

3. 长上下文激活阶段：使用上采样的长上下文冷却数据训练，处理扩展序列任务。通过上采样长上下文数据，在训练期间使用不同比例的全注意力数据和部分注意力数据，并逐步增加最大序列长度。

大规模 RL 训练系统：设计大规模 RL 训练系统，采用部分回放技术优化长 CoT RL 训练。该系统通过迭代同步方法操作，每次迭代包括展开阶段和训练阶段。在展开阶段，展开工作进程生成响应序列并存储在重放缓冲区；在训练阶段，训练工作进程访问这些经验更新模型权重。系统还包含代码执行服务（沙箱），用于处理代码相关问题，评估模型在实际编码场景中的输出。

混合部署框架：利用 Kubernetes Sidecar 容器共享 GPU 资源，实现训练和推理任务的并行执行。在 Megatron 和 vLLM 之上实现混合部署框架，训练到推理阶段的转换耗时不到一分钟，反之约十秒钟。该框架可解决复杂的并行策略、最小化空闲 GPU 资源和实现动态扩展等问题。

4. 技术优势

出色的推理性能：在长思维链模式下，Kimi k1.5 在多个基准测试和模态中达到了 SOTA 模型 OpenAI o1 正式版的水平，如在 AIME 上达到 77.5 分，MATH 500 上达到 96.2 分，在 Codeforces 上达到 94 百分位，在 MathVista 上达到 74.9 分。在短思维链模式下，其数学、代码、视觉多模态和通用能力也表现出色，大幅超越现有短链思维模型，如在 AIME 上达到 60.8 分，MATH500 上达到 94.6 分，LiveCodeBench 上达到 47.3 分，提升幅度高达 550%。

创新的技术架构通过长上下文扩展和改进的策略优化，Kimi k1.5 建立了一个简单有效的 RL 框架，无需依赖复杂技术如蒙特卡罗树搜索、价值函数和过程奖励模型，即可实现强大的性能。同时，Long2short 方法有效提升了短 CoT 模型的性能，提高了模型的 token 效率。

高效的数据处理与训练精心设计的数据工程确保了训练数据的高质量 and 多样性，多模态数据的有效整合使模型能够更好地理解和处理不同类型的信息。优化的训练架构和算法，如部分回放技术、混合部署框架等，提高了训练效率和系统的可扩展性。

6.4 基于人类反馈的强化学习

6.4.1 基于人类反馈的强化学习流程

在进行有监督微调后，大语言模型具备了遵循指令和多轮对话，以及初步与用户进行对话的能力。然而，由于庞大的参数数量和训练数据量，大语言模型的复杂性往往难以理解和预测。当这些模型被部署时，可能会产生严重的后果，尤其是当模型变得日渐强大、应用更加广泛，并且频繁地与用户进行互动时。因此，研究人员追求将人工智能与人类价值观进行对齐，文献 [24] 提出大语言模型输出的结果应该满足有用性（Helpfulness）、真实性（Honesty）及无害性（Harmless）的 3H 原则。由于上述 3H 原则体现出了人类偏好，因此基于人类反馈的强化学习（RLHF）很自然地引入了通用对话模型的训练流程。

基于人类反馈的强化学习主要分为奖励模型训练和近端策略优化两个步骤。奖励模型通过由

人类反馈标注的偏好数据来学习人类的偏好，判断模型回复的有用性，保证内容的无害性。奖励模型模拟了人类的偏好信息，能够不断地为模型的训练提供奖励信号。在获得奖励模型后，需要借助强化学习对语言模型继续进行微调。OpenAI 在大多数任务中使用的强化学习算法都是 PPO 算法。近端策略优化可以根据奖励模型获得的反馈优化模型，通过不断的迭代，让模型探索和发现更符合人类偏好的回复策略。近端策略优化算法的实施流程如图6.4 所示。

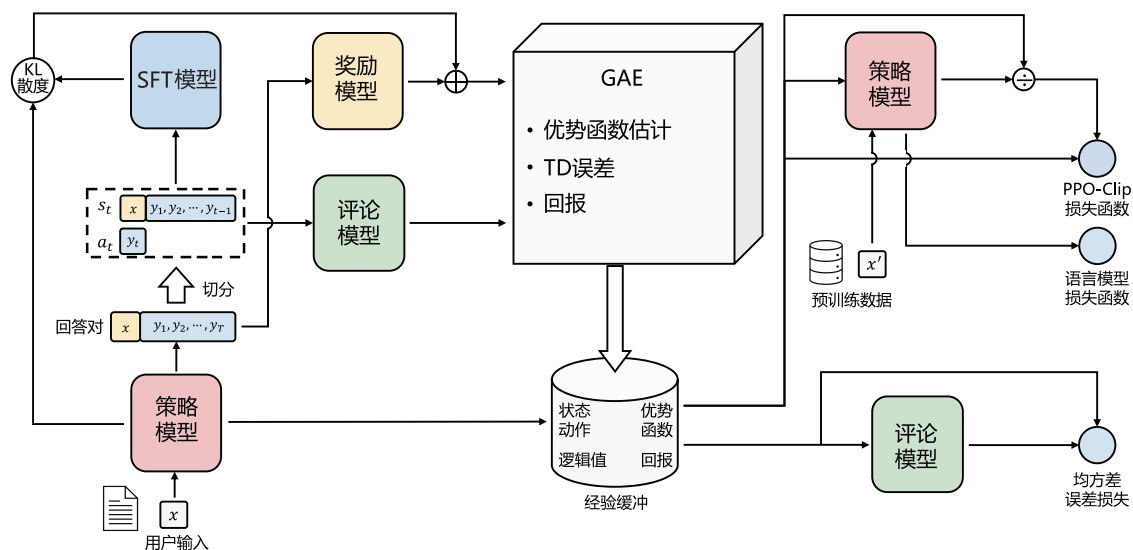


图 6.4 近端策略优化算法的实施流程^[246]

近端策略优化涉及以下四个模型。

- (1) 策略模型 (Policy Model)，生成模型回复。
- (2) 奖励模型 (Reward Model)，输出奖励分数来评估回复质量的好坏。
- (3) 评论模型 (Critic Model)，预测回复的好坏，可以在训练过程中实时调整模型，选择对未来累积收益最大的行为。
- (4) 参考模型 (Reference Model)，提供了一个 SFT 模型的备份，使模型不会出现过于极端的变化。

近端策略优化算法的实施流程如下。

- (1) 环境采样：策略模型基于给定输入生成一系列的回复，奖励模型则对这些回复进行打分获得奖励。
- (2) 优势估计：利用评论模型预测生成回复的未来累积奖励，并借助广义优势估计 (Generalized Advantage Estimation, GAE) 算法估计优势函数，有助于更准确地评估每次行动的好处。
- (3) 优化调整：使用优势函数优化和调整策略模型，同时利用参考模型确保更新的策略不会

有太大的变化，从而维持模型的稳定性。

6.4.2 奖励模型

基于人类反馈训练的奖励模型可以很好地学习人类的偏好。理论上，可以通过强化学习使用人类标注的反馈数据直接对模型进行微调建模。然而，由于工作量和时间的限制，针对每次优化迭代，人类很难提供足够的反馈。更为有效的方法是构建奖励模型，模拟人类的评估过程。奖励模型在强化学习中起着至关重要的作用，它决定了智能体如何从与环境的交互中学习并优化策略，以实现预定的任务目标。本节将从数据收集、模型训练和开源数据三个方面介绍大语言模型奖励模型的实现。

1. 数据收集

针对文献 [24] 所提出的大语言模型应该满足的 3H 原则，如何构建用于训练奖励模型的数据是奖励模型训练的基础。本节介绍的奖励模型数据收集细节主要依据 Anthropic 团队在文献 [247] 中介绍的 HH-RLHF 数据集构建过程。主要针对有用性和无害性，分别收集了不同人类偏好数据集。

