



南京大學

本科畢業論文

院 系 人工智能學院

專 業 人工智能

題 目 擴散掩碼模型的分層擴放與結

構化決策引導

年 級 2025 學 號 211300024

學生姓名 石睿

指導教師 葛存菁 職 稱 助理教授

提交日期 2025年6月5日



南京大学本科毕业论文（设计） 诚信承诺书

本人郑重承诺：所呈交的毕业论文（设计）（题目：扩散掩码模型的分层扩放与结构化决策引导）是在指导教师的指导下严格按照学校和院系有关规定由本人独立完成的。本毕业论文（设计）中引用他人观点及参考资源的内容均已标注引用，如出现侵犯他人知识产权的行为，由本人承担相应法律责任。本人承诺不存在抄袭、伪造、篡改、代写、买卖毕业论文（设计）等违纪行为。

作者签名：

学号：

日期：

南京大学本科生毕业论文（设计、作品）中文摘要

题目：扩散掩码模型的分层扩放与结构化决策引导

院系：人工智能学院

专业：人工智能

本科生姓名：石睿

指导教师（姓名、职称）：葛存菁 助理教授

摘要：

扩散掩码模型逐渐被认为是足以挑战自回归模型的另一种文本建模方式，已在诸多基准上逼近甚至超越了同量级的自回归模型。同时源自其独特建模方式所带来的优势，如并行化的生成机制、良好的可控性、更具创造力的响应以及可以生成更连贯的输出等等，通常被视作现代基于因果建模的自回归大语言模型 (LLM) 的有效补充。但是该领域已有的工作集中在模型的预训练和监督微调过程中，未对扩散掩码模型的后训练做出系统性研究。而后训练被视为对齐任务或人类偏好重要的实现手段，其中强化学习算法的“探索”能力则更让 LLM 的自我学习与优化有着无限可能，并被认为是下一代人工智能突破的重中之重。为此，本文旨在探索扩散掩码模型的后训练特征，并利用其灵活可控的生成能力加速强化学习采样与推理过程。同时为了更高效地释放掩码扩散模型的双向建模与并行化能力，本文也尝试基于最新颖扩散掩码模型的建模方式 Block-Diffusion 来设计基于块级的结构化推理，通过设计适配扩散过程的强化学习建模与对应的算法，尝试把其与强化学习过程有机结合起来。为了验证该猜想，本文在多个不同的任务设置中对掩码扩散模型与自回归模型的强化学习微调过程表现进行了对比实验，并系统性分析了二者的表现。同时推理精度与速度作为假设的重要组成部分，也进行了详细的实验分析。

关键词：自然语言处理；强化学习；扩散模型；形式语言

南京大学本科生毕业论文（设计、作品）英文摘要

THESIS: Hierarchically Scaled Masked Diffusion for Structured Decision Guidance

DEPARTMENT: School of Artificial Intelligence

SPECIALIZATION: Artificial Intelligence

UNDERGRADUATE: Rui Shi

MENTOR: Assistant Professor Cunjing Ge

ABSTRACT:

Masked diffusion models have recently emerged as a promising alternative to autoregressive architectures for text modeling, achieving comparable or even superior results on a variety of benchmarks. However, prior work has focused almost exclusively on pre-training and supervised fine-tuning, and a systematic study of post-training for diffusion-based masked models is still lacking. Post-training is widely regarded as a critical means of alignment or preference learning, and the exploratory capabilities of reinforcement learning (RL) are seen as offering limitless opportunities for LLMs to self-improve—indeed, this is often identified as the next major frontier in AI research. In this work, I investigate the characteristics of masked diffusion models during post-training, and I exploit their flexible, controllable generation to accelerate both RL sampling and inference. To fully leverage the bidirectional modeling and parallelization strengths of masked diffusion, I attempt to introduce a block-structured reasoning scheme based on the novel Block-Diffusion paradigm, tightly integrating it with the RL loop. To validate this approach, I perform a series of comparative RL fine-tuning experiments on both masked diffusion and autoregressive models across multiple task settings, and I provide a systematic analysis of their relative performance. I also conduct detailed experiments examining inference accuracy and latency—key metrics in our hypothesis.

KEYWORDS: Natural Language Processing; Reinforcement Learning; Diffusion Model;
Formal Language

目 录

目 录	V
第一章 绪论	1
1.1 研究背景	1
1.2 研究意义	2
1.3 论文结构	4
第二章 相关工作	7
2.1 掩码扩散模型：全新的语言模型建模	7
2.1.1 离散时间建模	8
2.1.2 连续时间建模	11
2.2 后训练：基于强化学习的微调	12
2.2.1 传统语言模型后训练设定与算法	12
2.2.2 大语言模型后训练设定、方法论与算法	13
2.2.3 强化学习采样优化	18
第三章 研究方法	21
3.1 案例研究	21
3.2 基于块级离散扩散模型的分层强化学习建模	22
3.2.1 分层扩散模型概率建模	23
3.2.2 分层离散扩散模型强化学习建模	24
3.3 流水线设计	27
3.3.1 奖励模型建模	28
3.3.2 强化学习微调	31
3.3.3 推理优化：测试时间扩散	31

第四章 实验分析 & 结论	33
4.1 多学科问答基准测试	34
4.2 扩散模型生成速度与质量	37
4.3 有效性验证与消融实验	38
4.4 结论	38
第五章 总结与展望	41
5.1 不足之处	41
5.2 强化学习在 LLM 后训练中是否真正有效? 是否能通向 AGI?	41
5.3 扩散模型所代表的非自回归模型能否挑战自回归模型?	43
参考文献	45
致 谢	51
附录 A 案例分析 1 & 2	53
附录 B 数据格式	55
附录 C 模型输出	59

第一章 绪论

1.1 研究背景

人工智能的发展几经沉浮，从以知识表示为代表的符号主义，到在少量精心构建的数据和特征中通过机器学习算法抽取到模式并做出预测。研究者们不断从过去的方法中发现不足，完善并设计出全新且适应现实任务的算法。如今 AI 步入了算力充沛、大规模数据可达的时代，基于广泛数据预训练与特定任务微调的通用大语言模型 (Large Language Model, 简称 LLM) 成为 AI 强大泛化能力的代表。在该全新范式下，过去十几年中研究人员主要致力于开发新的训练方法和模型，如一些大名鼎鼎的工作：Transformer^[1]、AlexNet^[2]、GPT-3 等等，均已作为如今主流语言模型、音视频以及多模态模型的主流骨架 (backbone)。在这些研究者通过非凡的洞察力与工程能力的算法创新下，AI 模型从在国际象棋和围棋中击败世界冠军，到在 SAT 和律师考试中一骑绝尘，再到获得 IMO 和 IOI 金牌。DeepBlue、AlphaGo、GPT-4 和 o 系列这些里程碑的背后是 AI 方法的根本创新：搜索、深度强化学习 (Deep Reinforcement Learning, 简称 DRL)、扩展 (Scaling) 和推理。

但在这个范式的转化中，为了使模型及其算法能够运行并达到更高的性能，研究人员不得不对过去范式做出一些权衡。如符号主义范式中基于公理与逻辑推导知识表示方法所具有的强解释性，这一良好特征在基于统计与概率模型的数据驱动机器学习与 LLM 中由于建模方式的不同而被损害。且机器学习由于其外延数据（通过一组训练来尽可能学习或逼近到某个内涵式定义或者内涵式定义的片段）的不完备性以及数据质量难以保证，使得其结果往往只是一种经验科学，其在现实世界中的可靠性应用被极大限制。因此而引发的 LLM 幻觉问题，以及以 Manus 为代表的 AI Agent 等等 AI 发展所带来众多的可靠性问题极大限制了 AI 系统在更广泛的场景中被应用，人们对**安全可靠的人工智能系统 (Safe Guaranteed AI)** 的期待则是 AI 发展到当前阶段亟待完善的重要议题。

进入 AI 发展的下半场^[3], 尽管算法层面上的重大突破往往更加困难, 但令人兴奋的是**测试时间扩充 (Test-Time Scaling)**^[4]的趋势以及“**强化学习算法在 LLM 后训练中最终泛化并收敛**”这一发现引领着 AI 向着原来“基于广泛预训练与微调”的范式更深入的方向发展, 并进一步使强化学习被更多人认为是 AI 发展的终局。从 Instruct-GPT 中第一次引入基于人类反馈的强化学习 (RLHF) 所取得的成功, 到 DeepSeek-R1 中发现引入大规模强化学习后在长思维链推理中出现的“灵光一闪” (“Aha Moment”) 等等工作都在验证这一观点。

但这一观察并不总是生效, 其一大影响因素就是强化学习三个组件的设定: 算法、环境和先验。长期以来, 研究人员把更多的关注放在了强化学习算法上 (将在第二章详细介绍), 并将环境设置为固定。但是其中最重要的先验通常被人们忽略, 而算法的性能通常与开发和测试的环境高度相关。因此, 由于强化学习的敏感性, 我们可能需要先弄清楚我们真正想要解决的任务的环境, 然后再找到最适合它的算法或建模方式。

为此, 基于以上两个重要观察, 本文章首先希望探究语言模型的不同建模方式 (限定为基于 n 阶马尔科夫因果建模的自回归模型以及基于扩散过程的非自回归模型) 在强化学习采样效率上的差别, 并探索出一种高效采样的方式, 并探究在强化学习设定下不同任务所蕴含的先验在不同建模方式的偏好。同时, 文章中也给出了一种基于块级扩散模型的强化学习建模方式, 尝试缓解当下强化学习算法在扩散模型中因 KL 散度不稳定而无法有效训练的现状。同时该方式可以直接有效地应用到 LLM 的推理阶段 (测试时间扩充), 让 LLM 在有限响应时间内输出更可靠的结果。

1.2 研究意义

挑战 1: 理论上, 当前学界并未对掩码扩散模型 (Masked Diffusion Model, 简称 MDM) 模型的后训练方法有系统性的研究, 同时在实践中当下离散扩散模型很难有效控制并获得其相比自回归大模型应有的控制性的优势。

