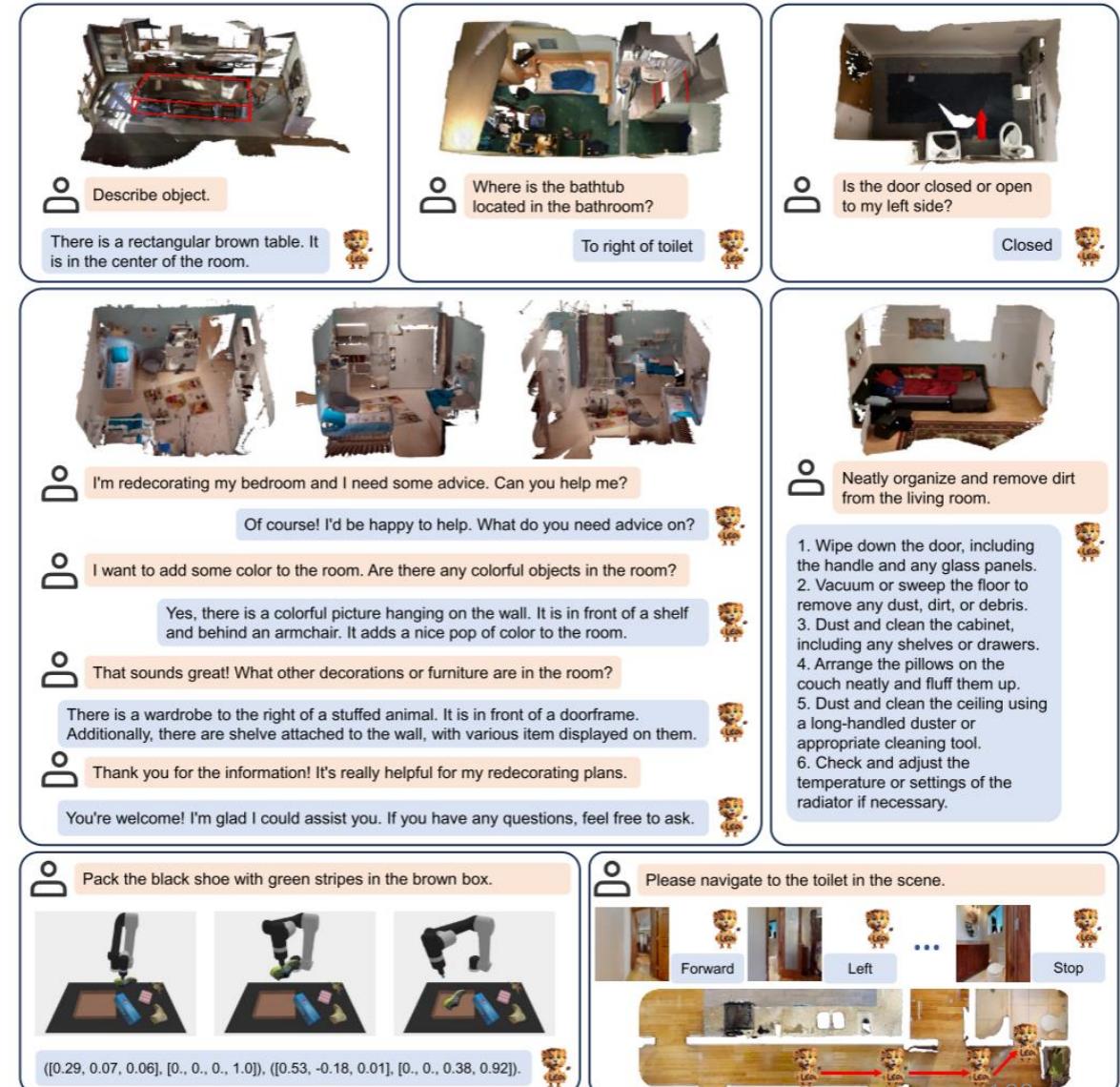
	Scan2Cap (val)					ScanQA (val)					SQA3D
	C	B-4	M	R	Sim	C	B-4	M	R	EM@1	EM@1
<i>Task-specific models</i>											
Scan2Cap (GPT-3) (Chen et al., 2021)	35.2	22.4	21.4	43.5	-	-	-	-	-	-	41.0 [†]
3DJCG (Cai et al., 2022)	47.7	31.5	24.3	51.8	-	-	-	-	-	-	-
Vote2Cap-DETR (Chen et al., 2023)	61.8	34.5	26.2	54.4	-	-	-	-	-	-	-
ScanRefer+MCAN (Chen et al., 2020)	-	-	-	-	-	55.4	7.9	11.5	30.0	18.6	-
ClipBERT (Lei et al., 2021)	-	-	-	-	-	-	-	-	-	-	43.3
ScanQA (Azuma et al., 2022)	-	-	-	-	-	64.9	10.1	13.1	33.3	21.1	47.2
<i>Task-specific fine-tuned</i>											
3D-VisTA (Zhu et al., 2023c)	66.9	34.0	27.1	54.3	53.8	69.6	10.4	13.9	35.7	22.4	48.5
3D-LLM (FlanT5) (Hong et al., 2023)	-	-	-	-	-	69.4	12.0	14.5	35.7	20.5	-
LEO	68.4	36.9	27.7	57.8	54.7	80.0	11.5	16.2	39.3	36.6	53.7

Table 3: Results on CLIPort robot manipulation. We compare with results from Shridhar et al. (2021). *seen* indicates in-domain tasks. *unseen* marks OOD tasks with novel colors or objects.

	separating-piles		packing-google-objects-seq		put-blocks-in-bowls	
	seen	unseen	seen	unseen	seen	unseen
Transporter	48.4	52.3	46.3	37.3	64.7	18.7
CLIP-only	90.2	71.0	95.8	57.8	97.7	44.5
RN50-BERT	46.5	44.9	94.0	56.1	91.8	23.8
CLIPort (single)	98.0	75.2	96.2	71.9	100	25.0
CLIPort (multi)	89.0	62.8	84.4	70.3	100	45.8
LEO	98.8	75.2	76.6	79.8	86.2	35.2

Table 4: Results on object navigation. We compare LEO with similar imitation learning agents in Habitat-web (H.w.) (Ramrakhy et al., 2022) and CortexBench (VC-1) (Majumdar et al., 2023). S: success rate; L: SPL. [†] LEO is not trained on HM3D scenes.