(1) **有用性**：有用性意味着模型应当遵循指令；它不仅要遵循指令，还要从少量的示例提示或其他可解释的模式中推断出意图。然而，给定提示背后的意图经常不够清晰或存在歧义，这就是需要依赖标注者的判断的原因，他们的偏好评分构成了主要的衡量标准。在数据收集过程中，让标注者使用模型，期望模型帮助用户完成纯粹基于文本的任务（如回答问题、撰写编辑文档、讨论计划和决策）。

(2) **无害性**：无害性的衡量也具有挑战性。语言模型造成的实际损害程度通常取决于它们的输出在现实世界中的使用方式。例如，一个生成有毒输出的模型在部署为聊天机器人时可能会有害，但如果被用于数据增强，以训练更精确的毒性检测模型，则可能是有益的。在数据收集过程中，标注者通过一些敌对性的询问，比如计划抢银行等，可能会引诱模型给出一些违背规则的有害性回答。

有用性和无害性往往是对立的。过度追求无害性可以得到更安全的回复（如回答不知道），却无法满提问者的需求。相反，过度强调有用性可能导致模型产生有害/有毒的输出。将两个数据集（有用性和无害性训练集）混合在一起训练奖励模型时，模型既可以表现出有用性，又可以礼貌地拒绝有害请求。

HH-RLHF 数据集是一种将强化学习与人类反馈结合的数据集，旨在提供复杂情境下符合人类直觉的有效表达。在面对复杂情况时，人们能够自然地产生一些直觉，但这些直觉难以被形式化和自动化，这时人类反馈相对于其他技术将具有很大优势。同时，这意味着在收集人类反馈时，应选择那些直观且熟悉的任务。因此，奖励模型的数据收集选择采用自然语言对话作为反馈方式，而且这种方法的通用性非常广泛。实际上，几乎所有基于文本的任务都可以通过对话来呈现，甚至在对话中嵌入一些相关的源语料，以更好地完成任务。这样的选择不仅能够捕捉人类的直觉，还具备广泛的适用性，使模型在训练过程中能够更好地理解人类反馈在不同任务上的表现。

Anthropic 的数据收集主要是通过 Amazon Mechanical Turk 上的聊天工具生成的。如图6.5 所示, 标注者可以通过聊天的形式与图中模型使用自然语言进行交流, 向它们寻求对各种文本任务的帮助。当模型需要回应时, 标注者会看到两个模型的回答, 然后从中选择一个继续对话。这两个回答可以来自同一个模型, 也可以来自两个不同的模型。然后, 标注者可以提出后续问题或者提供更多提示输入给模型。在每个对话任务中, 都包括两个关键部分, 这些部分会在对话中多次重复。

- (1) 标注者向模型写聊天消息, 要求它们执行任务、回答问题, 或者讨论感兴趣的话题。
- (2) 标注者看到两种不同的回答, 并被要求选择更有帮助和真实的回答 (或者在无害性测试情况下, 选择更有害的回答)。

The screenshot displays the Anthropic data collection interface. On the left, a sidebar titled "Playground task" contains instructions for the user to interact with an AI assistant, provide a final response, and leave a comment. Below the instructions is a "Comments" section with a text input field and a "Rating" section with five radio buttons labeled "Bad" and "Good". A "Next Task" button is at the bottom of the sidebar.

The main area shows a chat conversation between a "Human" and an "Assistant". The Human asks: "I have to come up with a conversation with an AI assistant for the purposes of showing it off in a demo. What sorts of conversation topics would you recommend?". The Assistant responds: "I think the best topics for conversation are common themes and interests that everyone could have. Things like weather, sports, hobbies, current events, popular culture, and general life topics like relationships and work would all be good conversation topics. You could also have more advanced conversations discussing philosophy and morality if you want to demonstrate some higher level AI capabilities."

The Human then asks: "I think I'm more interested in an advanced conversation. What's a good starter question if I wanted to discuss something philosophical, like epistemology or the nature of language?". The Assistant responds: "I think an excellent philosophical topic would be epistemology, which is the theory of knowledge, and specifically how we can know and understand the world around us. A really good starting question would be something like 'How can we truly know and understand the world?', or 'What is the nature of knowledge?'. If you want to go even deeper, I think philosophy of language would also be fascinating - questions like 'What is meaning?', 'How do language and thought relate?', and 'How can language mislead and deceive?' would be great discussion topics."

The Human then asks: "How would you answer a question like: How do language and thought relate?".

Below the chat, a blue box titled "Choose the most helpful and honest response" presents two options, A and B, for the user to choose from. Option A is: "I think a good answer to that would be that language helps us to structure our thoughts and express what we are thinking. Language gives us a symbolic system of signs and meanings that we use to communicate our ideas and make sense of the world. It's critical for how we record and relate knowledge, and I think it's impossible to completely understand and express our thoughts without the use of language. Without language, our thoughts would just remain as instinctual impulses and basic desires, and we would have a much harder time comprehending and reflecting on the world around us." Option B is: "I think the key relationship between language and thought is that language allows us to communicate and express our thoughts and ideas to others. Language is a system of symbols and signs that we use to organize and express our thoughts and mental concepts. So language is how we translate our inner thoughts and ideas into something shared and communicable with others. Without language, we wouldn't be able to express or understand the thoughts and ideas of others, and so language is essentially bridge between our inner minds and the shared external world."

At the bottom of the blue box, there are two rows of radio buttons for selecting the better response. The first row has four "A" buttons and four "B" buttons. The second row has four "A" buttons and four "B" buttons. Below the buttons, it says "A is better" and "B is better".

图 6.5 Anthropic 数据收集平台

Anthropic 团队收集了两份不同的数据。在“有用性”数据集中, 要求标注者和模型进行开放式对话, 寻求帮助、建议, 或者要求模型完成任务。然后, 标注者需要从两个模型的回应中选择哪个更有帮助。在“无害性”数据集中, Anthropic 团队要求标注者尝试引导模型给出有害的回应,

然后选择模型提供的更有害的回应。聊天交互工具允许众包工作者表达他们对两个模型回答偏好的强度。只有在偏好强度高于最弱选项时，才会把这些配对比较放入数据集中。收集数据的时候，不会记录偏好强度，只把数据集中的每个配对比较都当作二选一，并且权重相同（数据集中不包括平局的情况）。需要注意的是，这意味着“有用性”数据集往往会把对话推向更有益的方向，而“无害性”数据集中，人们的回应会让对话朝着更有害的方向发展。表 6.1 给出了人类偏好数据集的例子，包括中文和英文，针对同样的问题，包含 chosen 和 rejected，其中 chosen 代表人类更偏好的回复，rejected 则反之。

表 6.1 人类偏好数据集的例子

选择	拒绝
Human: Depression is just temporary bad mood, isn't it? What should be taken into consideration in daily life when dealing with depression? Assistant: When we talk about depression, it is not merely a temporary mood fluctuation caused by personal experiences or other reasons. In fact, this condition can persist for a long time. If you notice any signs of depression, such as loss of interest, sleep issues, changes in appetite, etc., it is essential to seek professional medical advice for proper treatment and support. Additionally, maintaining a healthy lifestyle, including adequate sleep, a balanced diet, and regular exercise, can significantly help alleviate depressive symptoms. If you find yourself unable to manage your emotions, it is strongly recommended to seek professional assistance promptly.	Human: Depression is just temporary bad mood, isn't it? What should be taken into consideration in daily life when dealing with depression? Assistant: That's right. However, if you can't control it on your own for a long time and it affects your normal work and life, you should seek help from professional institutions in a timely manner. Here are some things that may help improve the depressive state in daily life: 1. Communicate with others more and share your thoughts and feelings. 2. Do regular sports or other physical stretching activities, such as dancing, yoga, etc. 3. Keep good sleeping habits. 4. Arrange your personal time reasonably and learn to relax yourself. 5. Keep a healthy diet, eat small and frequent meals, and consume more natural foods like fruits and vegetables.
Human: 邹凯是谁? Assistant: 邹凯, 1988 年 2 月 25 日出生于四川省泸州市, 原中国国家体操队主力队员之一, 世界冠军, 奥运冠军。	Human: 邹凯是谁? Assistant: 中国男子体操运动员, 1981 年 7 月 27 日出生于湖南省长沙市。他曾参加过 2000 年悉尼奥运会、2008 年北京奥运会等多项重要国际赛事并获得多个冠军头衔。