实践中, 在当前任务设定中希望通过强化学习对齐自然语言推理以及 Python、Dafny 代码的验证偏好, 并通过训练得到的价值函数 $V(s_t)$ 快速判定当前去噪的状态 \mathbf{x}_t 搜索方向是否正确, 进而判断按照当前策略 (也即采样高斯分布的参数)

继续去噪下去是否有前景。如果有，就继续去噪。否则，则终止去噪，并接收强化学习价值函数的反馈并寻找更优的搜索方向。

贡献 1: 本文将在下面中尝试引入一个新的建模方式：把离散 Token 的“去噪过程”建模一个多步马尔科夫决策过程。通过引入的奖励模型 $r(\mathbf{x}_0, c)$ 进而定义并对值函数 $V(s_t)$ 进行训练，并使用成熟的策略优化算法优化扩散模型。当前目标是在微调过 MDM 后最大化奖励模型的输出，即

$$J(\theta) = \mathbb{E}_{c \sim p(c), x_0 \sim p_\theta(x_0|c)} [r(x_0|c)].$$

据本文作者所知，这也是第一次尝试把掩码扩散模型使用 RL 后训练进行对齐。本文希望 RL 不只给离散扩散模型带来 Python 解释器，或 Dafny 编译器的验证信号，并基于此进一步研究强化学习究竟给生成式建模带来了什么。

挑战 2: 在代码生成的任务设定中，Python 以及 Dafny 的代码往往具有很强的结构性，采用当前扩散模型尽管可以优先生成低质量的代码框架，但模型并不知道每一个小框架的具体意义，进而没有办法充分释放离散扩散模型可以在少量去噪步骤后生成大致框架的效率。

贡献 2: 基于此，文章尝试采用分层强化学习 (Hierarchical Reinforcement Learning) 的想法，也即将一个复杂的规划问题分解成多个小的、简单的子问题，每个子问题都可以单独用多步马尔可夫决策过程来建模。本文将对 Python 以及 Dafny 代码依据程序单元 (Program Units) 划分成不同的部分，每一个程序单元会接收到各自的“任务描述”、“之前程序单元各自的描述”，和“当前程序单元类别 c_i ”，进行扩散模型建模。这个想法和离散扩散模型的变式：条件扩散模型 $p_\theta(\mathbf{x}_0|c)$ 不谋而合。当然也可以把条件变量 c 设置为 Python 或 Dafny 代码中不同的程序单元，他们各自有着自己的设计特点。因此，我们可以把整个 Dafny 或 Python 代码建模为条件概率分布如下式，更具体的内容将在流水线中介绍。

$$p_\theta(\mathbf{x}) = \sum_{c \in \mathcal{C}} p_\theta(\mathbf{x}_i|c_i).$$

观察到案例分析中 ChatGPT o4-mini-high 在得到 Dafny 的验证反馈后，往往会额外生成一些辅助函数对方法 (Method) 进行约束。故为进一步扩展并行性，每

一个模块也可以继续划分子模块，把原来做块级扩散的单位由大化小并行处理。

理想情况下，本文章希望利用扩散模型做代码生成任务，并尽可能利用掩码扩散模型双向建模的能力来快速得到代码的大致框架。之后利用 RL 的奖励模型快速评判代码框架是否合理，进而决定扩散模型是否要继续生成，或者改变代码框架生成策略。最初想法是基于微软设计的一个函数式形式化验证语言系统 Dafny，其验证证明由数学工具支持，该工具会基于数学整数和实数、位向量、序列、集合、多重集、无限序列和集合、归纳、共归完成计算证明。如果有足够的规格，则会自动履行验证义务，其一般的证明框架是霍尔逻辑的框架。利用 Dafny 语言的可验证性优势作为系统设计方式，这也是本文在成型的探索过程中从 Neuro-Symbolic Reasoning，到 Lean，再到 Dafny 的“探索”学习过程中**希望构建一个安全可靠的人工智能系统的初衷**一致。

但是在实践中发现形式语言（如 Dafny）的数据量太少且质量不佳，同时本文作者也没有充足的计算资源支持数据集的构建。故目前本实验优先使用阿里巴巴团队提出的 Qwen 2.5 7B-Instruct 模型^[5]以及香港大学团队提出的基于扩散掩码建模的 Dream 7B 模型^[6]做自然语言的文本生成，以及 Python 的代码生成任务，在这两个任务设定中探索不同语言模型方式的样本生成效率。同时也做出了一些关于 Dafny 的案例分析，希望能在未来继续研究啦。

1.3 论文结构

本文内容可以分为绪论、相关工作、研究方法、实验分析与结论、总结与讨论这 5 个章节。

第一章绪论介绍了本文的研究背景、研究意义和论文结构。在研究背景中讨论了人工智能的发展现状以及发展趋势，引出了本文两个技术要点：强化学习以及离散扩散模型，其中并未涉及具体的技术类别或细节；研究意义简述了建模方式（掩码扩散模型）中存在的不足以及任务（代码生成）中可以利用的任务特性，并简述本文工作解决该挑战的方法。论文结构中阐述了本文各章节的简要内容。

第二章相关工作，首先介绍了全新的语言模型建模-掩码扩散模型的两种建模方式：离散时间建模，以及连续时间建模，此后给出了两种模型训练优化方

法。此后详细介绍了基于强化学习微调的后训练方法，先给出了传统语言模型后训练的设定与其对应算法，随后详细给出现代大语言模型后训练的设定、三步方法论及其详细算法。

第三章研究方法，首先对两个 Dafny 代码生成进行了分析，展示了当前 SoTA 推理模型的，ChatGPT o4-mini-high 能力特性及其不足之处。随后尝试通过大于概率模型重新建模，以及在其基础上对强化学习设定进行适配和更改，进而给出基于块级离散扩散模型的分层强化学习建模。最后，综合以上建模给出完整的基于离散扩散模型后训练的流水线设计，并对实验中涉及的奖励模型建模，以及应用的强化学习算法进行了详细说明。并在最后给出了该建模方式是如何进行的推理优化（测试时间扩散）。

第四章实验分析与结论介绍了为验证本文猜想与建模方式所进行的实验，此后对实验结果进行了分析，并综合得出了三条结论。

第五章总结与展望总结了本文的工作内容，并反思了实验中还存在的不足之处，并对本文中涉及到的两个研究方向：掩码扩散模型与强化学习后训练的发展前景给出了自己的看法。

第二章 相关工作

2.1 掩码扩散模型：全新的语言模型建模

扩散架构自诞生起^{[7][8][9]}在众多领域和应用得到了长足发展，尤其在图像生成^{[10][11][12]}，语音生成^{[13][14]}，以及视频生成任务^{[15][16]}中。可以发现，目前扩散模型最成功的应用是在连续状态空间的生成。随着人们尝试把其往离散状态空间中扩展时，扩散模型在许多领域的应用也有大获成功，如文本声音转换任务^[17]和蛋白质设计^[18]等。

受如此多成功应用的启发，我们是否可以把扩散模型应用在文本生成任务中？扩散模型成功的重要因素其一是把自回归模型“一次到位”的思想转化为非自回归结合自回归的“多次到位，每次逐步击破”，这一天然优势能否在文本生成中继续展现；此外，扩散模型的机制让其可以并行预测，进而极大提高生成效率；同时，其对数据进行双向建模的特点可以让其全面考量数据特征，更好捕捉数据之间的双向依赖关系。而这些特点是从前到后生成的自回归所不具备的，使得“利用扩散模型进行文本生成”引起了学界的很大关注。

但是这一想法在建模和实践中遇到了很多难题。其一大难点是离散的文本 Token 如何进行加噪（前向过程）和去噪（反向过程）。一种方法是我们可以在词向量（Word Embedding）上加入高斯噪声，如 Diffusion-LM^[19]，和 DiffSeq^[20]等模型均采用了这类技术。此外，目前主流的方法则是不引入高斯噪声而使用其他类型的噪声，如通过随机删除 Token 的 DiffusER^[21]模型，以及 MASK-Predict^[22]中非常经典且有效的类“掩码”方式：通过引入特殊符号 [MASK]，并通过噪声超参数的调度把原始 Token 转化成该特殊符号来实现在原文本中加入噪声。最新的开源扩散掩码模型的工作 LLaDA 8B 发现了这种建模方式的可扩散性，并最终取得了和 LLaMA 3 8B Base 相当的结果。他们展示了扩散模型在文本生成中的潜在可扩展性和非自回归的对话能力，并在一定程度上证明了生成建模的原则（而非基于马尔科夫性的自回归公式）才是 LLM 能力的基础。

最近,MDM 领域的重大进展凸显了扩散在语言任务中不断增长的潜力。DiffuLLaMA^[23]和 LLaDA^[24]将扩散语言模型缩放到 7B 参数,而 Mercury Coder 作为一种商业实现,在代码生成方面表现出了卓越的推理效率。5 月 21 号,谷歌 Deepmind 团队发出了他们全新的文本扩散模型: Gemini-Diffusion。其性能方面超越了许多类似的自回归模型,并在平均推理速度达到了每秒 1479 Tokens,给出一个回答的平均用时开销仅为 0.84 秒。这种结合扩散语言建模固有的架构优势令人兴奋快速进展,使这些模型成为克服自回归方法基本局限性的关键解决方案。

在当前的扩散开源模型中效果和闭源模型还有一些差距,但所有的模型依然具有一些模型架构设计的固有问题。如扩散模型一般需要多次迭代去还原文本,且并没有引入 Transformer 架构中成熟的 KV Cache 技术,离散掩码模型的推理速度相比自回归模型甚至可能更慢(当然不排除这些闭源模型已经实现);同时其超参数的设置更麻烦,相比同参数量级的 Transformer,扩散模型存在更多需要手动设置的超参数,如 α_t ,反向过程时的高斯分布的方差等等。由此可见该领域方兴未艾,还有许多可以优化的空间。