	MP3D-val		HM3D-val	
	S(↑)	L(↑)	S(↑)	L(↑)
H.w. (shortest)	4.4	2.2	-	-
H.w. (70k demo)	35.4	10.2	-	-
VC-1 (ViT-B)	-	-	57.1	31.4
LEO	23.1	15.2	23.1 [†]	19.1 [†]





北京航空航天大學
BEIHANG UNIVERSITY

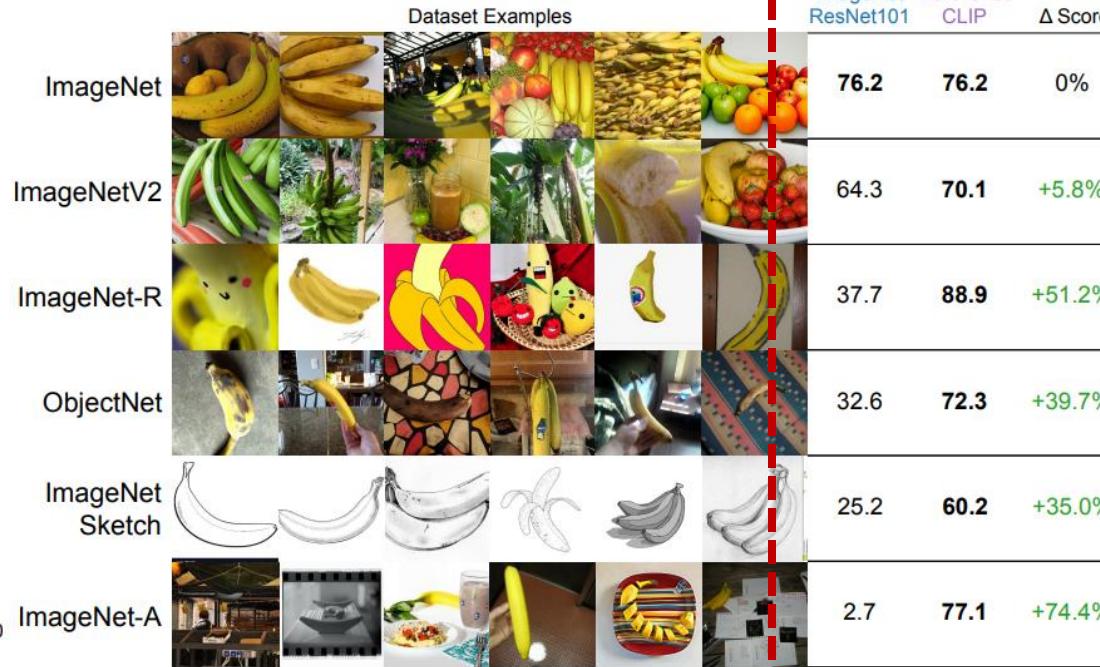
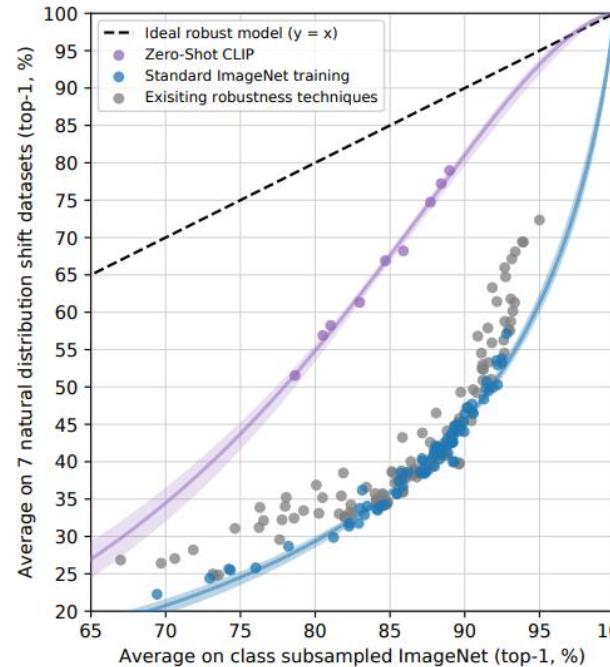
人工智能研究院
Institute of Artificial Intelligence

Intro to VLLMs Safety



多模态大模型的显著优势：应对不同数据分布的强大泛化性

基于图文对比学习预训练的CLIP 在zero-shot分类中能够准确预测各种不同的OOD样本



相比于传统task-specific模型
(ResNet101等)
CLIP在各类OOD基准数据集
上的表现具有突破性的提升



多模态大模型的显著优势: 应对不同数据分布的强大泛化性

基于指令微调的视觉语言大模型LLaVa 能够准确理解图像中的unusual和复杂对象

Visual input example, Extreme Ironing:



Source: <https://www.barnorama.com/wp-content/uploads/2016/12/03-Confusing-Pictures.jpg>

What is unusual about this image?

The unusual aspect of this image is a man ironing clothes on the back of a minivan or van. This is not a typical place to perform this activity, as one would usually iron clothes in a more stationary and safe location, such as a home, using a regular ironing board. The scene depicted in the image is peculiar as it involves a makeshift ironing setup on a vehicle, which can be both unsafe and unconventional. Additionally, it is not clear how the man is able to maintain balance and stability while ironing clothes in such an unstable environment.

[Start a new conversation, and clear the history]

What's happening in the scene?

The image depicts a man standing on top of a yellow SUV in a busy city street. He is holding a portable ladder, seemingly ironing clothes while standing on the vehicle. Around the scene, there are other cars, a traffic light, and a fire hydrant. Additionally, there are two other people in the vicinity, one close to the man on the car and another further away. The man on the ladder appears to be attracting attention due to his unconventional choice of ironing his clothes on top of a moving car. The city street around him is bustling with activity, adding to the unique nature of the scene.

User
LLaVA

User
LLaVA

User
GPT-4 [36]

User
BLIP-2

User
OpenFlamingo

Challenging examples from LLaVA-Bench (In-the-Wild):



ICHIRAN Ramen [[source](#)]



Filled fridge [[source](#)]

Annotation

A close-up photo of a meal at ICHIRAN. The chashu ramen bowl with a spoon is placed in the center. The ramen is seasoned with chili sauce, chopped scallions, and served with two pieces of chashu. Chopsticks are placed to the right of the bowl, still in their paper wrap, not yet opened. The ramen is also served with nori on the left. On top, from left to right, the following sides are served: a bowl of orange spice (possibly garlic sauce), a plate of smoke-flavored stewed pork with chopped scallions, and a cup of matcha green tea.

An open refrigerator filled with a variety of food items. In the left part of the compartment, towards the front, there is a plastic box of strawberries with a small bag of baby carrots on top. Towards the back, there is a stack of sauce containers. In the middle part of the compartment, towards the front, there is a green plastic box, and there is an unidentified plastic bag placed on it. Towards the back, there is a carton of milk. In the right part of the compartment, towards the front, there is a box of blueberries with three yogurts stacked on top. The large bottle of yogurt is Fage non-fat yogurt, and one of the smaller cups is Fage blueberry yogurt. The brand and flavor of the other smaller cup are unknown. Towards the back, there is a container with an unknown content.

Question 1 What's the name of the restaurant?

What is the brand of the blueberry-flavored yogurt?

Question 2 Describe this photo in detail.