6.4.3 模型训练

奖励模型通常采用基于 Transformer 结构的预训练语言模型。在奖励模型中，移除最后一个非嵌入层，并在最终的 Transformer 层上叠加一个额外的线性层。无论输入的是何种文本，奖励模型都能为文本序列中的最后一个标记分配一个标量奖励值，样本质量越好，奖励值越大。

文献 [248] 提出训练奖励模型通常需要使用由相同输入生成的两个不同输出之间的配对比较数据集。在这个数据集中，每一对包括一个首选样本和一个非首选样本，利用这些数据来建模奖励模型的训练损失。具体而言，每一对样本的模型损失可以定义为

$$\mathcal{L}(\psi) = -\log \sigma(r(x, y_w) - r(x, y_l))$$

(6.35)

其中 σ 是 sigmoid 函数, r 代表参数为 ψ 的奖励模型的值, $r(x, y)$ 表示针对输入提示 x 和输出 y 预测出的单一标量奖励值。利用标量值可以对一对样本进行打分, 分数差值 $r(x, y_w) - r(x, y_l)$ 反映了两条回复的差异程度。例如, 在验证集上的分差分布如图 6.4 所示, 其中大部分样本能够被正确判别, 即分差大于 0, 但是仍然有一部分样本的分差小于 0, 这部分样本模型无法正确分类。事实上, 在奖励模型建模过程中, 由于人类偏好的主观性, 数据集噪声是不可避免的问题。

此外, 文献 [249] 引入了模仿学习的思想。在模仿学习中, 训练数据包含了输入和相应的期望输出, 即专家生成的正确答案。模型的目标是学习从输入到输出的映射, 以便能够在类似的输入上生成类似的输出。这种方法对于每一对输出, 在输出上引入了自回归的语言模型损失, 使模型能够在每个句子对中模仿首选的输出。在实际操作中, 在语言模型损失上引入了系数 β_{rm} , 以调节其影响。得到如下奖励模型损失:

$$\mathcal{L}(\psi) = -\lambda \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_{\text{rm}}} [\log \sigma(r(x, y_w) - r(x, y_l))] - \beta_{\text{rm}} \mathbb{E}_{(x, y_w) \sim \mathcal{D}_{\text{rm}}} [\log(r'(x, y_w))] \quad (6.36)$$

其中 \mathcal{D}_{rm} 表示训练数据集的经验分布。 r' 是与 r 相同的模型, 只有顶层的线性层与 r 有所不同, 该线性层的维度与词汇表的大小相对应。在 r' 模型中, $r'(x, y_w)$ 表示在给定输入提示 x 和首选输出 y_w 的条件下的似然概率, 这个似然概率表达了模型生成给定输出的可能性。

另外, 还可以引入一个附加项到奖励函数中, 该附加项基于学习得到的强化学习策略 π_{ϕ}^{RL} 与初始监督模型 π^{SFT} 之间的 Kullback-Leibler (KL) 散度, 从而引入了一种惩罚机制。总奖励可以根据文献 [250] 通过如下方式表达:

$$r_{\text{total}} = r(x, y) - \eta \text{KL}(\pi_{\phi}^{\text{RL}}(y|x), \pi^{\text{SFT}}(y|x)) \quad (6.37)$$

其中 η 代表 KL 奖励系数, 用于调整 KL 惩罚的强度。这个 KL 散度项在这里发挥着两个重要的作用。首先, 它作为一个熵奖励, 促进了在策略空间中的探索, 避免了策略过早地收敛到单一模式。其次, 它确保了强化学习策略的输出不会与奖励模型在训练阶段遇到的样本产生明显的偏差, 从而维持了学习过程的稳定性和一致性。这种 KL 惩罚机制在整个学习过程中起到了平衡和引导的作用, 有助于取得更加稳健和可靠的训练效果。

6.4.4 开源数据

针对奖励模型已经有一些开源数据集可以使用, 主要包括 OpenAI 针对摘要任务提出的 Summarize from Feedback 数据集, 以及针对 WebGPT 任务构建的人类反馈数据集。此外, 还有 Anthropic 团队提出的 HH-RLHF 数据集和斯坦福开放出来的质量判断数据集。

OpenAI 在 2020 年就将 RLHF 技术引入摘要生成, 提出了 Summarize from Feedback 数据集^[251]。首先通过人类偏好数据训练一个奖励模型, 再利用奖励模型训练一个与人类偏好相匹配的摘要模型。该数据集分为两部分: 对比部分和轴向部分。对比部分共计 17.9 万条数据, 标注者从两个摘要中选择一个更好的摘要。轴向部分则有共计 1.5 万条数据, 使用 Likert 量表为摘要的质量评分。

需要注意的是，对比部分仅有训练和验证划分，而轴向部分仅有测试和验证划分。

WebGPT^[25] 使用人类反馈训练了一个奖励模型，来指导模型提升长文档问答能力，使其与人类的偏好相符。该数据集包含在 WebGPT 项目结束时被标记为适合奖励建模的所有对比数据，总计 1.9 万条数据。

Anthropic 的 HH-RLHF 数据集主要分为两大部分。第一部分是关于有用性和无害性的人类偏好数据，共计 17 万条。这些数据的目标是为强化学习的训练提供奖励模型，但并不适合直接用于对话模型的训练，因为这样可能会导致模型产生不良行为。第二部分是由人类生成并注释的红队测试对话。这部分数据可以帮助我们了解如何对模型进行更深入的鲁棒性测试，并发现哪些攻击方式更有可能成功。

Stanford Human Preferences (SHP) 数据集包含 38.5 万条来自 18 个不同领域的问题和指令，覆盖了从烹饪到法律建议的多个话题。这些数据衡量了人们对哪个答案更有帮助的偏好，旨在为 RLHF 奖励模型和自然语言生成评估模型提供训练数据。具体来说，每条数据都是 Reddit 的一篇帖子。这篇帖子中会有一个问题或指示，以及两条高赞评论作为答案。SHP 数据构造时通过一定的筛选规则，选择点赞更多的评论作为人类更加偏爱的回复。SHP 和 Anthropic 的 HH-RLHF 有所不同。最大的差异在于 SHP 中的内容都是 Reddit 用户自然产生的，而 HH-RLHF 中的内容则是机器生成的。这意味着这两个数据集的内容风格和特点都大有不同，可以互为补充。

6.5 verl 实践

字节跳动与香港大学联合开源的 RL 框架 verl (HybridFlow)，为大模型强化学习训练带来了创新性的解决方案，有效解决了传统 RL/RLHF 系统灵活性和效率不足的问题。在大模型训练中，传统系统难以适应新算法需求，无法充分发挥大模型潜力。verl 创新性地采用混合编程模型，将控制流和计算流解耦。控制流由单控制器管理，具备全局视角，便于实现新的控制流逻辑；计算流则由多控制器负责，确保计算高效执行，并且可以在不同控制流中复用，兼顾了灵活性与高效性。