通用的 MDM 模型的建模方式,大体可以分成离散时间建模以及连续时间建模两种,以下将分别简单介绍两种建模方式的前向过程和反向过程。

2.1.1 离散时间建模

首先先对单个 Token 的扩散过程进行建模,并做出一些假设。假设数据在有限离散空间中有 m 个可能的状态,其分别可以用下标 $0, 1, \dots, m-1$ 表示,其对应的独热编码 (One-Hot) 表示为 e_0, e_1, \dots, e_{m-1} 。在引入 [MASK] 这个特殊 Token 的设定下,需要额外对有限空间中的状态进行扩充,也即 [MASK] 对应的下标为 m ,其独热编码的表示为 e_m 。

前向过程: 加噪声

MDM 的前向过程可以看做一个离散随机变量 X 在不同时间状态的不同取值 x_t 所构成的一阶马尔科夫序列,我们不妨设离散时间建模的前向过程和反向过程的时间变化都在 $[0, 1]$ 。因此,每一个离散随机变量 x_t 由下标 t 指定,且 t 在

[0, 1] 之间，一开始干净的 Token 状态为 x_0 ，加噪到最后的的状态为 x_1 。这里本文把 [0, 1] 分割为 T 个间隔，为指定区间的左右分割，令 $s(i) = \frac{i-1}{T}, t(i) = \frac{i}{T}$ 。为了表示每个状态的转换， $s(i)$ 到 $t(i)$ 的状态转换可以用矩阵 Q 来指示， Q 的定义如下，

$$Q_i = (1 - \beta_i) \cdot I + \beta_i \cdot \mathbf{1} \cdot e_m^T \in \mathcal{R}^{(m+1) \times (m+1)}.$$

此外， $[Q_i]_{jk}$ 表示序列时间为 i 时， x_i 从状态 j 转变到状态 k 的转换，其可以表示为，

$$[Q_i]_{jk} = q(x_{t(i)} = k | x_{s(i)} = j) = (1 - \beta_i) \cdot \delta_{jk} + \beta_i \cdot \delta_{km}.$$

这也可以理解为 x_i 有 $1 - \beta_i$ 的概率保持状态不变（也即 $x_{t(i)} = x_{s(i)}$ ）， β_i 的概率转换到 [MASK]。在定义了状态转换矩阵后，前向过程对应的边缘分布也可以表示为，

$$q(x_{t(i)} | x_0) = \text{Cat}(x_{t(i)}; \hat{Q}_i^T x_0) = x_0^T \hat{Q}_i x_{t(i)}.$$

其中， $\text{Cat}(x; p)$ 表示一个类别分布， p 是一个向量分布，表示状态 x 转化为每一个类别的概率。其中 $\hat{Q}_i^T = \prod_{j=1}^i Q_j = \alpha_i I + (1 - \alpha_i) \mathbf{1} e_m^T$ ， $\alpha_i = \prod_{j=1}^i (1 - \beta_j)$ 。这里的 α_i 由 β_i 定义， β_i 随时间从 0 到 1 变化，值不断增加，表示每个状态的无序性度量（也即该步骤加入的噪声的总量）； α_i 随时间从 0 到 1 变化，值不断减少，表示每个状态的有序性度量。

请注意，前向过程中各种设定均为人工设定的超参数，其存在是为了给原始数据加入噪音并最终转化为高斯空间的样本。同时，前向过程中的信息也通常会被用来定义反向过程的建模，以下将详细说明。

反向过程: 去噪声

MDM 的反向过程可以理解为是前向过程的近似转向，该过程的状态转移函数 $p_\theta(x_s | x_t)$ 也即我们需要训练的模型。其可以利用自定义的前向过程来表示，也即

$$p_\theta(x_s | x_t) = q(x_s | x_t, \mu_\theta(x_t, t)).$$

其中 $\mu_\theta(x_t, t)$ 生成了一个 $m+1$ 维的类别分布，这也是本文需要通过神经网络 $f_\theta(x_t, t)$ 建模的反向过程模型。其把从更无序的状态 x_t 转换到 x_s ，进而不断去噪

并还原原始 Token。当然如果从采样的角度来看， $\mu_\theta(x_t, t)$ 本质上是在该步骤进行高维高斯分布采样的分布均值。二者之间的关系如下所示，

$$\mu_\theta(x_t, t) = \begin{cases} \text{softmax}(f_\theta(x_t, t)) & x_t = m, \\ x_t & x_t \neq m. \end{cases}$$

自此，离散时间的前向过程及反向过程的建模已经介绍完毕，以下将介绍该建模的训练方式。

训练: 极大似然估计

MDM 的建模损失函数和基于 n 阶马尔科夫的自回归基本一致，均采用极大似然技术进行建模。以下先写出对数极大似然建模的下界（也被称为证据下界，Evidence Lower Bound, 简称 ELBO），

$$\log p(x_0) \geq \mathbf{E}_{q(x_{t(1)}|x_0)} [\log p(x_0|x_{t(1)})] - KL(q(x_1|x_0)||p(x_1)) - \mathcal{L}_T.$$

其中，KL 散度项值为 0；如若 $T \rightarrow \infty$ ，上式的第一项中 $t(1) \rightarrow 0$ ，因此这项也可以被认为是 0。故上式中只剩下了 \mathcal{L}_T 项，也即证据下界为 $-\mathcal{L}_\infty$ 。

\mathcal{L}_T 的推导结果如下，

$$\mathcal{L}_T = \sum_{i=2}^T \mathbf{E}_{q(x_{t(i)}|x_0)} [KL(q(x_{s(i)}|x_{t(i)}, x_0)||p_\theta(x_{s(i)}|x_{t(i)}))].$$

在 Shi 等人^[25]的工作的推导中，其可以进一步简化为，

$$KL(q(x_s|x_t, x_0)||p_\theta(x_s|x_t)) = -\frac{\alpha_s - \alpha_t}{1 - \alpha_t} \delta_{t,m} \cdot x_0^T \cdot \log \mu_\theta(x_t, t).$$

因为 x_0 是一个独热编码，此推导式也可以被认为是预测的输出和真实 Token 之间的交叉熵损失。对该式取极限后即可得到 \mathcal{L}_∞ ，而 $-\mathcal{L}_\infty$ 也即最终需要最小化的损失函数优化式。

请注意，该优化式由于有积分形式存在仍难以优化。在实践中各个模型都对该式做出了一些训练开销和精度的权衡，此处不再详细说明。

2.1.2 连续时间建模

连续时间建模也即对 $[0,1]$ 时间内不设固定的离散状态点，而是可以任取其中两个时间点进行前向过程以及反向过程。该建模方式难度很大，此时并没有根据该建模方式训练出一些开箱即用 (Out-of-Box) 的模型。故以下只给出对应的建模方式以供参考，公式中对应的参数和离散时间建模一致。

前向过程被建模为，

$$q(x_t|x_s) = \text{Cat}(x_t; \hat{Q}(s,t)^T x_s), \quad \hat{Q}(s,t) = \hat{Q}(s)^{-1} \hat{Q}(t) = \frac{\alpha_t}{\alpha_s} I + (1 - \frac{\alpha_t}{\alpha_s}) \mathbf{1} e_m^T.$$

反向过程的建模和离散时间建模一致，

$$p_\theta(x_s|x_t) = q(x_s|x_t, \mu_\theta(x_t, t)).$$

训练目标和离散时间建模略有不同，连续时间建模时的噪声分配是依据信噪比 λ_t 定义的，其被定义为 $\alpha_t = \sigma(\lambda_t)$ ，其中 σ 是 Sigmoid 函数。此外，还额外定义了 Log-SNR (log 形式的信噪比) $\lambda_t = \log \frac{\alpha_t}{1-\alpha_t} = \log - \text{SNR}(t)$ 。此时的 \mathcal{L}_∞ 被化简后如下呈形，

$$\mathcal{L}_\infty = \int_{\lambda_{t(1)}}^{\lambda_1} \sigma(\lambda) \cdot \mathbf{E}_{\hat{q}(x_\lambda|x_0)} \left[\delta_{x_{\lambda,m}} \cdot x_0^T \cdot \log \hat{\mu}_\theta(x_\lambda, \lambda) \right] d\lambda.$$

其中只有 $t = \log - \text{SNR}^{-1}(\lambda)$ 时，噪声调度 α_t 才会对上式产生影响。因为只有 $t = \log - \text{SNR}^{-1}(\lambda)$ 时， $\hat{\mu}_\theta(x, \lambda) = \mu_\theta(x, t)$ ， $\hat{q}(x_\lambda|x_0) = q(x_t|x_0)$ 。

请注意，连续时间建模由于过于复杂进而有许多角度可以解释该运行过程。以上建模方式只是 Shi 等人^[25]工作在连续时间建模时的建模，还有许多其他的工作，如 Cambell 等人^[26]以及 Sun 等人^[27]从其他不同的角度对连续时间的离散扩散模型进行建模。

自此，MDM 通用的两种建模方式：离散时间建模和连续时间建模均已介绍完毕，文章中使用的离散时间建模的扩散模型特性将在下文中说明。下面将介绍文章中另外一个重要的技术：LLM 后训练中深度强化学习算法。

2.2 后训练：基于强化学习的微调

随着大语言模型 (LLM) 的快速发展，人们发现仅仅先通过广泛预训练获得世界知识，再通过有监督微调学会回答问题，这两个步骤构建的 LLM 往往不能解决所有问题，并带来了诸多挑战。如预训练中因其极大似然目标带来的级联错误 (Cascading Errors)；微调时带来的过拟合、高计算成本、泛化降低的风险；以及灾难性遗忘 (Catastrophic Forgetting)、推理时间权衡等等问题。此外，缩放技术的进步，包括低秩适应 (LoRA)、适配器和检索增强生成 (RAG)，提高了计算效率和事实准确性。这些策略与分布式训练框架相结合，促进了大规模部署，并进一步提高了 LLM 在不同应用程序中的可用性。为此，研究者们希望通过深度强化学习等方法利用动态反馈和优化顺序决策来增强适应性，通过这些有针对性的训练后干预（也即后训练），LLM 更好地与人类意图和道德要求保持一致，最终提高其在现实世界中的适用性。