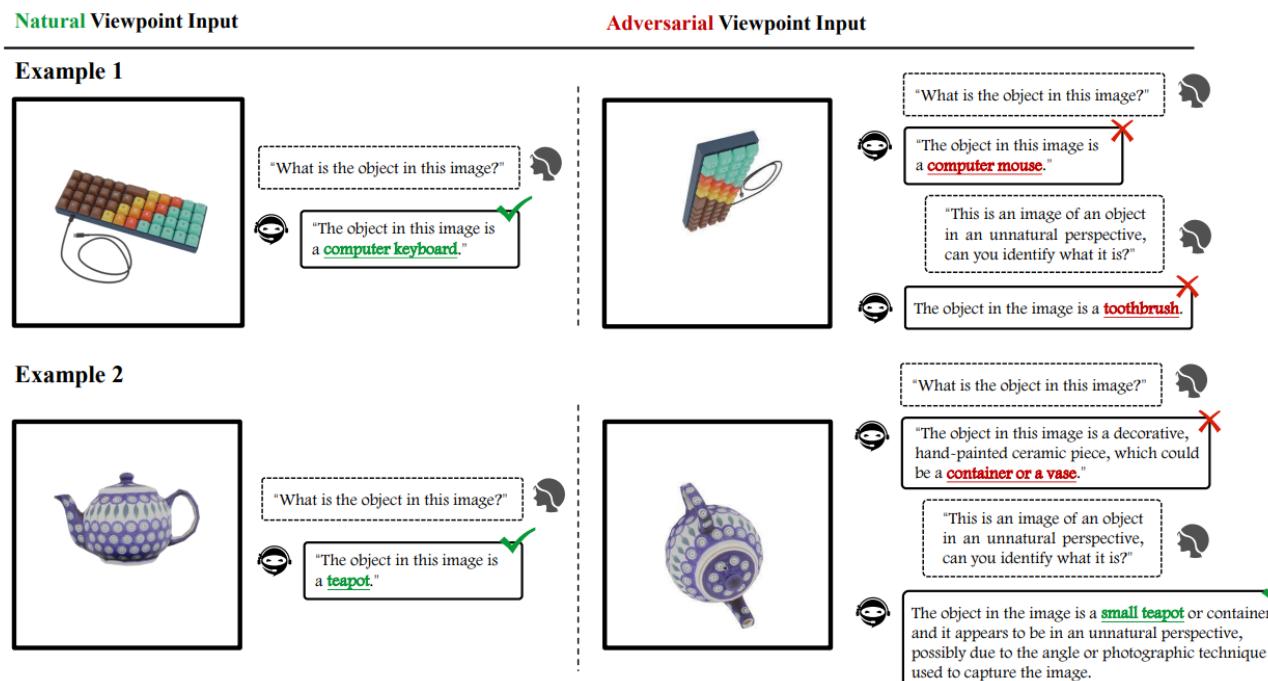
Is there strawberry-flavored yogurt in the fridge?



然而目前的评估大多关注2D-OOD变换 忽略了3D视角变换

我们观察到：在应对视角OOD样本时 视觉语言大模型的性能发生显著下降

- Vision question answering



- Zero-shot classification

Models	Vision Encoder	# Toatal Params	ImageNet	IM3D	ImageNet-V+
CLIP [49]	ALFBF [36]	ViT-B/16	210M	66.06% (↓ 13.18%)	52.88% (↓ 39.84%)
	ResNet-50	102M	65.12%	53.36% (↓ 11.76%)	2.51% (↓ 62.61%)
	ViT-B/16	150M	76.94%	66.60% (↓ 10.34%)	37.99% (↓ 38.95%)
	ViT-B/32	151M	72.74%	58.59% (↓ 14.15%)	29.24% (↓ 43.50%)
	ViT-L/14	428M	81.96%	76.16% (↓ 5.80%)	48.49% (↓ 33.47%)
BLIP [35]	ViT-B/16	224M	70.02%	70.73% (↑ 0.71%)	40.08% (↓ 29.94%)
	ViT-L/14	449M	73.86%	76.38% (↑ 2.52%)	50.05% (↓ 23.81%)
BLIP-2 [34]	ViT-G/14	1.2B	77.40%	83.76% (↑ 6.36%)	57.92% (↓ 19.48%)

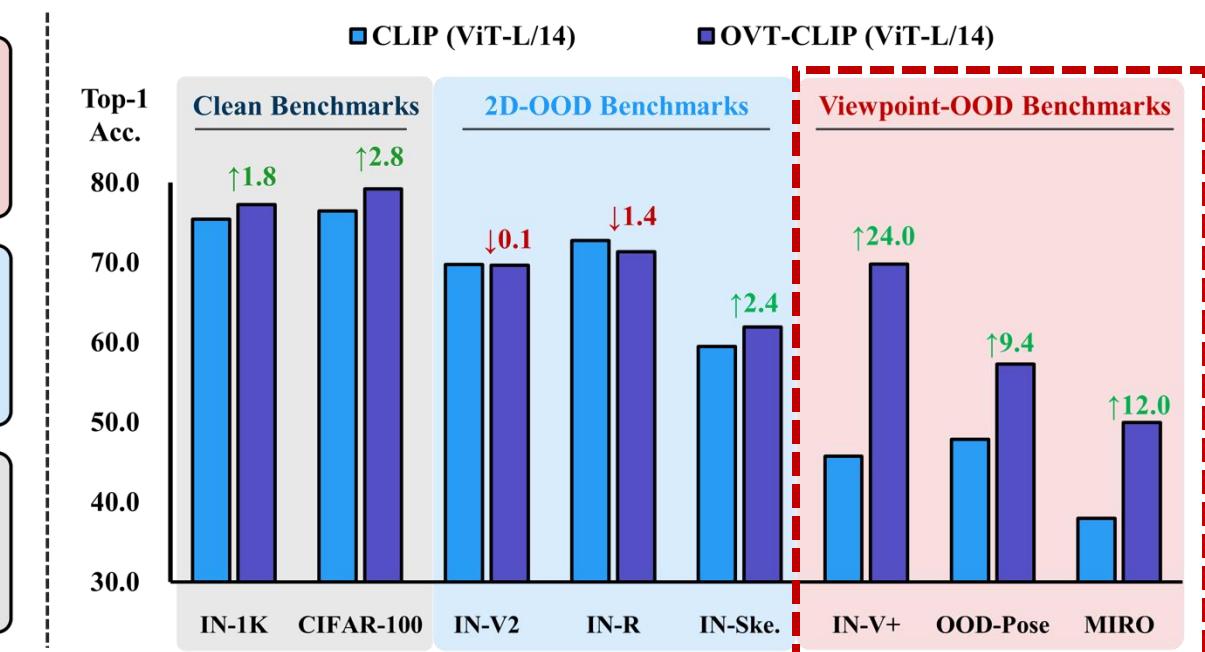
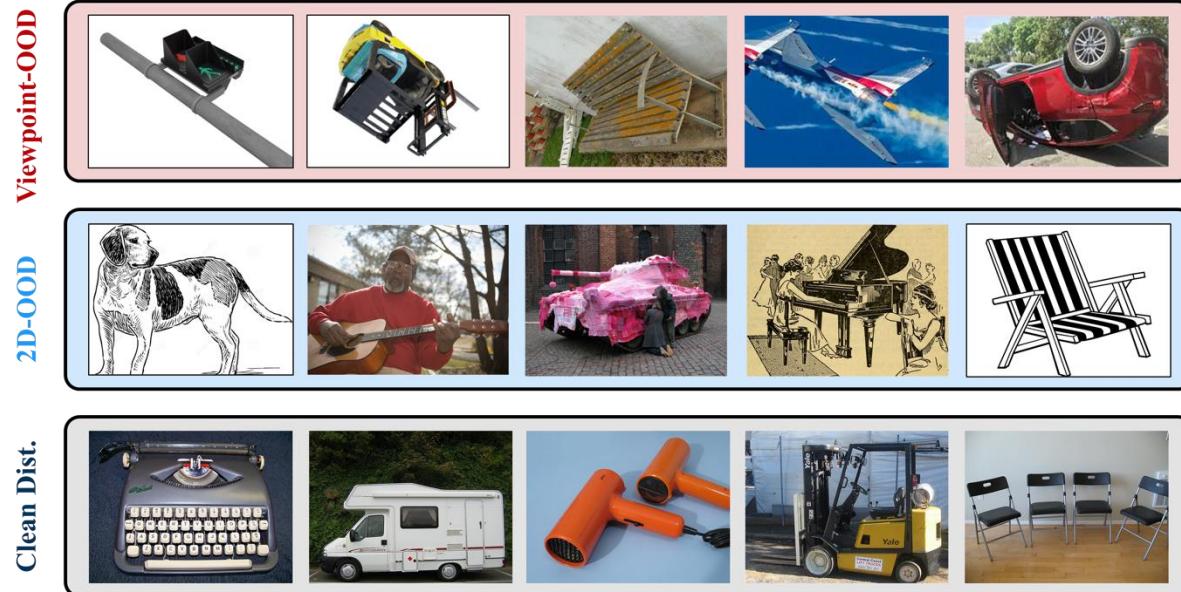
• Showei Ruan et.al., Improving viewpoint robustness for visual recognition via adversarial training.