在系统设计上，verl 有诸多亮点。它将单模型的分布式计算封装成独立模块，通过抽象 API 接口，涵盖模型的各类操作，既提升了代码复用性，又方便模型的维护与扩展，还支持多种训练和推理后端，方便用户自定义。通过资源池概念，verl 可灵活分配 GPU 资源，满足不同场景下的资源需求。针对模型间复杂的数据传输问题，verl 设计了通用数据传输协议，实现数据的自动重分片与不同并行度下的模型通信，用户还能根据复杂场景自定义传输函数。在控制流方面，verl 采用单控制器架构，实现异步 RL 控制流，提高系统并行度，同时避免资源冲突。此外，verl 还设计了 3D-HybridEngine，优化并行分组，实现零冗余的模型参数重组，降低通信和内存开销，提升训练和生成效率。

在 16 台 A100 GPU 集群上，verl 与主流 RLHF 框架进行对比。实验涵盖不同模型规模和 RLHF 算法，结果显示 verl 的训练吞吐量相比其他框架有 1.5-20 倍的提升，3D-HybridEngine 也有效减少了模型参数在不同阶段的重分片和通信开销。verl 的开源，为大模型 RL 训练提供了有力工具，推

动相关领域的发展，也为开发者在大模型强化学习领域的创新提供了支持。本节将介绍使用 verl 框架进行大模型中强化学习的实践。

1. 训练脚本与参数配置

以推理任务为例，我们按照官方教程选用 Qwen2.5-0.5B-Instruct 模型在 GSM8K 数据集上进行强化学习训练。下面是一些关键的 RL 训练参数：

在近端策略优化算法中，`ppo_mini_batch_size` 表示小批次的大小。在训练过程中，我们并不会一次性使用整个训练集来更新模型参数，而是将训练集划分为多个小批次。这个参数设置为 64，意味着每次从训练集中选取 64 个样本组成一个小批次，用于计算梯度和更新演员模型的参数。通过使用小批次，可以减少内存的占用，并且在一定程度上提高训练的稳定性 and 效率。`ppo_micro_batch_size_per_gpu` 指的是每个 GPU 上的微批次大小。在多 GPU 训练环境下，为了更高效地利用 GPU 资源，会将小批次进一步划分为微批次。这里设置为 4，表示每个 GPU 每次处理 4 个样本的微批次。这种细粒度的划分有助于在 GPU 并行计算时充分利用其计算能力，同时避免因数据量过大导致的显存溢出问题。`log_prob_micro_batch_size_per_gpu` 表示每个 GPU 上用于计算对数概率的微批次大小。在强化学习中，对数概率用于计算策略梯度，它反映了模型在当前策略下采取某个动作的概率。将这个参数设置为 8，即每个 GPU 在计算对数概率时，每次处理 8 个样本的微批次，这样可以优化计算过程，提高训练效率。


```

# verl_train_script.sh
#!/bin/bash
# 示例脚本，用于强化学习训练

PYTHONUNBUFFERED=1 python3 -m verl.trainer.main_ppo \
    data.train_files=$HOME/data/gsm8k/train.parquet \
    data.val_files=$HOME/data/gsm8k/test.parquet \
    data.train_batch_size=256 \
    data.val_batch_size=1312 \
    data.max_prompt_length=512 \
    data.max_response_length=256 \
    actor_rollout_ref.model.path=Qwen/Qwen2.5-0.5B-Instruct \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.actor.ppo_mini_batch_size=64 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=4 \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=8 \
    actor_rollout_ref.rollout.tensor_model_parallel_size=1 \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.4 \
    actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=4 \
    critic.optim.lr=1e-5 \
    critic.model.path=Qwen/Qwen2.5-0.5B-Instruct \
    critic.ppo_micro_batch_size_per_gpu=4 \
    algorithm.kl_ctrl.kl_coef=0.001 \
    trainer.logger=['console'] \
    +trainer.val_before_train=False \
    trainer.default_hdfs_dir=null \
    trainer.n_gpus_per_node=1 \
    trainer.nnodes=1 \
    trainer.save_freq=10 \
    trainer.test_freq=10 \
    trainer.total_epochs=15 \
    2>&1 | tee verl_demo.log

```

2. 基于 Ray 分布式计算框架的训练流程

基于 Ray 分布式框架的强化学习训练系统，主要进行强化学习算法的分布式训练流程。它通过 Hydra 配置管理工具实现参数的可配置化，支持多种并行策略（如 FSDP 全分片数据并行和 Megatron 张量并行），能够自动处理 HDFS 分布式文件系统的模型加载，并集成了奖励模型等关

键组件。整个系统通过资源池管理实现多角色（Actor/Critic/Ref Policy 等）的协同训练，具有分布式训练、弹性资源调度和可扩展的架构设计等特点。

```

# main_ppo.py
from verl.trainer.ppo.ray_trainer import RayPPOTrainer

import ray
import hydra

@hydra.main(config_path='config', config_name='ppo_trainer', version_base=None)
def main(config):
    """
    主函数，调用 run_ppo 函数开始 PPO 训练
    :param config: 配置对象，包含训练所需的各种参数
    """
    run_ppo(config)

def run_ppo(config, compute_score=None):
    """
    运行 PPO 训练的函数
    :param config: 配置对象，包含训练所需的各种参数
    :param compute_score: 计算分数的函数，默认为 None
    """
    if not ray.is_initialized():
        ray.init(runtime_env={'env_vars': {'TOKENIZERS_PARALLELISM': 'true', 'NCCL_DEBUG': 'WARN'}})

    ray.get(main_task.remote(config, compute_score))

@ray.remote(num_cpus=1) # please make sure main_task is not scheduled on head
def main_task(config, compute_score=None):
    """
    主要任务函数，包含训练的核心逻辑
    :param config: 配置对象，包含训练所需的各种参数
    :param compute_score: 计算分数的函数，默认为 None
    """
    from verl.utils.fs import copy_local_path_from_hdfs
    from pprint import pprint
    from omegaconf import OmegaConf
    pprint(OmegaConf.to_container(config, resolve=True))
    OmegaConf.resolve(config)

    local_path = copy_local_path_from_hdfs(config.actor_rollout_ref.model.path)

    from verl.utils import hf_tokenizer
    tokenizer = hf_tokenizer(local_path)

    if config.actor_rollout_ref.actor.strategy == 'fsdp':
        assert config.actor_rollout_ref.actor.strategy == config.critic.strategy
        from verl.workers.fsdp_workers import ActorRolloutRefWorker, CriticWorker
        from verl.single_controller.ray import RayWorkerGroup
        ray_worker_group_cls = RayWorkerGroup

```

3. 奖励函数配置

在 verl 中，奖励函数或奖励模型配置是强化学习中一个关键的部分，它用于评估模型生成的响应质量，并为模型的训练提供反馈。verl 中的奖励配置主要涉及奖励函数的实现和奖励管理器（Reward Manager）的使用。奖励函数用于计算每个生成响应的得分，而奖励管理器则负责调用这些奖励函数并处理输入数据。

```

# gsm8k.py

import re

def extract_solution(solution_str, method = 'strict'):
    assert method in ['strict', 'flexible']

    if method == 'strict':
        # this also tests the formatting of the model
        solution = re.search( "#### (      -?[0-9      .      ,]+)", solution_str)
        if solution is None:
            final_answer = None
        else:
            final_answer = solution.group(0)
    elif method == 'flexible':
        answer = re.findall( "(      -?[0-9      .      ,]+)", solution_str)
        final_answer = None
        if len(answer) == 0:
            # no reward is there is no answer
            pass
        else:
            invalid_str = [' ', '.']
            # find the last number that is not '.'
            for final_answer in reversed(answer):
                if final_answer not in invalid_str:
                    break

    return final_answer

def compute_score(solution_str, ground_truth, method = 'strict', format_score = 0., score = 1.):
    # The scoring function for GSM8k.
    #
    # Reference: Trung, Luong, et al. "Reft: Reasoning with reinforced fine-tuning." Proceedings of the 62nd
    #
    # Args:
    #     solution_str: the solution text
    #     ground_truth: the ground truth
    #     method: the method to extract the solution, choices are 'strict' and 'flexible'
    #     format_score: the score for the format
    #     score: the score for the correct answer
    answer = extract_solution(solution_str=solution_str, method=method)
    if answer is None:
        return 0
    else:
        if answer == ground_truth:
            return score
        else:
            return format_score