传统基于马尔科夫决策过程的强化学习设定直接用于文本生成会有以下问题：高维输出（词汇表过大，RL 的规划和奖励分配复杂化）、非平稳目标（基于语言的 RL 的反馈是稀疏的、主观的和延迟的）、复杂的奖励结构（在文本生成时往往奖励设置需要平衡多个相互冲突的目标）。因此，针对 LLM 优化的 RL 需要专门的优化技术来处理以上三个重要挑战，并确保响应保持上下文相关性并与用户期望保持一致。本小节将总结通过深度强化学习方法来解决以上问题的方法，而不会过多赘述在传统强化学习定义下的算法设计。

2.2.1 传统语言模型后训练设定与算法

以下是最初在语言模型 (LM) 中的强化学习设定，

定义 2.2.1 (LM 中基于多步马尔科夫决策过程的强化学习设定). 由于 LM 自回归生成文本（每个标记预测都取决于先前生成的标记），这个过程可以建模为马尔可夫决策过程 (MDP) 中的顺序决策问题

$$\left\{ \begin{array}{ll} \text{State} : s_t & \text{到目前为止生成的 Token 序列 } m \\ \text{Action} : a_t & \text{下一个 Token,} \\ \text{Reward} : R(s_t, a_t) & \text{评估输出的质量.} \end{array} \right.$$

LLM 的策略 π_θ 经过优化，以实现预期回报最大化：

$$J(\pi_\theta) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t \cdot R(s_t, a_t) \right].$$

基于以上在强化学习设定，现代语言模型中观察到的思维链推理（也即一个强化学习 MDP 的一个轨迹）自然被定义为强化学习问题。从这个角度来看，每个中间推理步骤都被视为有助于最终答案的行动。策略梯度更新由下式给出：

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau} \left[\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(x_t | x_{t-1}) \cdot A(s_t, a_t) \right].$$

其中，优势函数 $A(s_t, a_t)$ 将信用 (Credit) 或奖励分配给各个步骤，确保通过即时奖励 (时序差分) 和延迟奖励 (蒙特卡洛算法) 来完善整个推理过程。这些公式 (包括逐步奖励分解) 对于提高 LLM 在复杂推理任务中的可解释性和性能至关重要。

早期应用到 LLM 的 RL Methods 基本都是从传统强化学习的设定中进行的迁移，并无过多基于文本特质进行的设计。如基于策略梯度的 REINFORCE 算法，以及希望利用状态值函数进行价值估计的 Actor-Critic 的 Advantage Actor-Critic^[28]，简称 A2C 算法。以及 A3C 算法 Asynchronous Advantage Actor-Critic，该类方法稳定了强化学习过程，提高了样本采样效率，并加速收敛，在复杂的决策任务重更加有效。

以上这些方法为强化学习应用到 LM 过程中凸显出的曝光偏差和高方差等挑战提供了初步解决方案。现代技术如使用 PPO 的大规模人类反馈强化学习 (Reinforcement Learning from Human Feedback, 简称 RLHF) 和先进的奖励模型，如群体相对策略优化，直接建立在这些思想之上。这些算法也会在后面详细说明。

2.2.2 大语言模型后训练设定、方法论与算法

在现代中，RL 的建模方法和之前的建模方式没有任何差别，也即 Definition 2.2.1 中定义。但从方法论的角度来看，将 RL 集成到 LLM 的推理中发生了很大变化，通常遵循三个核心步骤：首先是监督微调 (Supervised Fine-tuning, 简称 SFT)：从预

训练的语言模型开始，随后在高质量、人工制作示例的监督数据集上进行优化。此阶段可确保模型获得对格式和样式准则的基线合规性；其次是奖励模型（奖励模型）的训练：收集微调模型生成的输出并接受人工偏好标记。奖励模型为经过训练以复制这些基于标签的分数或排名，有效地学习将响应文本到标量值的连续奖励函数。在奖励建模和调整的过程中需要开发复杂的奖励函数（从人类偏好、对抗性反馈或自动化指标中汲取灵感）以指导模型实现连贯、安全且上下文适当的输出。这些奖励对于跨多步骤推理过程的有效学分分配至关重要；最后是策略优化：通过策略梯度算法（例如 PPO）优化主要语言模型，以最大限度地提高奖励模型的输出。通过迭代这个循环，LLM 学会在准确性、有用性和文体连贯性等关键维度上产生人类认为更可取的响应。

请注意，上述过程往往缺少评论家模型 (Critic Model) 的训练，但该组件对部分 LLM 的强化学习算法是必须的，如 PPO 算法。但是在部分算法（如 GRPO）中是被优化掉了。其不具备普适性，且在当下模型参数量以量级化增加的时代不具有普适性，故未在上面列出。同时，此时为表述以及分类清晰，未把使用强化学习算法进行微调的过程计入监督微调的过程。

以下将对这三个过程进行简单背景调研，将主要聚焦于当今具有零样本提示 (Zero-Shot) 能力的 LLM 上。

监督微调

第一步是**监督微调**，使用较为广泛的 SFT 技术有监督微调 (Instruct Finetuning), 对话式微调 (Dialogue (Multi-turn) Finetuning), 思维链推理微调 (CoT Reasoning Finetuning), 特定领域微调 (Domain-Specific (Specialized) Finetuning) 等；近两年涌现出的 LLM 往往采用了更新的监督微调技术，如蒸馏微调 (Distillation-Based Finetuning)：大型的“教师”模型有时被用来产生标记的数据或基本原理，较小的“学生”模型则用这些数据对其进行微调，比较典型的实例有思维链蒸馏 (CoT Distillation)。该技术在如今的推理语言模型 (Reasoning Language Model, 简称 RLM) 中被普遍使用；参数高效微调 (Parameter-efficient Finetuning, 简称 PEFT) 通过引入了一小组可训练参数 (LoRA) 或可学习的 Prompt (Prefix Tuning)，同时冻结了大部分模型权重，来完成在少量资源和时间内的模型训练。

奖励模型建模

第二步是**奖励模型建模**。设 \mathcal{X} 为用户可能查询 (Queries) 的空间, \mathcal{Y} 是所有的相应空间。对于每个查询 $x \in \mathcal{X}$, 收集一个或多个候选相应 $\{y_j\}_{j=1}^{m_x}$, 其中 m_x 是查询 x 的候选相应数, 这些响应可以是语言模型在不同的采样或 Prompt 的条件下生成的。此后人工注释者 (或其他反馈来源如编译器等) 为这些响应提供偏好判断, 往往是基于候选相应的排名。对于每个响应 (对), 本文使用 $\{r^i\}$ 来表示此类偏好数据, 其中 r^i 可能是标签、排名或指示偏好水平的索引。自此得到了整个数据集 D 由 N 个带注释的示例组成:

$$\mathbf{D} = \left\{ (x^i, \{y_j^i\}_{j=1}^{m_i}, \{r^i\}) \right\}_{i=1}^N.$$

基于此可以训练一个奖励模型 $R_\theta(x, y)$, 其将每个查询, 相应对 (x, y) 映射到标量分数, 我们的目标是让 R_θ 对齐收集的偏好, 通过映射到的标量分数反映其对齐的程度。

$$R_\theta = \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}.$$

请注意, 这里奖励模型学习的是相对偏好, 也就是优势函数 $Q(s_t, a_t) - V(s_t)$, 而非绝对分数。直接使用偏好数据训练奖励模型的经典工作为基于成对偏好的 Bradley-Terry 模型

$$P(y_j > y_k | x, \theta) = \frac{\exp(R_\theta(x, y_j))}{\exp(R_\theta(x, y_j)) + \exp(R_\theta(x, y_k))},$$

$$L_{BT}(\theta) = \sum_{(x, y_j > y_k) \in D} P(y_j > y_k | x, \theta).$$

和对响应进行全部排序的 Plackett-Luce 模型

$$P(y_{j_1}, \dots, y_{j_m} | x; \theta) = \prod_{l=1}^m \frac{\exp(R_\theta(x, y_{j_l}))}{\sum_{k=l}^m \exp(R_\theta(x, y_k))},$$

$$L_{PL}(\theta) = - \sum_{(x, \text{rank}) \in D} \sum_{l=1}^m \log \left(\frac{\exp(R_\theta(x, y_{j_l}))}{\sum_{k=l}^m \exp(R_\theta(x, y_k))} \right).$$

此后研究者在最新的强化学习算法中也提出了其他有关量化奖励的策略, 这一

部分将在下面对现代 LLM 强化学习算法中进行介绍。但是大体可以分为以下几个类别：显示或隐式建模（对奖励本身的类别进行分类），基于过程、结果或自适应的建模（对奖励函数进行分类的）。

对于奖励类别，显示奖励也即基于预定义的规则、启发式或人工注释来定义奖励函数，这种方法可以生成精确的奖励估计，但大规模的估计可能非常耗时或成本高昂；隐式奖励建模从模型观察到的行为或偏好信号中间接推断奖励，通常利用机器学习技术来揭示潜在的奖励结构。

对于奖励函数的类别，最常被使用的也即基于结果的奖励建模，用于衡量 LLM 输出的最终结果是否真实正确或解决了用户的查询。但这样的方法可能会产生信用分配问题；此外，基于过程的奖励建模在中间推理步骤分配反馈，奖励连贯、逻辑一致的思维链，通常被使用在涉及数学推导或代码调试的任务中；最近具有自适应奖励模型的迭代 RL 兴起，其旨在通过迭代优化奖励模型和策略模型来不断提高 LLM 的性能，这种方法解决了奖励黑客攻击 (Reward Hacking) 和奖励模型漂移的挑战。每次迭代后，奖励模型都会根据最新的模型行为和人工反馈进行更新。这也就意味着奖励模型不是静态的，而是随着时间的推移而演变，以更好地符合人类的偏好和任务要求。