注：IM3D为多视角数据集，ImageNet-V+是包含对抗性视角的基准数据集



和常见的2D-OOD变换相比 3D视角变换对于视觉语言模型更具挑战性

针对CLIP 我们在常用的2D-OOD基准和视角OOD基准进行性能的比较：





应对3D视角变换的鲁棒性 对于AI系统在安全敏感领域的应用 具有重要意义

如自动驾驶领域 系统需要能够准确的识别来自异常视角的视觉输入 否则将造成安全事故

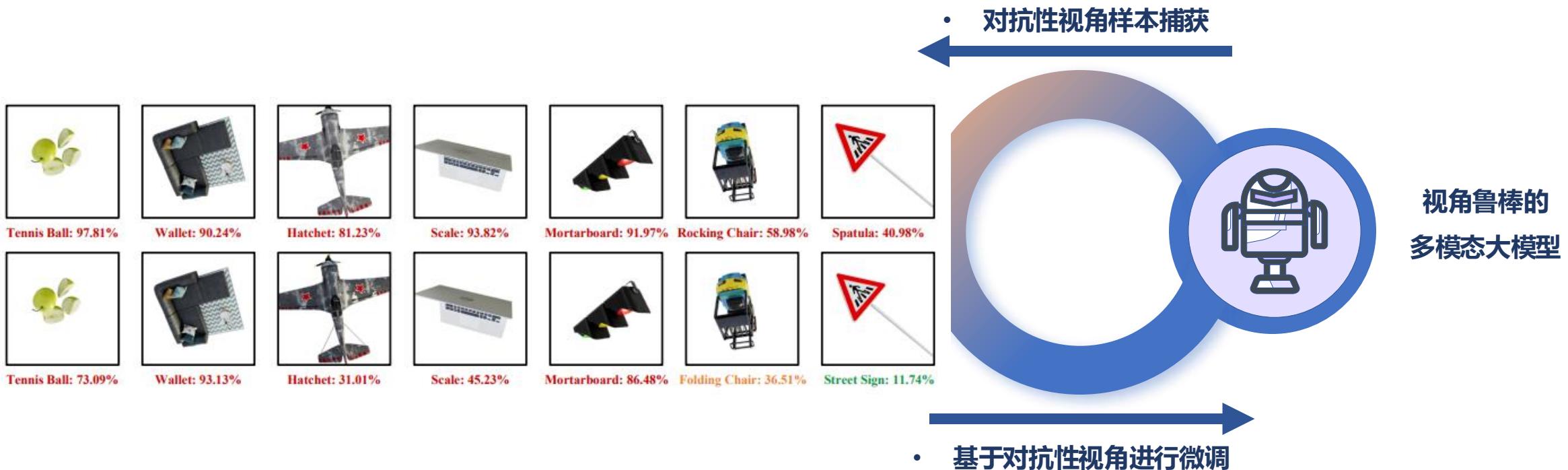


鉴于当前多模态大模型在自动驾驶等安全敏感领域开始逐渐应用 增强其视角鲁棒性具有重要意义



如何提高多模态大模型对于视角变换的鲁棒性？

1. 从数据增强和对抗性训练的范式考虑：通过某种手段 挖掘大模型的脆弱视角
2. 基于这些脆弱视角进行微调 从而提高模型对不同视角的一致性表征能力



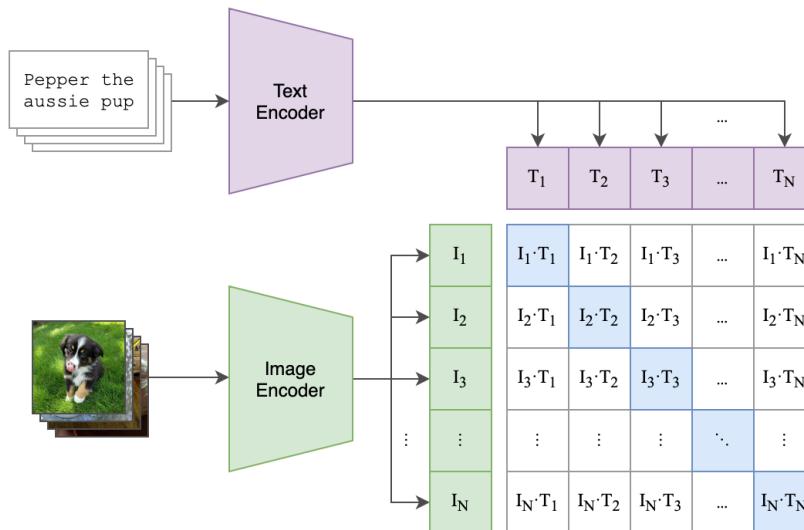


当前主流多模态基础模型的训练范式

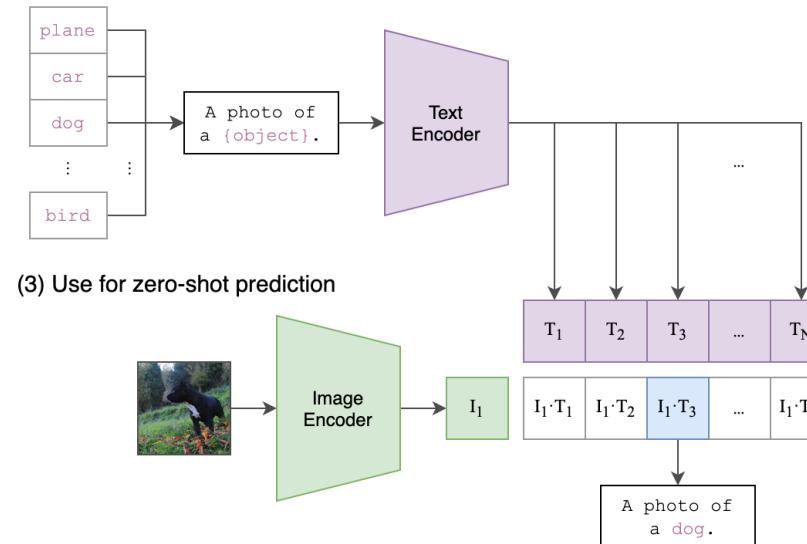
尽管多模态学习的范式众多，但以 CLIP & ALIGN & BLIP 等模型为代表的图文对比学习范式占据主流