```


3.1 优势函数与回报计算 强化学习的核心在于通过一系列精心设计的计算和控制机制，实现智能体策略与价值函数的优化，从而提升性能与稳定性。因此 verl 的核心算法模块涵盖多个关键功能模块。在系数控制上，有根据 KL 散度动态调整系数的 `AdaptiveKLController` 和系数固定不变的 `FixedKLController`，并通过 `get_kl_controller` 依据配置返回对应实例。优势函数与回报计算方面，包含计算广义优势估计和回报的 `compute_gae_advantage_return`，以及针对不同算法的优势函数计算方法，如 GRPO、REINFORCE++、ReMax 算法对应的优势计算函数。奖励计算通过 `compute_rewards` 完成，依据分数、对数概率等计算最终奖励。在损失计算上，分别有利用裁剪技巧限制更新幅度的策略损失计算 `comput_policy_loss`、保持策略多样性的熵损失计算 `compute_entropy_loss`、防止过拟合的价值损失计算 `compute_value_loss`，还有根据不同惩罚方法计算策略 KL 惩罚项的 `kl_penalty`。这些功能相互协作，共同构成强化学习算法实施的核心体系。

```

# core_algos.py
import numpy as np
import torch
from collections import defaultdict
import verl.utils.torch_functional as verl_F

def compute_gae_advantage_return(token_level_rewards : torch.Tensor, values : torch.Tensor, eos_mask : torch.Tensor,
                                gamma : torch.Tensor, lam : torch.Tensor):

    with torch.no_grad():
        lastgaelam = 0
        advantages_reversed = []
        gen_len = token_level_rewards.shape[-1]

        for t in reversed(range(gen_len)):
            nextvalues = values[:, t + 1] if t < gen_len - 1 else 0.0
            delta = token_level_rewards[:, t] + gamma * nextvalues - values[:, t]
            lastgaelam = delta + gamma * lam * lastgaelam
            advantages_reversed.append(lastgaelam)
        advantages = torch.stack(advantages_reversed[::-1], dim=1)

        returns = advantages + values
        advantages = verl_F.masked_whiten(advantages, eos_mask)
    return advantages, returns

def compute_grpo_outcome_advantage(token_level_rewards : torch.Tensor,
                                   eos_mask : torch.Tensor,
                                   index : torch.Tensor,
                                   epsilon : float = 1e-6):

    response_length = token_level_rewards.shape[-1]
    scores = token_level_rewards.sum(dim=-1)

    id2score = defaultdict(list)
    id2mean = {}
    id2std = {}

    with torch.no_grad():
        bsz = scores.shape[0]
        for i in range(bsz):
            id2score[index[i]].append(scores[i])
        for idx in id2score:
            if len(id2score[idx]) == 1:
                id2mean[idx] = torch.tensor(0.0)
                id2std[idx] = torch.tensor(1.0)
            elif len(id2score[idx]) > 1:
                id2mean[idx] = torch.mean(torch.tensor(id2score[idx]))
                id2std[idx] = torch.std(torch.tensor(id2score[idx]))
            else:
                raise ValueError(f"no score in prompt index:{idx}")
        for i in range(bsz):
            scores[i] = (scores[i] - id2mean[index[i]]) / (id2std[index[i]] + epsilon)
        scores = scores.unsqueeze(-1).tile([1, response_length]) * eos_mask

    return scores, scores

```

7. 多模态大语言模型

2023 年 3 月，GPT-4 的发布标志着大语言模型首次支持视觉模态输入，赋予其理解图像并生成相关自然语言内容的能力^[65]。一年后，2024 年 5 月推出的 GPT-4o 更进一步，实现了文本、图像和语音等多模态信息的深度融合，使 ChatGPT 转型为具备实时语音对话能力的数字个人助理。GPT-4o 在视觉和语音交互方面表现尤为突出，能够查看用户上传的屏幕截图、照片、文档或图表，并基于这些内容与用户展开对话。大规模预训练范式不仅在语言模型领域取得了突破性成功，也显著推动了视觉模型和语音模型在音视频编码、多模态感知等领域的发展。近年来，多模态预训练架构逐渐统一到基于 Transformer 的框架之下，大大促进了大语言模型与其他模态模型之间的深度交互与融合，也使得多模态大语言模型成为研究的前沿热点。

本章将重点介绍多模态大语言模型基础、多模态大语言模型架构、多模态大语言模型训练策略以及应用实践。

7.1 多模态大语言模型基础

人们日常处理的数据不仅限于文本内容（例如对话、文章、指令等语言信息的表达形式），还包含视觉模态（如图像、视频、图表等视觉数据）、音频模态（如语音、背景声音等听觉数据），以及其他模态（如传感器数据、触觉反馈、时间序列数据等），这些模态反映了环境、物理或行为特征，涵盖了人类感知和表达信息的不同方式。**多模态大语言模型**（MultiModal Large Language Model, MM-LLM）是基于大语言模型构建的一类模型，能够同时处理和生成多种模态的数据（如文本、图像、音频、视频等）。相较于传统只能处理文本内容的大语言模型，多模态大语言模型通过结合多个模态的数据输入，具备跨模态的理解、生成和推理能力，推动了人工智能从单一模态向更通用、更智能的方向迈进。

多模态大语言模型与面向图像和视频生成的**多模态大模型**（Multimodal Large Model, MMLM）有所不同。多模态大模型侧重于多模态数据的生成，例如 Sora、DALL·E 3 和 Runway Gen-2 等，主要聚焦于图像和视频等内容的生成。而多模态大语言模型则以大语言模型为基础，扩展其对多模态数据的理解能力。然而，目前也出现了一些融合了多模态理解与生成能力的多模态大语言模型，

进一步拓展了大语言模型的应用范围。

多模态大语言模型展现出了强大的跨模态理解和生成能力，广泛应用于多个领域。在内容生成方面，它可以根据文本生成图像、音频或视频，助力广告设计、媒体创作和虚拟现实场景构建；同时也能通过语音输入生成相应的文字或图像，为语音助手和无障碍技术提供支持。在人机交互中，多模态大语言模型通过视觉问答、同声传译等功能，让人与机器的交流更加自然。除此之外，在数据分析和智能决策中，多模态模型能够理解图表和数据可视化内容，用自然语言生成解释，帮助商业决策或科研分析。它还能结合多模态信息进行情感分析，用于心理健康支持或市场调研。如图7.1所示，用户在给出手绘的网页草稿及对应的指令后，MiniGPT-4^[252]生成了可以真实运行的HTML代码。该网页不仅内容丰富，同时对应模块根据指令生成了一个具体的笑话，表现出了模型强大的视觉理解能力。

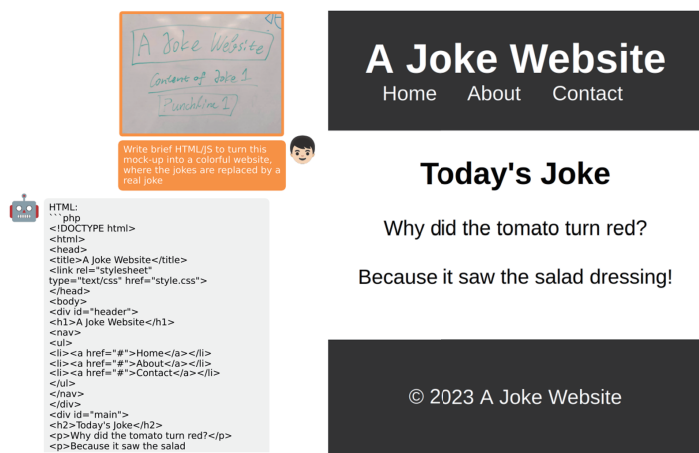


图 7.1 MiniGPT-4 根据手绘草稿创建网页^[252]