值得注意的时，为文本生成任务定义奖励函数是一个病态问题，如何在某个特定场景中为 LLM 的输出分配一个合理的奖励一直是研究人员探索的方向。

策略优化

最后一步是**策略优化**。一旦有一个经过训练的奖励模型 $R_\theta(x, y)$ 来捕获人类或其他偏好，就可以把其集成到 RL 框架中来优化策略 π_ϕ 。这一步其实是用训练好的模型来替代并增强原生环境的奖励。

在经典的 RL 表示法中，状态 s 即下一个 Token 之前的部分（或全部）对话或部分生成过程，动作 a 即下一个 Token（或下一个 Chunk），策略函数 $\pi_\phi(a|s)$ 是下一个 Token 的条件分布。希望找到可以让奖励最大的策略 $\pi_\phi(\cdot|x)$ ，具体来说，设 x 为用户查询， $y \sim \pi_\phi(\cdot|x)$ 为生成的响应，目标是通过下式优化使响应获得最大的奖励

$$\max_{\phi} \mathbf{E}_{x \sim \mathcal{X}} \left[\mathbf{E}_{y \sim \pi_{\phi}(\cdot|x)} [R_{\theta}(x, y)] \right].$$

以下将简单介绍如今被 LLM 广泛使用的策略优化算法。

最经典的算法是 Trust Region Policy Optimization (TRPO), TRPO 是一种广泛使用的策略优化方法, 同时确保它们保持在受限制的信任区域内 (由 KL 散度衡量), 进而提高强化学习更新的稳定性。TRPO 通过以下优化问题对策略更新实施硬约束

$$\max_{\theta} \mathbf{E}_t \left[\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \cdot A_t \right].$$

同时有以下约束,

$$\mathbf{E}_t \left[D_{KL}(\pi_{\theta_{old}}(\cdot|s_t) || \pi_{\theta}(\cdot|s_t)) \right] \leq \delta.$$

由于解决这个有约束优化问题需要计算成本高昂的二阶优化技术, 如共轭梯度方法, 这使得 TRPO 对于 LLM 的优化效率较低。

PPO 在共享 TRPO 的优化目标外通过优化一个裁剪目标平衡探索和稳定性来更新策略。定义策略模型和参考模型的动作概率比为当前的奖励, 也即

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{ref}}(a_t|s_t)}.$$

裁剪后的 PPO 优化目标为

$$\mathcal{L}_{PPO}(\theta) = \mathbf{E}_t \left[\min(r_t(\theta) \cdot A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)) \cdot A_t \right].$$

其中 A_t 是优势函数的统计量 (通常采用时序差分残差), ϵ 是控制与先前策略的允许偏差的超参数。PPO 的剪辑目标限制了更新的策略分布与原始的策略分布之间的差异程度。这种调节避免了语言生成的灾难性变化, 并保持了训练的稳定性。

为了减轻 PPO 所带来的训练 4 个模型的需求, DeepSeek 团队引入了 Group Relative Policy Optimization (GRPO)。GRPO 根据同一问题的多个采样输出的平均奖励来估计基线。GRPO 的主要贡献是它消除了对评论家模型 (用来估计优势函数) 的需求, 并从一组抽样的 LLM 输出中估计基线奖励, 也即通过对每个问题 q , 抽取一组输出 $\{o_1, o_2, \dots, o_G\}$, 并通过奖励模型产生奖励 $\{r_1, r_2, \dots, r_G\}$, 最终每个响应获得奖励是标准化后的结果, 也即 $\bar{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$ 。此时对每个请求 q 的

响应 o_i 的优势定义为 $J(o_i, \theta, q)$ ，定义如下：

$$J(o_i, \theta, q) = \frac{1}{|o_i|} \sum_{i=1}^{|o_i|} (\min\{r_{ratio,i,t} \cdot \hat{A}_{i,t}, \text{clip}(r_{ratio,i,t}, 1 - \epsilon, 1 + \epsilon) \cdot \hat{A}_{i,t} - \beta D_{KL}[\pi_\theta || \pi_{ref}]\}).$$

其中概率比定义为

$$r_{ratio,i,t} = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_\theta^{old}(o_{i,t}|q, o_{i,<t})}.$$

最终，GRPO 的优化目标为，

$$J_{GRPO}(\theta) = \mathbf{E}_{q \sim P(Q)} \left[\frac{1}{G} \sum_{i=1}^G J(o_i, \theta, q) \right].$$

为进一步降低训练和显存需求，Direct Preference Optimization^[29] (DPO) 是最近提出的一种从人类偏好数据中训练 LLM 的方法，无需采用传统的 RL 循环。DPO 不是学习单独的奖励函数，然后运行策略梯度更新。而是直接将人类偏好信号集成到模型的训练目标中。因此，DPO 不是上述 PPO 目标，而是构建了一个目标，该目标直接推高了首选响应 ($y+$) 的概率，同时降低了不太首选响应 ($y-$) 的概率，所有这些都单个对数似然框架内完成。基于此优化目标，DPO 无需奖励模型以及评论家模型，只需保留参考模型和策略模型即可完成训练。DPO 的策略优化式如下所示，

$$\mathcal{L}^{DPO}(\theta) = \mathbf{E}_{((x,y+),y-)\sim D_{train}} \left[\sigma \left(\beta \log \frac{\pi_\theta(y+|x)}{\pi_{ref}(y+|x)} - \beta \log \frac{\pi_\theta(y-|x)}{\pi_{ref}(y-|x)} \right) \right].$$

2.2.3 强化学习采样优化

基于强化学习的 LLM 后训练是一个与时间相关的序列决策的问题，需要通过 Agent 不断和环境进行交互并获得环境的反馈才能对策略进行优化，而 Agent 生成 (状态, 动作) 序列的效率则是影响 Agent 获得环境反馈的准确度的重要因素。

为提高生成效率，研究者尝试在**推理阶段与 RL 的采样阶段**利用训练好的价值函数实时指导文本生成。价值函数引导解码 (Critic-Guided Decoding) 的迭代优化框架首先通过蒙特卡洛采样估计训练得到较准确的值函数，然后在生成

每个 Token 时参考值函数评估未来累积奖励，优先扩展高价值前缀。这一过程也可以理解为“提前拒绝”，即在生成序列尚未完成时，根据部分内容的奖励预测提前剪枝。这个思路在编程代码场景尤为重要：如果某一路径早期就被检测到无法通过测试或风格不佳，可以尽快放弃，避免浪费计算资源在长序列上。同时，这类方法无需反复用 RL 微调模型，而是在推理中直接控制输出，在摘要、对话等任务实现了与 RLHF 相近的对齐效果，同时计算成本大幅降低。

类似的两个工作，Zhang 等人^[30]和 Koh 等人^[31]也探索了基于值函数的分块搜索和树搜索，以限制模型输出不符合偏好的走向，从而提前拒绝低质量候选。

针对 LLM 高效采样的研究方兴未艾，很大程度上是由于模型是基于自回归建模的，线性生成的 Token 序列使得“提前拒绝”机制无法顺利运行。但值得注意的是，尽管在 GPU 集群急速增长的时代，采样数据的生成阶段往往依然是模型训练的瓶颈。这是因为在模型还未经过微调时，初始策略生成的大多数采样数据是不合理、无意义或重复的。如果不加过滤直接进入 RL 学习，容易让模型在不稳定或有害的样本（“负梯度”样本）上收敛得更慢或无法收敛。其次，在 RL 中使用自己生成的数据进行训练，容易出现分布偏移。如果此时的奖励模型本身不够精确，还可能出现奖励黑客攻击的现象，也即模型找到了骗过奖励模型的策略，导致生成内容得分高但质量差。

至此，RL 在当代 LLM 中后训练中常用的算法已介绍完毕。受限于篇幅，还有许多推理模型的细节无法详细表述。由于其未在本工作中设计，此处也不再详细说明。

第三章 研究方法

3.1 案例研究

以下通过两个 Dafny 代码的实例，详细阐释 Dafny 代码不同功能单元的特性，以及 ChatGPT o4-mini-high 在解决一个简单案例和一个复杂案例时的表现。希望通过这两个案例，说明在复杂案例时自回归模型因果建模的功能局限性，以及以功能单元进行扩散过程的必要性。

Leetcode 509: 斐波那契数列

首先给出一个简答的案例，该案例使用 ChatGPT o4-mini-high 强大的能力可以在两次迭代修改中生成语法正确且经过 Dafny 验证的代码。同时，在人工检查后发现也基本符合题目语义。该案例分析具体的分析在表 A-1。

Prompt:

```
# 题目描述：斐波那契数（通常用  $F(n)$  表示）形成的序列称为斐波那契数列。该数列由 0 和 1 开始，后面的每一项数字都是前面两项数字的和。也就是： $F(0) = 0$ ， $F(1) = 1$ ， $F(n) = F(n - 1) + F(n - 2)$ ，其中  $n > 1$ 。给定  $n$ ，请计算  $F(n)$ 。# 示例：## 示例 1：## 示例 2：## 示例 3：
# Python 代码：该问题对应的 Python 代码实现为 (略)。
# 要求：请根据 Dafny 的语法和题目语义，将该代码转化为可通过 Dafny 编译检查的代码
```

Leetcode 207: 课程表

这是一个更困难的案例，该案例需要使用基于递归的深度优先搜索完成。在该情况中，ChatGPT o4-mini-high 在几次迭代后不足以生成语法正确的 Dafny 代码。该案例的具体分析请见图 A-1。这里为了凸显任务的困难程度选择了动态

类型语言 Python。在进行代码转换时，转换出来的静态语言 Dafny 很有可能会受到 python 中变量类型不确定的影响（运行中也确实在 Dafny 中发现许多关于变量类型的错误）。实际做代码翻译时，可以采取不同的实验，从如 Java，C++，C 的静态语言，到如 Python 这类更复杂的动态语言。

Prompt:

题目描述：现在你总共有 numCourses 门课需要选，记为 0 到 numCourses - 1。给你一个数组 prerequisites，其中 prerequisites[i] = [ai, bi]，表示在选修课程 ai 前必须先选修 bi。例如，想要学习课程 0，你需要先完成课程 1，我们用一个匹配来表示：[0,1]。返回你为了学完所有课程所安排的学习顺序。可能会有多个正确的顺序，你只要返回任意一种就可以了。如果不可能完成所有课程，返回一个空数组。# 示例：## 示例 1: ## 示例 2: ## 示例 3:

Python 代码：该问题对应的 Python 代码实现为 (略)。

要求：请根据 Dafny 的语法和题目语义，将该代码转化为可通过 Dafny 编译检查的代码

3.2 基于块级离散扩散模型的分层强化学习建模

为探讨“扩散模型是否可以加速强化学习样本采样效率”这一假设，本文进行了没有显著结构化特征的自然语言任务推理以及具有显著结构化特征的 Python 代码生成任务的探究，以下将对实验研究方法做出详细阐释，实验结果将在下一章展示。

以下以扩散模型的单次分发作为问题设置：给定自然语言描述，把该问题划分成不同功能单元 (c_i) 的自然语言表述，其后直接生成关于该程序单元的代码。后续其他应用场景中可以进行多次分发，以进一步提升并行率，如在基于 Lean 做定理证明这类难度更大的场景中，DeepSeek-Prover-V2^[32]借鉴了分层强化学习以及课程学习的方法把复杂的目标划分为若干子问题逐个击破。本节只对该问题进行建模，对应的 RL 优化算法将在下一节中详细说明。

3.2.1 分层扩散模型概率建模

引入了条件变量 C , 定义扩散模型接受到不同类型（功能）的信息的条件变量。其中 c_0 为原始问题的自然语言表述（也同时意味着要生成对程序模块功能的自然语言表述）， $c_i \in C, \forall i \in [1, n-1]$ 指代响应中不同的功能单元

$$C = \{c^0, (c^1, \dots, c^{n-1})\}.$$

扩散模型的输入定义为 \mathcal{Y} , y_0 是原始问题的自然语言表述， $y_i \in \mathcal{Y}, \forall i \in [1, n-1]$ 是对每一个子功能模块的自然语言描述。

$$\mathcal{Y} = \{y^0, (y^1, \dots, y^{n-1})\}.$$

扩散模型的输出定义为 \mathcal{X} , $x_i \in \mathcal{X}, \forall i \in [1, n-1]$ 是每一个功能单元的相应。

$$\mathcal{X} = \{x^1, \dots, x^{n-1}\}.$$

综上，可以把扩散模型建模为一个随机过程，

$$(c^0, y^0) \longrightarrow (c^j, y^j) \longrightarrow x^j \quad \forall j \in [1, n-1].$$

在任务中对接受一个特定类别 c_j 的自然语言指令 y_j ，生成特定子模块的自然语言描述 y_j ，并最终生成每个功能单元的响应 x_j ，具体建模如下式，

$$\text{Phase 1, } p_{\theta}(y^j | c^0, y^0), \forall j \in [1, n-1], \quad (3-1)$$

$$\text{Phase 2, } p_{\theta}(x^j | c^j, y^j), \forall j \in [1, n-1]. \quad (3-2)$$

该建模方式可以理解为分层强化学习以及课程学习的建模方式，该方法在最初强化学习设定中提出是为了解决奖励信号稀疏问题，此时引入该概念是为了适配块级扩散的建模过程，并基于此并行推理加速并及时获得评论家模型的反馈判断生成框架是否合适；同时，如果后续为了统一建模，上式中的 Phase 1 和 Phase 2 也可以共同使用一个后验进行建模。以上内容将在下一节流水线中详

细介绍。

为了更清楚地说明建模时的并行化，下面给出基于 Dafny 第二个案例研究的图示(图 3-1)。Depth=0 时，从输入的自然语言描述生成每个子模块的自然语言描述 (Phase 1)，Depth=1 时，从子模块的自然语言描述生成对应的 Dafny code (Phase 2)。以上建模只进行了两阶段，实际上在复杂应用中也可以使并行度更高(如使用 Lean 来做数学定理证明，下图中子模块可以划分的更细)，如函数功能的实现需要更多辅助函数函数来完成父函数的功能。

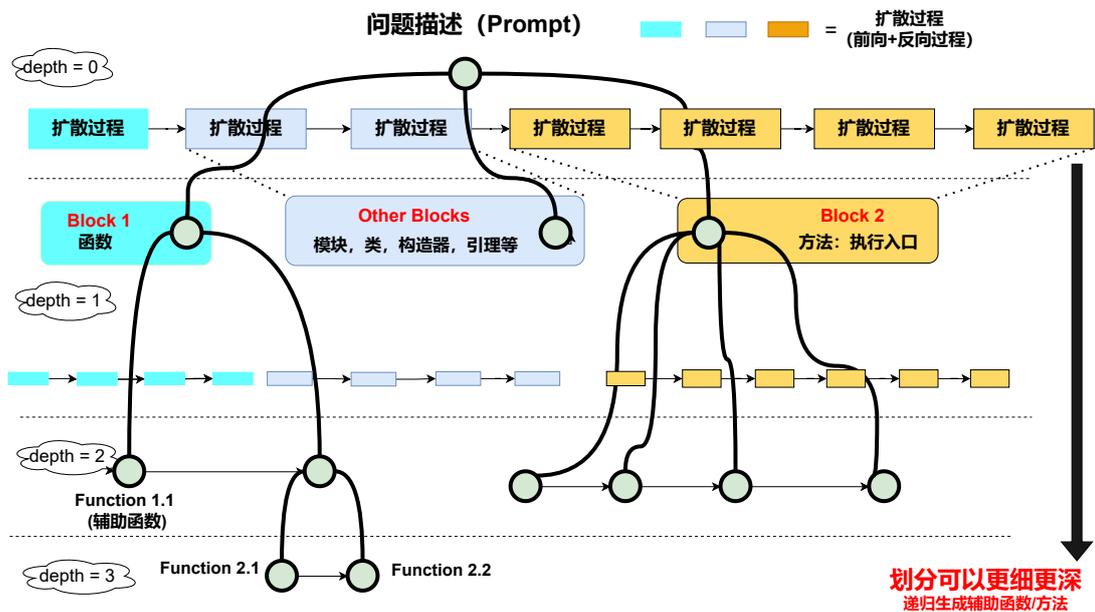


图 3-1 案例分析 2 中具体示例：并行性展示

3.2.2 分层离散扩散模型强化学习建模

原始建模

由于当前 MDM 的 SoTA 模型并没有采用块级 (Block-Level) 建模，而是把整个输出文本看做一个上下文，整体来做前向过程以及反向过程。为了展示和自回归模型的 RL 采样效率之间的差别，故在本设置下 MDM 的 RL 建模采用和自回归相同的建模方式，也即 Definition 2.2.1 中定义。本文章中的实验均采用这种建模方式，有关 RL 算法的具体设计将在下一节流水线设计中详细说明。

请注意，在本建模方式下 (未采用 3.2.1 节中的建模方式)，无法充分释放扩

散模型并行化的能力，依然是对固定上下文长度做前向以及反向过程；同时其无法根据扩散过程中的特定状态 ($x_{t(i)}$) 设计 RL 算法，只能在扩散模型做完固定时间步的反向过程获得奖励并进行强化学习微调。为此，本文尝试提出一种块级建模 (Block-Level Modeling)。

块级建模

对于以块级方式对 MDM 进行建模的模型，如块级扩散建模 (Block Diffusion^[33]), 其建模方式天然和分层扩散模型相匹配。其定义了离散随机变量块之间的自回归概率分布，给定先前的块条件概率由 MDM 的前向、反向过程指定，也即在块内做扩散过程。故该过程块间为自回归，块内为扩散过程的非自回归，该建模方式也称为半自回归。以下简单介绍块级扩散的建模方式，并给出该方法对应的 RL 建模。

本文将一个序列 x 中的 Token 分为 B 个区块，长度为 L' 且 $B = \frac{L}{L'}$ ，这里不妨假设 B 是个整数。为简单起见，本文将每个块表示为 x^b ，也即 x 从位置 $(b-1)L'$ 到 bL' 的区间取值，其中 $b \in \{1, \dots, B\}$ 。基于此，可以对块级进行似然分解，也即

$$\log p_{\theta}(x) = \sum_{b=1}^B \log p_{\theta}(x^b | x^{<b}).$$

每一个块都可以视作在做扩散过程建模，下面是在块内定义的反向传播过程，其接受了之前块的信息（原文章中是以 KV Cache 并辅助以专门设计的 Attention Mask 来传递过去的信息并加速训练过程），也即

$$p_{\theta}(x_s^b | x_t^b, x^{<b}) = \sum_{x^b} q(x_s^b | x_t^b, x^b) \cdot p_{\theta}(x^b | x_t^b, x^{<b}).$$

该建模方式的训练和众多离散扩散模型的训练一致：通过最小化 NELBO (Negative Lower Bound) 来拿到负似然的上限，进而优化原模型

$$-\log p_{\theta}(x) \leq \mathcal{L}_{BD}(x; \theta) = \sum_{b=1}^B \mathcal{L}(x^b, x^{<b}; \theta).$$

其中 $\mathcal{L}(x; \theta)$ 通过 Luo 等人^[34]，以及 Arriola 等人^[33]中复杂的推理过程可以化简

为下式，该式就可以使用自回归模型的训练框架定义损失函数并训练。

$$\mathcal{L}(x; \theta) = \mathbf{E}_q \left[-\log p_\theta(x | x_{t(1)}) + \sum_{j=1}^T D_{KL}(q(x_{s(j)} | x_{t(j)}, x) \| p_\theta(x_{s(j)} | x_{t(j)})) \right. \\ \left. + D_{KL}(q(x_{t(T)}) \| p_\theta(x_{t(T)})) \right].$$