特点： ➤ 对比学习形式的目标 ➤ 图像/文本 双流编码器架构 ➤ 海量的图文对 作为训练数据集

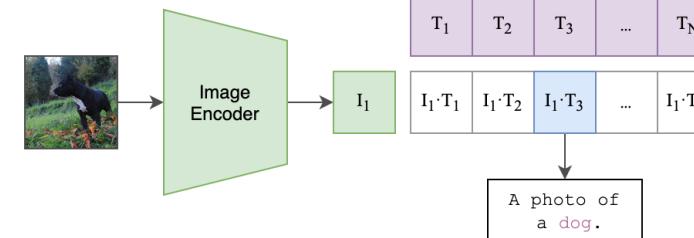
(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



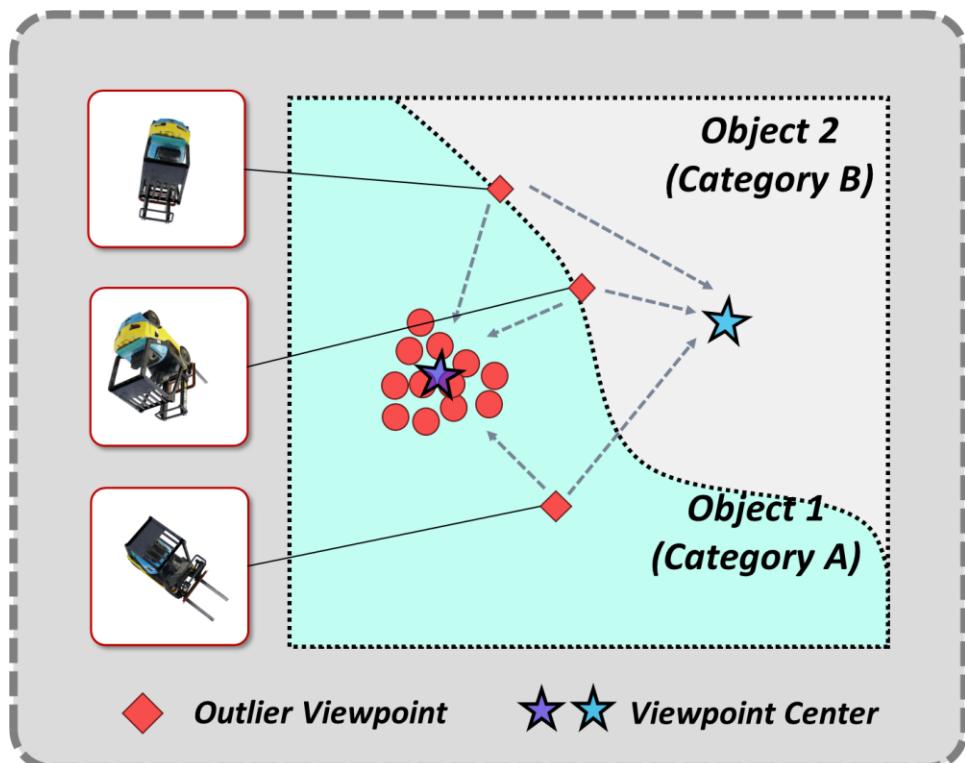
- 因此，有必要考虑对这类模型展开视角鲁棒性的增强



增强多模态模型对脆弱视角的一致性表征

我们的思路是 在所挖掘的脆弱视角的基础上

强化多模态模型对于这些视角的表征能力 使其能够和正常的视角在表征空间中足够接近



类似于图-文对齐目标

我们针对多模态模型提出了跨视角对齐目标：

在高维嵌入空间中 **最小化** 脆弱视角和中心视角的表征距离

$$\min_{\mathbf{W}_v, \mathbf{W}_t} [\mathcal{L}_{ITC} + \lambda \cdot \underbrace{\sum_i \sum_{j \neq j'} d(z_{ij}^I, z_{ij'}^I)}_{\mathcal{L}_{VC}},$$

$$\min_{\mathbf{W}_v, \mathbf{W}_t} \left[\mathcal{L}_{ITC} + \lambda \cdot \max_{\mathcal{O}=\{O_i\}_{i=1}^N, |O_i|=K} \sum_{i=1}^N \sum_{j \in \mathcal{O}} l(z_{ij}^I, z_{C_i}^I) \right],$$

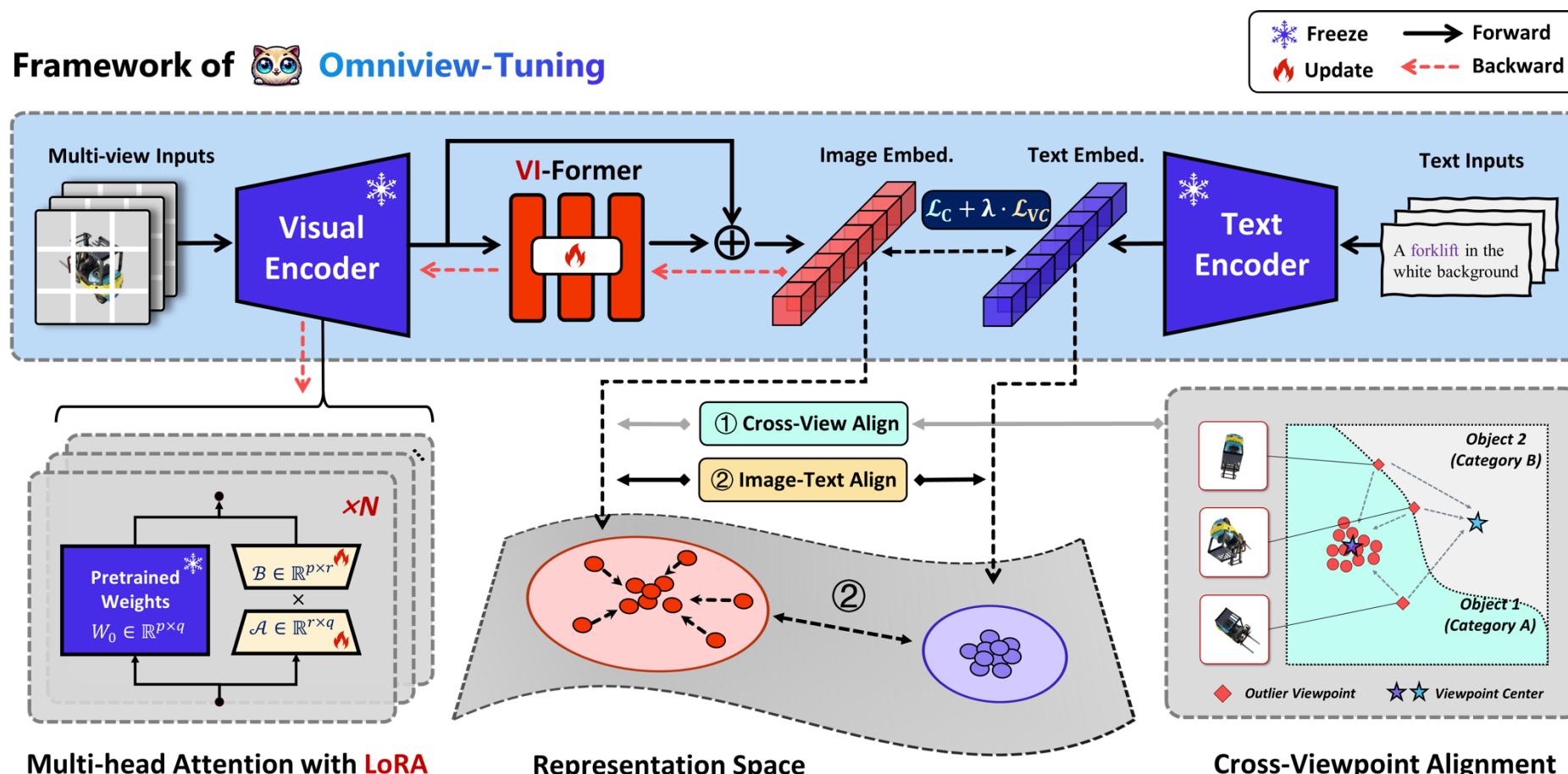
$$\text{where } l(z_{ij}^I, z_{C_i}^I) = \max [d(z_{ij}^I, z_{C_i}^I) + m, 0],$$