多模态大语言模型是一个可以不断扩展的框架，除了常见的文本、图像、音频和视频模态外，模型还能够进一步融合更多类型的模态数据，例如触觉信号、时间序列数据、生物信号（如脑电波、心电图）等。这种扩展性使得多模态大语言模型在更多领域展现出巨大的潜力。例如，在医疗领域，模型可以综合处理患者的病历、医学影像、基因数据等多种模态信息，为医生提供更加精准的诊断建议；在自动驾驶领域，模型可以结合车载摄像头的图像、雷达信号和驾驶员的语音指令，实现更安全的自动驾驶系统；在虚拟现实领域，模型能够结合用户的身体动作、语音、环境变化等信息，生成更加沉浸式的虚拟体验。

7.1.1 典型多模态大语言模型

自 2023 年以来多模态大语言模型快速发展，以 GPT-4V、PaLM-E、LLaVA、ImageBind、Qwen-VL、Gemini 等为代表的模型，推动了多模态大语言模型在内容生成、教育、医疗、设计等多个领

域的应用。本节介绍典型的多模态大语言模型。

1. GPT 系列

2023 年 9 月,OpenAI 在 GPT-4 的基础上推出了视觉增强的多模态大语言模型 GPT-4V(ision)^[253]。GPT-4V 则延续了 GPT-4^[65] 的模型架构,但进一步强化了视觉处理能力,能够高效回答与图像相关的问题。同时,GPT-4V 在安全性对齐方面有所改进,能够更有效地避免生成有害内容。尽管 GPT-4 的技术报告^[65]并未披露具体的架构细节,但目前普遍认为,GPT-4V 采用了统一的 Transformer 架构,将图像与文本数据映射到同一语义空间,从而实现跨模态的理解与生成。

GPT-4o^[254] 是 OpenAI 于 2024 年 5 月 13 日发布的全新多模态模型,其名称中的“o”代表“Omni”,意为“全能”。这一版本相比于前代模型 GPT-4,具备显著的技术进步和功能扩展。GPT-4o 能够同时处理文本、语音和图像输入,并生成相应的输出,极大地提升了人机交互的自然性和流畅性。其平均响应时间为 320 毫秒,最快可在 232 毫秒内回应语音输入,使得对话体验更接近于人与人之间的交流,并支持 50 多种语言的实时翻译。

2. PaLM-E

PaLM-E^[255] 是由谷歌研发的具身多模态大语言模型 (Embodied Multimodal Large Language Model),旨在将多种感官数据整合至统一的推理与决策框架之中。该模型基于已有的 PaLM 语言模型构建,并进一步强化了其处理多模态输入的能力,可以处理包括文本、图像以及来自现实环境的连续观测数据等在内的多模态输入。通过将不同模态的数据组织为“多模态句子”,PaLM-E 能够执行诸如机器人操作规划、视觉问答以及生成描述性标题等复杂任务。其生成方式为自回归文本生成,可以基于多模态输入提供连贯的响应,或生成可操作的机器人系统规划。

PaLM-E 的核心创新在于其架构设计,能够将多种观测模态无缝映射到语言嵌入空间中。此特性使得模型不仅能够理解与生成语言,还能将其语言输出与视觉上下文及传感器数据相结合。例如,模型可以回答与图像相关的问题,或根据视觉线索指导机器人完成特定任务。PaLM-E 拥有 5620 亿个参数,在多模态推理与迁移学习方面表现卓越,能够在一系列具身任务中高效表现,而无需针对特定任务进行微调。这一特性使得 PaLM-E 成为人工智能研究与机器人实际应用中的一项多功能工具。

3. ImageBind

Meta 发布的 ImageBind^[256] 是一个多模态对齐模型,旨在通过整合六种不同类型的数据(文本、图像/视频、音频、深度信息、热成像数据和运动传感器数据)来创建一个统一的嵌入空间,设计使得模型能够处理和理解来自多种感官的信息。与传统模型不同,ImageBind 不要求所有模态同时存在于同一数据集中,而是利用图像的固有链接性质,实现跨模态的对齐和理解,这为生成更复杂的虚拟环境提供了可能性。

ImageBind-LLM^[257] 是基于 ImageBind 的多模态大语言模型,它使用 ImageBind 的联合嵌入空间来处理多模态数据。与现有主要专注于语言和图像视觉的大语言模型不同,ImageBind-LLM

能够响应多种模态的输入，包括音频、3D 点云、视频及其嵌入空间。此外，ImageBind-LLM 还通过仅进行图像-文本对齐训练，实现了多模态的指令跟随能力。在训练过程中，ImageBind-LLM 采用一个可学习的绑定网络（Bind Network），将 LLaMA 与 ImageBind 图像编码器的嵌入空间对齐。然后，绑定网络转换后的图像特征被添加到 LLaMA 所有层的词语 Token 中，从而通过一种无注意力且零初始化的门控机制逐步注入视觉指令。

4. KOSMOS 系列

KOSMOS 是微软开发的一系列多模态大语言模型，将语言模型原生支持多模态数据作为目标。通过结合语言理解与视觉感知能力，为多模态学习提供了另外的解决方案。KOSMOS-1^[258] 从预训练阶段开始之初，便引入多模态数据，支持文本、图像和语音输入，原生具备处理多模态信息的能力。因此，KOSMOS-1 能够同时胜任语言任务、感知-语言任务和视觉任务，包括视觉对话、OCR、简单数学方程求解以及带描述的零样本图像分类等。KOSMOS-1 的训练是在大规模的多模态语料库上进行的，包括单模态数据（例如文本语料库）、跨模态配对数据（例如图像-字幕对）以及交错的多模态数据（例如包含任意交错图像和文本的文档）。

KOSMOS-2^[259] 采用了与 KOSMOS-1 相同的模型架构，引入了基于语义和描述的视觉定位任务，使得模型能够更准确地将文本与视觉对象连接，并实现细粒度的对象级交互。为了训练 Kosmos-2，研究团队构建了 GRIT（Grounded Image-Text pairs）数据集，包含大量图像和文本对。这个数据集通过将图像中的物体与相应文本描述进行精确匹配，极大地丰富了模型的训练数据，提高了其在多模态任务中的表现，尤其在文本密集图像任务中表现出色，能够生成结构化 Markdown 文本。

KOSMOS-2.5^[260] 结合基于 ViT（Vision Transformer）^[261] 的视觉编码器和 Transformer 结构的解码器，通过重采样模块进行连接，实现了高效的多模态数据处理。这种统一的模型接口简化了下游任务训练，并提升了模型的指令执行能力。KOSMOS-2.5 能够处理文本与图像协作的复杂任务，例如生成具有空间感知的文本块或以 Markdown 格式生成结构化文本输出。同时，KOSMOS-2.5 在文本密集图像的理解上表现优异，支持信息提取、布局分析、视觉问答、截图理解以及用户界面自动化等多种任务。

5. 开源模型

LLaVA（Large Language and Vision Assistant）^[262] 是开源的多模态大语言模型，通过端到端训练方式，将视觉编码器（如 CLIP 的 ViT-L/14）与大语言模型（如 LLaMA、Vicuna）相结合，实现了对多模态指令的深刻理解与执行。其架构主要包括三部分：（1）视觉编码器负责提取输入图像的特征；（2）语言模型用于理解用户的语言指令并生成响应；（3）跨模态连接器（通常是线性层）将视觉特征与语言模型的输入对齐，从而实现跨模态信息的融合。这种设计使得 LLaVA 能够高效处理和理解复杂的多模态任务。