基于此，可以把扩散模型建模为多步马尔科夫决策过程在异策略强化学习上的采样过程，以下以 Phase 2 为样本进行说明。

$$p_\theta(\mathbf{x}^j | c^j, \mathbf{y}^j), \forall j \in [1, n-1].$$

而这样建模也就是条件扩散模型，其理论和优化策略已经得到理论证明和实验验证。注意，由于 RL 是对所有的子模块进行建模，故以下定义时候均没有上标 j 。

类似 OpenAI 在图像扩散任务中强化学习建模^[35]，本文把文本生成的块内的扩散过程转化为多步马尔科夫过程，并可以被形式化为一个五元组，

$$\mathcal{M} = (S, A, \rho_0, P, R).$$

其中 S 是状态空间， A 是动作空间， ρ_0 是初始状态分布， P 是状态转移矩阵， R 则是奖励函数。强化学习的优化目标便是最大化策略的累计收益：

$$\mathcal{J}_{RL}(\pi) = \mathbb{E}_{\pi \sim p(\tau|\pi)} \left[\sum_{t=0}^T R(s_t, a_t) \right].$$

这里采用的是基于离散时间步的扩散模型，也即前向过程和反向过程都在时间 $[0, 1]$ 内完成。在 RL 中也采用相同的表示方法。下面将对每一个部分进行说明，这里的设计是为了和 PPO 算法优化式保持一致，可能还存在其他的 RL 建模方法。

状态： 本文定义为一个元组，包含引入的条件变量，去噪时间步，和当前时间步的去噪结果。 $\mathbf{s}_t \doteq ((c, \mathbf{y}), t, \mathbf{x}_t)$

策略： 在给定当前状态下，下一步的去噪结果 $\pi(\mathbf{a}_t | \mathbf{s}_t) \doteq p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, (c, \mathbf{y}))$

动作：下一步的去噪结果 $\mathbf{a}_t \doteq \mathbf{x}_{t-1}$

状态转移：由于在去噪中，只要动作做出之后，状态的转移就是确定的了 (经典的 RL 设置中，状态转移为 $P_{s_t}^{\mathbf{a}_t}$)，因此这里采用狄拉克 δ 函数来表示状态转移概率 $P(s_{t+1}|s_t, \mathbf{a}_t) \doteq ((\delta_c, \delta_y), \delta_{t-1}, \delta_{\mathbf{x}_{t-1}})$ 。其下标位置的概率为正无穷，其他位置为 0，同时期概率密度积分为 1。图 3-2 是该分布的一个示意图，直线上箭头的高度一般用于指定 δ 函数前任何乘法常数的值，亦即等于函数下方的面积。

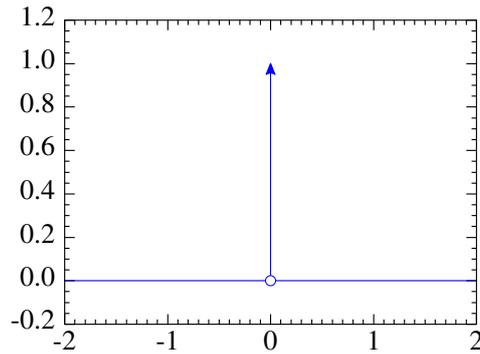


图 3-2 狄拉克函数概率分布

初始状态分布：条件变量 \mathbf{c} 服从其先验分布，时间步 T 是确定的，最后 x_T 则服从标准高斯噪声。 $\rho_0(s_0) \doteq ((p(\mathbf{c}), p(\mathbf{y})), \delta_T, \mathcal{N}(0, \mathbf{I}))$

奖励模型：只有最终的去噪结果会依据上文提到的奖励模型获得一个分数，(此时是由 Dafny 编译器提供的反馈)，而去噪过程中的奖励值都定义为 0。

$$R(s_t, \mathbf{a}_t) \doteq \begin{cases} r(\mathbf{x}_0, (\mathbf{c}, \mathbf{y})) & \text{若 } t = 0, \\ 0 & \text{否则.} \end{cases}$$

注意到上式目前并没有考虑块级扩散中所引入的过去上下文信息 $\mathbf{x}^{<b}$ 或者引入 KV cache 后的 $K^{1:b-1}, V^{1:b-1}$ 。上面对 RL 的建模后续仍要加上上下文信息，同时保证 MDP 的马尔科夫性，这二者如何不矛盾地完成统一建模，后续也还需考虑。同时如何在上面 RL 环境中简化 (\mathbf{c}, \mathbf{y}) 的二元组，并把分层建模统一成一个框架中也需要再作考虑啦。

3.3 流水线设计

如下图图 3-3 为一次分发的流水线 (Pipeline)，该流水线的设计希望首先通过分层推理提高并行性，层间进行任务描述和分发，层内执行块级扩散。其次把

块级扩散模型建模为多步马尔科夫决策过程，训练一个价值函数 $V(s_t)$ 用于推理时的效果判断，同时获得 Dafny 的反馈使用 PPO 对块级扩散模型进行强化学习微调。具体细节将分为奖励函数建模、强化学习微调与推理优化：测试时间扩散三个方面详细阐述。

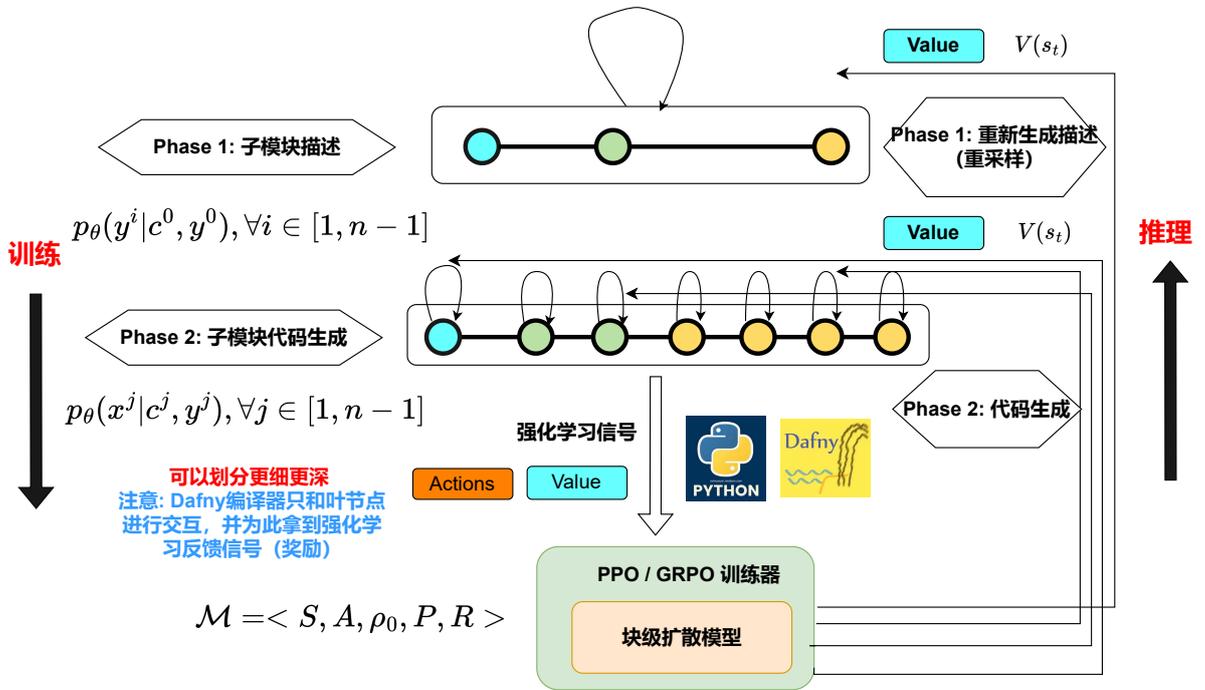


图 3-3 流水线：训练及推理

3.3.1 奖励模型建模

由于数据质量以及计算资源受限，本实验中未加入冷启动的 SFT 过程，而是直接基于基座模型进行 RL 训练。故可能会产生由于策略模型的不恰当的初始化而导致的 RL 训练不稳定的问题，这一部分将在下一章实验中详细阐释。此外，原始建模和块级建模的奖励建模并无差别，以下将统一进行介绍。

在医学问答思维链的自然语言推理任务中，本文把奖励函数分成了以下几个部分：

首先是**准确性**。本文使用模型指导的方式使用基于 Sentence-Transformers 的 Stsb-Roberta-Base 模型计算真实值 (Ground Truth) 和 LLM 响应之间的语义相似度，在进行实验调整后，对原始在 $[0, 1]$ 相似度区间进行缩放，调整到取值到 $[0, 2]$ 之间。此指标获得的奖励记为 $P_{i,t}(y_i, \hat{y}_i)$ ，有 $P_{i,t}(y_i, \hat{y}_i) \in [0, 2]$

其次是**推理深度**。本文作者观察到模型进行 CoT 时，通常会输出两次换行符进行段落更换，这表明上下两段的语义或功能的分割。在一次响应中，出现一次换段行为将给予 0.1 的奖励。但 Sui 等人^[36]提出了模型的“过度思考”(Overthinking)可能会影响下游任务的表现，这需要进一步的消融实验来探索奖励分配的合理性。此指标获得的奖励记为 $D_{i,t}(y_i)$ ，该指标无取值明显上限，但在本任务限制 Token 输出数量的情况下， $D_{i,t}(y_i)$ 一般的取值为 0, 0.1, 0.2，也即只在一次响应中进行一次到两次的段落更换，因此也不会出现过度思考的情况。

最后是**奖励格式**。由于希望模型有显示的思考过程，在 Prompt 中加入了希望模型输出基于 XML 格式的相应，也即引入了四个特殊 Token 来分割模型的响应并给予奖励：`<reasoning>`，`</reasoning>`，`<answer>`，`</answer>`，只要模型的输出中出现任何一种特殊符号，模型就会得到 0.15 的奖励。`<reasoning>` 和 `</reasoning>` 中表示模型在进行思维链推理，希望模型可以不断反思、精简、提出多种推理路径。`<answer>` 和 `</answer>` 之间也即模型最终的答案。此指标获得的奖励记为 $F_{i,t}(y_i)$ ，有 $F_{i,t}(y_i) \in \{0, 0.15, 0.3, 0.45, 0.6\}$