增强多模态模型对脆弱视角的一致性表征

我们提出了Omniview-Tuning (OVT) 框架：通过高效参数微调增强多模态模型的视角鲁棒性

Framework of Omnidview-Tuning



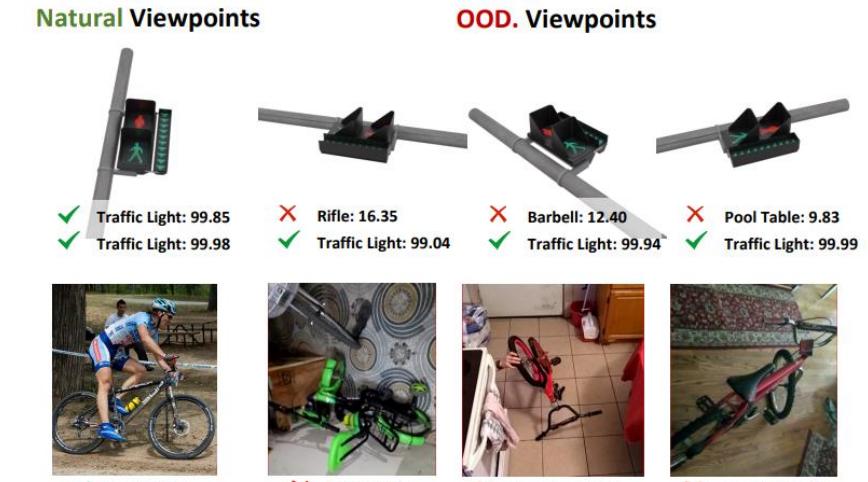


增强多模态模型对脆弱视角的一致性表征

我们的方法能够应用于各种主流的图像-文本多模态架构（如CLIP, BLIP）

在横跨不同数据分布的基准（包含2D-OOD, 视角-OOD, 以及干净数据）上展开zero-shot分类能力的评估

Model	Clean				Common-OOD				Viewpoint-OOD				Total Avg. Acc.				
	ImageNet-100 [47]	ImageNet-1K [13]	Cifar-100 [28]	Avg. Acc.	ImageNet-V2 [45]	ImageNet-Ske. [56]	ImageNet-OOD. [22]	ImageNet-Ren. [20]	OOD-CV [60]	Avg. Acc.	ImageNet-View. [15]	ImageNet-View.+ [47]	OOD-CV-Pose [60]	MIRO [7]	Avg. Top-1		
A. Comparisons with ViT-B/32 baselines																	
OpenAI CLIP	77.5/93.9	63.3/88.8	64.3/88.1	68.4/90.2	55.8/83.4	42.2/70.3	33.4/62.2	50.7/75.4	50.2/82.6	46.5/74.8	44.5/65.4	27.5/52.4	47.2/84.5	26.5/59.4	36.4/65.4	48.6/75.5	
Open CLIP	81.1 /95.3	66.5/89.9	75.8 / 94.0	74.5 / 93.0	58.1 /83.9	53.6 / 79.3	34.8/64.4	61.0 / 81.9	53.5 / 81.9	52.2 / 78.3	54.4/72.1	37.1/63.2	46.9/81.6	33.0/69.2	42.8/71.5	54.6/79.7	
OV-T-OpenCLIP	80.9/95.6	67.8/90.8	65.0/89.3	71.2/91.9	58.0/84.2	45.8/73.4	42.8/75.0	50.3/71.4	51.7/79.5	49.7/76.7	61.9 / 81.2	59.5 / 85.6	52.8 / 82.5	35.4 / 80.1	52.4 / 82.4	56.0 / 82.4	
MetaCLIP	80.7/95.6	67.6/90.5	77.7 / 95.2	75.3 / 93.8	59.5/85.4	55.9 / 81.4	32.4/62.5	63.2 / 83.8	52.0 / 84.2	52.6 / 79.5	61.4/67.6	41.0/67.8	48.9/ 87.9	34.8/73.2	46.5/76.4	56.3/82.0	
OV-T-MetaCLIP	80.7 / 95.6	69.7 / 92.0	71.8/93.0	74.0/93.5	60.6 / 85.8	47.8/75.8	43.5 / 73.8	49.0/70.8	50.1/80.1	50.2/77.2	64.0 / 79.2	54.8 / 80.4	55.1 / 84.8	35.6 / 77.0	52.4 / 80.3	56.9 / 82.3	
B. Comparisons with ViT-B/16 baselines																	
OpenAI CLIP	82.1/95.7	68.3/91.9	67.2/89.4	72.5/92.3	61.8/87.4	48.2/76.3	27.7/55.7	59.1/83.0	52.2/ 84.6	49.8/77.4	51.6/68.9	36.9/63.8	53.4/86.8	30.1/66.1	43.0/71.4	53.2/79.1	
Open CLIP	83.2/96.2	70.1/91.8	77.0 / 94.8	76.8 / 94.3	62.2/87.0	56.0 / 82.0	30.7/59.8	64.9 / 85.6	54.3/82.7	53.6 / 79.4	58.1/74.4	44.2/70.9	48.5/84.0	34.6/74.6	46.4/76.0	57.0/82.0	
OV-T-OpenCLIP	83.9 / 97.0	71.9 / 93.1	69.0/90.7	74.9/93.6	64.0 / 88.6	50.5/77.9	36.8 / 68.9	57.0/77.2	56.3 / 84.5	52.9/ 79.4	65.4 / 80.7	61.7 / 85.8	56.9 / 87.4	42.4 / 84.9	56.6 / 84.7	59.6 / 84.7	
EVA-CLIP	85.3 /96.5	74.6 / 94.2	87.5/98.0	82.5 / 96.3	67.0 / 89.8	57.6/82.3	21.3/47.3	69.6/87.5	53.1 / 83.1	53.7/78.0	61.8/76.6	44.3/69.4	53.9/87.4	32.9/73.2	48.2/76.6	59.1/82.1	
MetaCLIP	84.3/97.2	72.1/93.4	78.9 / 95.4	78.4/95.3	65.0/89.3	60.1 / 84.8	26.2/56.4	70.2 / 89.3	52.3/85.4	54.8 / 81.0	64.2/79.4	49.6/76.1	48.9/ 90.9	38.5/78.7	50.3/81.2	59.2/84.7	
OV-T-MetaCLIP	83.4/ 97.4	73.8/94.1	73.9/93.6	77.0/95.0	65.9/89.4	53.6/81.0	36.2 / 66.8	59.0/79.6	51.6/ 83.8	53.2/80.1	69.7 / 84.0	64.8 / 87.3	55.2 / 87.8	39.2 / 82.9	57.2 / 85.5	60.5 / 85.6	
C. Comparisons with ViT-L/14 baselines																	
OpenAI CLIP	86.5/97.4	75.4/94.6	76.5/93.3	79.5/95.1	69.8 /90.9	59.5/84.3	18.6/43.8	72.8/91.4	52.9/88.8	54.7/79.8	60.3/75.6	45.8/71.5	47.9/88.2	38.0/74.1	48.0/77.3	58.6/82.8	
Open CLIP	86.8/97.8	75.2/94.3	83.7 / 96.7	81.9 / 96.2	67.7/90.2	63.2 / 86.4	24.0/50.5	74.5 / 91.2	54.5/85.0	56.8/80.6	65.7/78.1	53.2/76.7	52.4/90.5	42.3/83.0	53.4/82.1	61.9/85.0	
OV-T-OpenCLIP	89.0 / 97.8	77.3 / 95.3	79.2/95.3	81.8/96.1	69.6/ 91.5	61.9/86.0	27.5 / 55.4	71.3/88.7	56.4 / 87.0	57.3 / 81.7	72.2 / 86.6	69.8 / 89.7	57.3 / 94.1	50.0 / 89.3	62.3 / 89.9	65.1 / 88.1	
EVA-CLIP	88.5/97.9	79.6 / 96.0	90.6	98.6 / 86.3	97.5	72.8 / 92.7	68.0/89.1	16.3/40.0	82.8 / 95.7	54.7/87.4	58.9/81.0	71.5/82.3	61.1/81.7	54.4/ 94.5	39.6/86.1	56.6/86.1	65.0/86.8
MetaCLIP	88.3/ 98.3	79.1/95.9	84.1/96.9	83.8/97.0	72.5/92.6	68.9 / 89.8	17.0/40.6	81.8/95.1	56.6 / 87.5	59.3 / 81.1	77.3/89.3	66.4/87.0	58.9 / 93.3	48.1 / 89.6	62.7/89.8	66.6 / 88.0	
OV-T-MetaCLIP	88.8 / 97.5	77.7/95.9	84.0/96.9	83.5/96.8	70.8/92.2	64.4/87.9	20.8 / 47.0	77.0/92.7	56.3/ 89.3	57.8/ 81.8	79.3 / 90.6	75.4 / 93.0	57.0/94.4	46.4/ 93.8	64.5 / 92.9	66.5/ 89.3	
D. Comparisons with BLIP ViT-B/16 baselines																	
BLIP	76.6/93.3	52.9/80.2	67.0 / 88.3	65.5/87.3	47.3/74.7	51.0 / 76.6	25.6/53.4	64.3 / 83.8	53.9/ 87.6	48.4/75.2	55.2/68.2	36.8/63.3	50.8/ 89.9	27.0/66.1	42.4/71.9	50.7/77.1	
OV-T-BLIP	82.2 / 97.0	61.7 / 88.8	66.6/ 88.9	70.2 / 91.5	53.7 / 82.9	46.5/74.2	33.8 / 62.7	57.4/77.9	56.4 / 87.3	49.6 / 77.0	62.6 / 79.0	54.8 / 79.9	55.2 / 89.5	31.5 / 73.2	51.0 / 80.4	55.2 / 81.8	