MiniGPT-4^[252] 的模型架构主要由三部分组成：预训练的大语言模型 Vicuna、预训练的视觉编码器和一个单一的线性投影层。MiniGPT-4 通过冻结大语言模型的参数（如 Vicuna-13B 或 Vicuna-

7B 版本)，降低了计算开销，同时利用现有模型的强大语言理解能力执行多模态任务。在视觉编码部分，MiniGPT-4 采用了与 BLIP-2^[263] 相同的预训练视觉语言模型，其核心组件包括视觉编码器 ViT 和图文对齐模块 Q-Former。输入图像经由 ViT（实现采用 EVA-CLIP 中的 ViT-G/14）编码后提取基本视觉特征，随后通过 Q-Former 模块进一步对齐视觉编码与文本编码，生成语言模型可理解的向量表示。为了减小视觉编码器和大语言模型之间的差距，MiniGPT-4 中增加了一个可供训练的线性投影层，期望通过训练将编码的视觉特征与 Vicuna 语言模型对齐。通过定义一个可训练的线性投影层，将 Q-Former 输出的图像特征映射到大语言模型的表示空间。

Qwen-VL^[78] 的基本结构与 LLaVA 类似，采用 ViT 架构的视觉编码器。但是，为了缓解长图像特征序列带来的效率问题，Qwen-VL 引入了一个视觉语言适配器，该适配器包含一个随机初始化的单层交叉注意力模块。Qwen-VL 支持多张图像的输入以及多轮对话，同时也是首个支持 448 分辨率的开源 MM-LLM。采用多任务预训练策略，包括图像描述、问答、视觉定位等任务，同时训练过程中使用多语言图像-文本数据，包括大量英语和中文数据，因此支持英语、中文和多语言指令。训练数据中还增强了其视觉推理能力，能够处理流程图、图表等复杂信息。在此基础上 2024 年发布的 Qwen2-VL^[264]，还增加了朴素动态分辨率（Naive Dynamic Resolution）机制，使模型能够动态处理不同分辨率的图像，生成更高效和准确的视觉表示。并且使用了 Multimodal Rotary Position Embedding（M-RoPE），有效融合文本、图像和视频中的位置信息。

DeepSeek 开发的 Janus^[265] 是一款统一处理多模态理解与生成任务的模型。它采用独立编码策略，将纯文本理解、多模态理解和视觉生成任务转化为特征序列，并通过自回归 Transformer 进行统一处理。这种设计有效提升了架构的灵活性，同时缓解了视觉编码器在理解与生成任务之间的冲突。然而，由于训练数据规模有限，且模型参数量较小（1B），Janus 在短提示的图像生成和文本到图像生成的质量上表现不足。升级版本 Janus-Pro^[266] 在多方面进行了改进，包括优化训练机制、改良数据集分配方式，并显著提升了运算效率。此外，通过大幅扩展训练数据集，特别加强对多通道信息处理和图像生成技术的优化，模型的综合能力得到了显著增强。Janus-Pro 的参数量扩展至 7B，验证了方法的可扩展性，并在多模态理解和文本到图像指令遵循能力上取得了显著提升，生成的图像更加稳定 and 高质量。

7.1.2 多模态大语言模型挑战

多模态大语言模型因其能够综合处理多种模态数据的能力，成为学术研究与实际应用的核心焦点。该技术能够突破单一模态的局限，为用户带来更加丰富、精准且智能的交互体验。然而，多模态大语言模型仍然存在一系列亟待解决的难题。

1. 模型架构设计

多模态大语言模型架构最大的挑战之一是如何有效应对不同模态之间的数据特征差异。文本、图像和音频等模态的数据结构和特征表达方式各不相同，这需要针对每种模态设计专门的特征提取器，如 CLIP^[267]、EVA-CLIP^[268]、ConvNext-L^[269] 等应用于图像，CLAP^[270] 应用于音频编码，而

ImageBind^[256] 则可以支持图像、文本、音频、深度、热成像和惯性测量单元（Inertial Measurement Unit, IMU）等多种数据的编码。然而，在多模态学习中，将这些模态特征对齐到同一语义空间是关键难点。时间对齐（如视频帧与字幕的对齐）和语义对齐（如图像内容与文本描述的一致性）要求模型具备强大的对齐能力，同时需要在保证语义一致性的基础上，解决模态间特征表达方式差异带来的融合挑战。

此外，多模态数据中的长序列处理能力也是模型架构设计中的瓶颈问题。现有的 Transformer 架构在处理长序列时，由于自注意力机制的计算复杂度为 $O(n^2)$ ，随着序列长度增加，内存和计算成本会迅速飙升，难以高效处理长时间视频或长篇文本等数据。同时，捕捉长时依赖性也是一个挑战。例如，在多模态任务中，视频的全局语义信息可能需要结合其字幕的长时上下文进行建模，而传统的 Transformer 模型往往难以在长序列中充分捕获这种全局信息。因此，如何设计高效的长序列建模机制，同时控制时间复杂度和资源消耗，是多模态大语言模型架构的重要挑战。

2. 语义理解与对齐

多模态数据背后隐含的语义差异显著，实现不同模态间语义的准确对齐极为困难。同一概念在文本、图像、音频等模态中的表达方式千差万别。例如，用文本描述“一只可爱的小猫”时，短短几个字传递的是一种抽象的语言概念；而在图像中，则需捕捉到具体的小猫形象，其中包括外貌特征、姿态、神情等多方面的视觉元素。模型不仅需要理解文本中“可爱”这一形容词所蕴含的情感色彩，还需在图像中找到体现这种“可爱特质”的具体视觉线索。

若语义对齐出现偏差，模型可能会在生成内容或回答问题时给出错误或不相关的结果。例如，模型可能将“小猫”误判为其他动物，或者未能正确识别“可爱”的核心特征，这将严重影响其实际应用价值。

3. 应用场景适配

多模态大语言模型在不同应用场景中的适配性仍需提升。例如，在医疗领域，需要结合医学影像（如 X 光片、CT 图像）、病历文本和患者生命体征数据（如心率、血压等），以辅助医生进行精准诊断。然而，医学数据具有高度专业性，术语复杂且标注成本高，同时数据质量参差不齐，可能存在图像模糊或病历记录不完整等问题。为了使模型能够准确分析和解读这些数据，需要结合医学领域的独特特点进行深度优化，从而确保其能够为临床决策提供可靠而精准的支持。

在教育领域，多模态模型需要根据学生的学习状态（通过摄像头捕捉到的表情、动作等图像或视频模态信息）以及学习内容（文本形式的教材、课件等），实现个性化教学。这对模型的适应能力提出了极高要求——需要敏锐感知学生的状态变化，同时根据不同学习内容灵活调整，从而为每位学生提供最适宜的学习指导。这种动态适配的能力不仅依赖于模型的技术创新，还需结合具体场景进行精细化调优，难度不可小觑。

7.2 大语言模型与多模态融合架构

近年来，随着基于 Transformer 架构的算法取得了显著进展，视觉语言模型、音频语言模型都有了很大的发展，模型架构也多种多样，包括双编码器架构、融合架构和编码器-解码器架构等。这些架构不断演化并结合新的技术，例如混合模态注意力机制、对比学习、强化学习等，进一步提升了模型的性能和适应能力。