结果表明 经过OV-T训练
模型在视角OOD样本上的表现显著提升
并维持了在其他数据分布上的泛化性



增强多模态模型对脆弱视角的一致性表征

Model	Visual Encoder	Real-world Domain					Synthetic Domain				
		OOD-CV (iid) [60]			OOD-CV (Pose) [60]		IM3D [47]			ImageNet-V+ [47]	
		$\beta@1.0$	$\beta@0.5$	$\beta@Adp.$	$\beta@1.0$	$\beta@0.5$	$\beta@Adp.$	$\beta@1.0$	$\beta@0.5$	$\beta@Adp.$	$\beta@1.0$
LLaVa-7b	OpenAI CLIP(ViT-L/14)	44.1	61.1	67.5	46.4	53.6	58.7	46.7	53.3	58.8	20.4
	<i>TeCoA⁴</i> [38](ViT-L/14)	41.9	58.9	65.5	36.1	41.6	49.2	26.3	30.1	42.6	8.7
	<i>FARE⁴</i> [49](ViT-L/14)	42.1	58.9	65.2	40.2	45.9	50.8	35.2	39.2	49.2	12.7
	OVT-CLIP(ViT-L/14)	43.5	59.5	65.9	46.5	53.6	59.1	49.4	54.0	61.8	26.4
LLaVa-13b	OpenAI CLIP(ViT-L/14)	45.4	68.0	70.6	48.6	58.6	60.8	48.7	56.7	60.8	21.2
	<i>TeCoA⁴</i> [38](ViT-L/14)	42.4	67.0	72.2	37.4	48.9	51.3	25.0	28.6	41.5	8.4
	<i>FARE⁴</i> [49](ViT-L/14)	43.9	66.7	71.1	41.9	52.1	54.8	36.1	41.4	48.6	12.1
	OVT-CLIP(ViT-L/14)	45.7	67.3	70.8	48.2	58.6	61.9	50.4	58.9	63.2	26.4



同时，在应用于视觉语言大模型
(LLaVa, Openflamingo) 时
我们的方法也能够有效提升其在
Image Caption任务上的视角鲁棒性



北京航空航天大學
BEIHANG UNIVERSITY

人工智能研究院
Institute of Artificial Intelligence

多模态大语言模型：基础原理与前沿应用

阮受炜

2024年5月

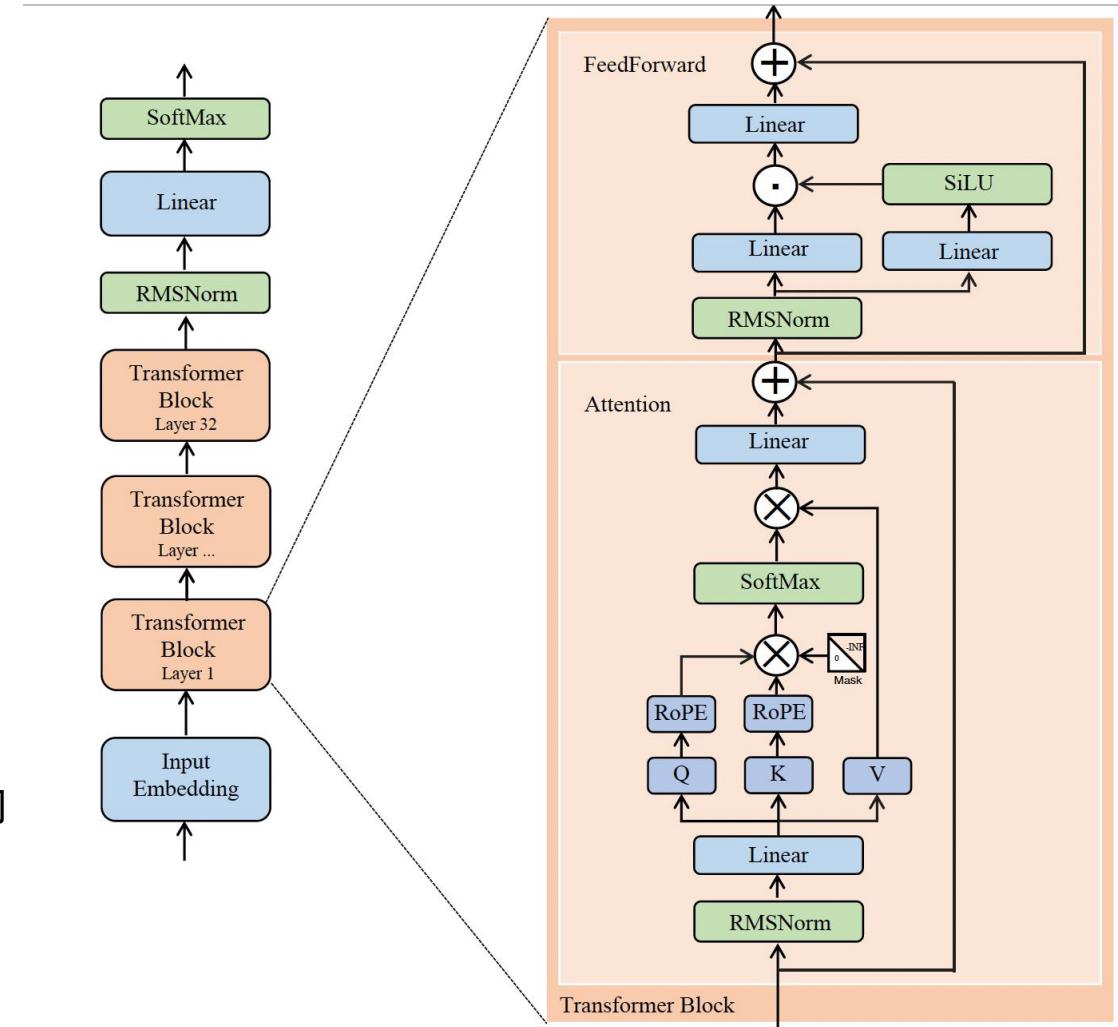
Part

04

LLM网络结构

LLAMA 网络结构

- 以开源的LLAMA为例
- Decoder Only
- Why Decoder Only?
 - 工程优势
 - 支持复用KV-Cache
 - 易于Scale up
 - GPT的分布式优化（流水线并行）比有复杂连接的EncoderDecoder的T5模型容易的多
 - 理论优势：
 - Encoder双向注意力的低秩问题：不带Causal Mask的注意力矩阵可能是低秩的；Decoder的单向注意力由于是三角阵必然满秩
- 跟一般的Transformer Decoder存在区别
 - RMSNorm、RoPE位置编码、KV Cache、Group Query Attention、PostNorm、FFN、SiLU



- 对比LayerNorm和RMSNorm

$$\text{LayerNorm} : y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta$$

$$E[x] = \frac{1}{N} \sum_{i=1}^N x_i$$

$$Var[x] = \frac{1}{N} \sum_{i=1}^N (x_i - E[x])^2$$

$$\text{RMSNorm} : y = \frac{x}{\sqrt{Mean(x^2) + \epsilon}} * \gamma$$

$$Mean(x^2) = \frac{1}{N} \sum_{i=1}^N x_i^2$$

- 计算量更小，计算速度更快
- 效果上没有明显差别

PreNorm & PostNorm

- Transformer 类模型中，LayerNorm 层有两种放置方式

➤ 到底是先加再Norm，还是先Norm再加残差

➤ PreNorm容易训练，对于深层网络学习的效果不太好

- 更偏重恒等分支，相邻层激活值差异小
- 残差连接效应强，使得梯度的回传更容易
- 近似等效于更宽的网络而不是深层网络

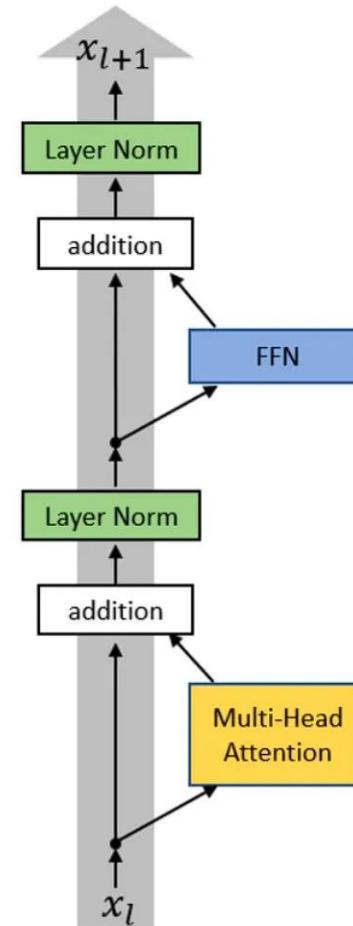
$$x_l = x_0 + F_0(x_0) + F_1(x_1/\sigma_1) + \dots + F_{l-1}(x_{l-1}/\sigma_{l-1}),$$

$$x_{l+1} = x_0 + F_0(x_0) + F_1(x_1/\sigma_1) + \dots + F_{l-1}(x_{l-1}/\sigma_{l-1}) + F_l(x_l/\sigma_l)$$

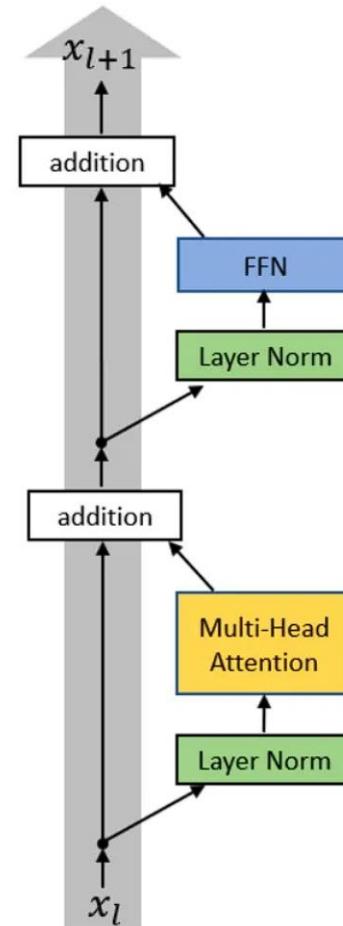
➤ PostNorm不太容易训练

- 使得梯度快速衰减，导致模型难以学习
- 与RMSNorm结合会导致分布问题，带来梯度消失

$$\begin{aligned} x_l &= \frac{x_{l-1}}{\sqrt{2}} + \frac{F_{l-1}(x_{l-1})}{\sqrt{2}} \\ &= \frac{x_{l-2}}{2} + \frac{F_{l-2}(x_{l-2})}{2} + \frac{F_{l-1}(x_{l-1})}{\sqrt{2}} \\ &= \dots \\ &= \frac{x_0}{2^{l/2}} + \frac{F_0(x_0)}{2^{l/2}} + \frac{F_1(x_1)}{2^{(l-1)/2}} + \frac{F_2(x_2)}{2^{(l-2)/2}} + \dots + \frac{F_{l-1}(x_{l-1})}{2^{1/2}} \end{aligned}$$



(a)



(b)

- Pre Norm: $x_{t+1} = x_t + F_t(\text{Norm}(x_t))$
Post Norm: $x_{t+1} = \text{Norm}(x_t + F_t(x_t))$