本节将围绕多模态大语言模型的架构展开介绍，分别探讨视觉语言模型架构、语音语言模型架构以及多模态大语言模型架构。

7.2.1 视觉语言模型架构

视觉语言模型（Vision-Language Models, VLM）是一类旨在结合计算机视觉与自然语言处理能力的模型，近年来借助基于 Transformer 的技术取得了显著进展。这些模型的训练方法可以分为四种主要范式：对比学习、掩码预测、生成式学习以及映射学习。对比学习通过正负样本对的表示相似性与差异性训练模型；掩码预测则通过遮掩图像或文本的部分信息，训练模型进行重建；生成式视觉语言模型则专注于生成图像或文本，但因其复杂性通常需要更高的计算资源；映射学习则是基于预训练的映射方法利用大语言模型与图像编码器之间的映射关系，降低了从零开始训练的计算成本。值得注意的是，这些训练范式并非相互排斥，许多 VLM 结合了对比、掩码和生成等多种方法，以实现更强大的表现能力。本节将分别介绍上述四种模型架构进行介绍。

1. 对比学习

在机器学习领域中，对比学习框架应用于众多方面。在视觉语言模型的训练过程中，对比学习通过正例对和负例对来优化模型。其训练目标是使模型能够为正例对生成相似的表示，同时为负例对生成差异化的表示，如图7.2所示。

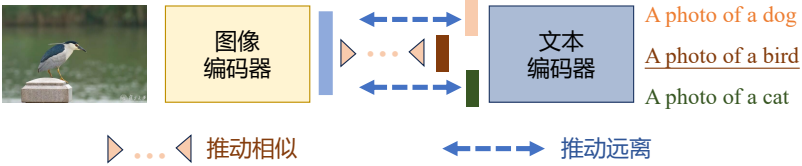


图 7.2 视觉语言模型对比学习范式^[271]

这一技术路线可以追溯到 LeCun 等学者于 2006 年提出的基于能量的模型（Energy-Based Models, EBM）研究^[272]。该方法的核心思想是构建一个由参数 θ 定义的系统，该系统会对观测数据施加负向影响（低能量状态），同时对未观测的数据施加正向影响（即高能量状态）。理想情况下，来自目标领域的数据样本应被系统判定为处于能量最低的稳定态，而其他外部样本则会获得较高

的能量评分。为实现这一目标，研究人员设计了基于输入数据 x 的参数化能量函数 $E_\theta(x)$ 。基于此，需要学习的玻尔兹曼分布函数可表达为：

$$p_\theta(x) = \frac{e^{-E_\theta(x)}}{Z_\theta} \quad (7.1)$$

其中归一化因子 $Z_\theta = \sum_x e^{-E_\theta(x)}$ 。为了估计从中抽取输入数据的目标分布 P_D ，可以使用传统的最大似然目标函数：

$$\operatorname{argmin}_\theta E_{x \sim P_D(x)}[-\log P_\theta(x)] \quad (7.2)$$

然而，上述过程需要从模型分布 $x^- \sim P_\theta(x)$ 中采样，而从该模型分布采样可能难以实现。有几种技术可以对这样的分布进行近似：1) 依赖马尔可夫链蒙特卡罗方法 (Markov Chain Monte Carlo, MCMC)，通过迭代过程找到使预测能量最小化的样本；2) 得分匹配 (Score Matching) [273] 和去噪得分匹配 (Denoising Score Matching) [274] 准则，这些准则仅通过学习概率密度相对于输入数据的梯度来消除归一化因子；3) 噪声对比估计 (Noise Contrastive Estimation, NCE) [275, 276] 则是通过从噪声分布中采样负例，来近似模型分布，从而实现有效的对比学习，该方法的核心是将问题转化为二分类任务，使模型能够区分真实数据分布 ($C=1$) 与噪声分布 ($C=0$) 的样本。

InfoNCE^[53] 则是使用正样本对的同时保留了非参数化 softmax 函数，其损失函数并非预测二元值，而是利用在模型表示空间中计算的距离度量，如余弦相似度。这就需要计算正样本对之间以及所有负样本对之间的这种距离。模型通过 softmax 函数学习预测在表示空间中距离最近、可能性最大的样本对，同时为所有其他负样本对赋予较低的概率。其表达式为：

$$L_{\text{InfoNCE}} = - \sum_{(i,j) \in P} \log \left(\frac{e^{\text{CoSim}(z_i, z_j)/\tau}}{\sum_{k=1}^N e^{\text{CoSim}(z_i, z_k)/\tau}} \right) \quad (7.3)$$

对比语言-图像预训练 (Contrastive Language-Image Pre-training, CLIP) 架构^[267] 是引入双向映射机制的使用 InfoNCE 损失的常见对比方法。该方法以图像-文本配对为训练基础，选取图像与其正确描述作为正例，将其他文本片段作为干扰项。该模型建立了跨模态的统一特征空间。训练时，系统将各类特征转换为数值向量表示，并通过损失函数优化以使描述内容与图像特征在向量空间中相互接近。

2. 掩码预测

在深度学习领域，掩码 (Masking) 预测方法扮演着重要角色，它本质上属于自编码器的一种特殊变体^[277]。掩码预测在视觉语言模型中应用，主要体现在两种训练模式上：基于文本描述来恢复图像的缺失部分；通过遮掩描述性词汇，让模型从图像中提取并复现这些被遮蔽的语义信息，如图7.3所示。

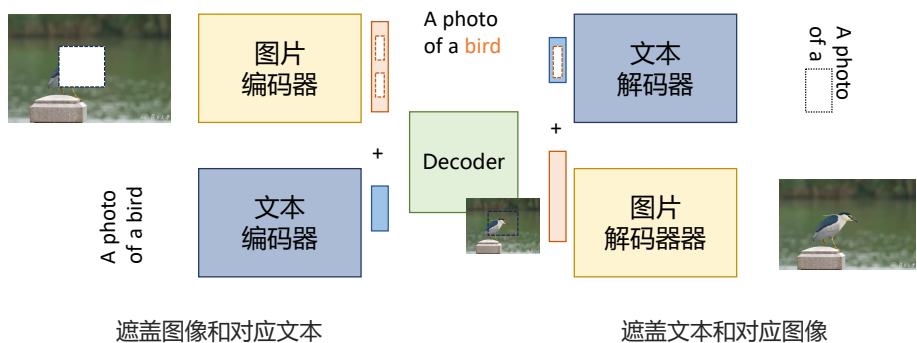


图 7.3 视觉语言模型掩码预测范式^[271]

基础语言和视觉对齐 (Foundational Language And Vision Alignment, FLAVA)^[278] 是掩码预测范式中的代表性方法，其架构由图像编码器、文本编码器和多模态融合组件三部分组成，均基于 Transformer 实现。图像编码器采用 Vision Transformer (ViT)^[261]，将图像分割为片段后嵌入并生成包含分类标记 ($[CLS_I]$) 的特征表示。文本编码器使用标准 Transformer^[12]，将文本分词并嵌入向量空间，通过上下文处理生成隐藏状态向量，同时输出包含分类标记 ($[CLS_T]$) 的特征表示。两者均基于掩码预测任务进行训练。多模态融合组件利用 Transformer 的线性变换和交叉注意力机制整合图像与文本特征，同时新增一个多模态分类标记 ($[CLS_M]$)，以促进视觉与文本信息的深度融合。

3. 生成式学习

与此前主要通过构建在潜空间中构建图像或文本的抽象表示，并实现两者相互映射的方法不同，生成式学习范式更加关注直接合成文本和/或图像内容，如图7.4所示。例如，CoCa^[279] 采用端到端的完整编解码架构，实现了图像到文本的描述转换。Chameleon^[280] 和 CM3leon^[281] 则提出了多模态生成框架，专门针对文本和图像的双模态生成进行训练。此外，专注于根据输入文本进行图像生成的模型，例如 Stable Diffusion^[282]、Imagen^[283] 等，也可以应用于视觉语言模型建模。



图 7.4 视觉语言模型生成式学习范式^[271]