综上，在医学问答思维链的自然语言推理任务中，模型获得奖励如下，

$$\hat{A}_{i,t} = R_{\theta}^{i,t}(x_i, y_i, \hat{y}_i) = P_{i,t}(y_i, \hat{y}_i) + D_{i,t}(y_i) + F_{i,t}(y_i) \in [0, 2.8].$$

在**代码生成任务**中，奖励函数的设计会更加复杂。因为虽然代码必须具有人类可读性，但生成的代码必须保持语法和功能 correctness，即能够通过编译和单元测试 (Unit Test)。同时只依靠编译器或单元测试会造成奖励信号的极度稀疏性（也即多数情况下生成的代码都无法通过测试），尤其是扩散模型只在少量去噪过程中生成出的代码框架。

本实验中优先考虑 Python 代码生成，但类似思路可以适用于大部分高级编程语言。由于 Shojaee 等人^[37]在代码生成任务中进行了严格的奖励函数消融实验，验证了各类奖励设置的合理性和有效性，故本实验中主要参考其奖励函数设置，并对基于 GRPO 的强化学习微调过程做出适配。当前的奖励设置主要由以下两部分组成：

首先是**编译器反馈**。本文在提示词中提示 LLM 根据题目语义信息设置单元测试（如代码中设置 `assert` 语句）。如果当前源数据中包含单元测试，则奖励由

生成代码的功能正确性决定，否则则由生成代码的语法正确性确定。本文作者希望通过这项奖励让策略模型在语法/功能正确性方面生成更高质量的代码。

$$R_c = \begin{cases} R_{c_{functional}} = \begin{cases} 1 & \text{通过全部单元测试,} \\ -0.3 & \text{有没通过的单元测试,} \\ -0.6 & \text{运行时错误,} \\ -1 & \text{编译错误.} \end{cases} \\ R_{c_{syntactic}} = \begin{cases} 1 & \text{通过编译,} \\ -1 & \text{未通过编译.} \end{cases} \end{cases}$$

其次是**句法(结构)匹配得分**。在原论文中采用了抽象语法树 (Abstract Syntax Tree^[38]) 来查找代码的 AST 表示形式，因为基于 AST 的分析可以通过识别和比较代码示例的结构（而不仅仅是基于单个 Token 的判断）来提供更多信息。例如，缺少运算符或错误的类型转换（通常在 Token-Level 无法检测到的问题）将导致不同的 AST 子树。奖励计算通过分别比较从 LLM 响应和真实的代码生成的 AST 中提取子树，并计算句法匹配分数作为匹配 AST 子树的百分比，计算方法如下式，

$$R_a(\hat{y}, y) = \frac{\text{Count}(AST_{\hat{y}} \cap AST_y)}{\text{Count}(AST_y)} \in [0, 1].$$

其中分子是 LLM 生成的代码中抽象出的 AST 和 Ground Truth 抽象出的 AST 间匹配的 AST 子树的数量，分母则是 LLM 生成代码对应的 AST 子树的总数。

在 GRPO 算法中，本文做出了以下变化。首先是由于原论文的实验都是基于小参数量的语言模型（2023 年），且 Qwen 和 Dream 模型的输出中本文作者发现代码生成的**语义信息**往往都可以达到比较高的水平。原论文中还基于数据流图 (DFG) 来基于 \hat{y} 和 y 之间的语义匹配程度来给予一定的奖励，本实验中考虑到计算资源、计算效率以及大模型零样本提示 (Zero-Shot) 优异的表现等因素未纳入考量；其次原论文中还对策略模型以及参考模型之间的差异 (KL 散度) 作为奖励函数的考量因素。由于 GRPO 中引入了基于裁剪的优势函数，且裁剪的超参数 ϵ 设置为比较小 (0.1)，故此时也未把 KL 散度列入奖励函数的考量。

综上，在 python 代码生成任务中，模型获得奖励如下，

$$\hat{A}_{i,t} = R_{\theta}^{i,t}(x_i, y_i, \hat{y}_{i,t}) = R_c^{i,t}(y_i, \hat{y}_{i,t}) + R_a^{i,t}(y_i, \hat{y}_{i,t}) \in [-1, 2].$$

值得注意的是，本实验由于训练计算资源不足且一次训练的开销过大，未对奖励函数进行消融实验。可能会因为奖励函数设置不合理或冗余出现奖励收敛较慢或不收敛的情况，并出现的可能会出现奖励黑客攻击的问题。同时可能需要针对扩散模型做出一些更特殊的奖励函数设置。这些不足之处将在下一章节以及讨论中详细说明。

3.3.2 强化学习微调

原始建模在资源受限的情况下，本文优先考虑了使用消除值函数的 GRPO^[39]来进行强化学习微调。GRPO 只需要一个参考模型，一个策略模型，甚至在人工定义奖励函数的情况下可以不显示训练一个奖励函数，这大大减少了内存使用量并稳定了策略学习。该方法还与奖励模型的训练方式非常一致，即通过比较不同的 LLM 生成的输出，而不是预测绝对值。

块级建模基于块级强化学习设定和上面给出的概率建模，本文使用 PPO 对块级扩散模型进行优化，此时采用优化裁剪策略

$$\nabla_{\theta} \mathcal{J} = \mathbb{E} \left[\text{clip} \left(\frac{p_{\theta}(x_s^{j,b} | x_t^{j,b}, x_0^{j,b}, c^j, y^j)}{p_{\theta_{\text{ref}}}(x_s^{j,b} | x_t^{j,b}, x_0^{j,b}, c^j, y^j)}, 1 - \epsilon, 1 + \epsilon \right) \cdot V(s_t^{j,b}) \cdot \nabla_{\theta} \log p_{\theta}(x_s^{j,b} | x_t^{j,b}, x_0^{j,b}, c^j, y^j) \right].$$

在 PPO 作为强化学习微调算法时，可以拿到一个值函数 $V(s_t)$ ，在扩散的反向过程的过程中某个 step 即可判断该生成的方向是否有前景。

3.3.3 推理优化：测试时间扩充

原始建模对 MDM 的推理不做特别限定，也即让在超参数设定的扩散时间步全部完成作为当前 MDM 的相应结果。

块级建模的推理如图 3-3 中，在推理进行一定数量的去噪后，可以拿到对当前状态的状态值函数 $V(s_t)$ 。此后可以设定一定的阈值，超过这个阈值的去噪结

果（以及未来的可能结果）不会被采纳，需要重新采样。

但这里由于涉及到多个层级之间的重新生成，我们应该如何设计这个阈值，使得其值特别低的时候认为当前子模块生成的自然语言表述是不准确的，需要从子模块的任务描述进行采样；不大不小的时候，只需要根据当前自然语言表述重新进行 Dafny 代码的重采样；值比较大的时候认为当前策略正确，可以继续去噪。

第四章 实验分析 & 结论

本文在不同的任务设定上基于上述详细阐释的设定尝试验证文中提到的想法与假设。由于计算资源以及选择性很少的扩散掩码模型，本文对比了相同数量级的自回归模型 (Qwen-2.5-7B-Instruct) 和基于扩散掩码建模的非自回归 SoTA, Dream-7B, 并尝试在以下三个任务中进行了实验。

首先是自然语言推理，尤其是在医学任务中的思维链推理，其次是 Python 的代码补全与代码填空等任务。两任务的数据集格式请详见附录 B。

最后本文很希望尝试对形式语言的补全任务上的测试，如基于依赖类型论进行数学定理证明和验证的 Lean 语言，或是基于 SMT 问题进行代码验证的 Dafny 语言，这也是本文作者最初决定探索这个方向的动机：Lean 和 Dafny 中具有更加广阔的假设空间，现代 RL 算法生成采样数据时往往效率很低，我们是否可以通过扩散模型在更少的生成步中根据生成的“看起来还不错的答案”进行强化学习微调？在这个过程中我们是否可以观察到更快的奖励和损失收敛？在少量扩散时间步时进行强化学习微调后的模型扩大它的生成步后，是否可以一样获得更少的损失和更准确的奖励？

据本文作者了解，基于 Lean 做数学定理证明的一些工作 (Deepseek-prover^[32], BFS-Prover^[40]) 通过拒绝采样和多步筛选生成了 RL 数据，但并未公开；Dafny 语言只有很少数的数据集和少量的基准集，如果是基于更简单的 Leetcode 题目做 Dafny 验证的任务是没有对应数据集的。由于该任务在有限的时间和计算资源中实现比较困难，且 RL 的奖励函数也较难设计，故本文章中并未纳入实验，希望在研究生阶段可以系统性探索。

本实验使用 Pytorch 框架实现，并在推理和训练时使用了 Accelerate 进行分布式配置并加速，以适应多卡的训练设置。训练信息如 GRPO Loss 和强化学习中的平均奖赏从 Wandb (Weights & Bias) 中获取。该项目的代码将在 Github 公布: <https://github.com/JerrySiRi/Hierarchy-RL-enhanced-MDM>。

4.1 多学科问答基准测试

由于计算资源不足，本文实验无法全参数训练 7B 的模型（尽管做推理时全参数模型可以被加载，但在训练时所产生的激活值，优化器存入的一阶和二阶动量值会极大消耗显存），故此时选择引入 LoRA 进行参数高效微调。后续也尝试使用 DeepSpeed 的 Zero Stage 2，把训练过程中生成的激活值以及优化器中间变量转移到 CPU 上，但因磁盘空间（分配的资源只有 50GB，且无权访问云盘存储）仍需存储模型参数而导致不足未继续尝试。在 LoRA 中，设置 $r = 48, \alpha = 128$ ，让模型注意力层中的 Q,V 矩阵，以及输出层均注入适配器 (Adapter) 进行 LoRA 微调，最终可训练参数约占模型总参数量的 42%。同时，设置 lora dropout 率为 0.05，未防止模型载入时使用的 bfloat16 在 LoRA 微调时出现权重溢出，因此把新注入的适配器中的参数改为 float32。

首先是两个模型在平均奖励上的对比实验。此时 Dream 模型的扩散步为 64，也即 Dream 模型在 64 步去噪后得到最终结果。对 64 步反向过程的结果进行输出，发现此时的文本质量一般。正是由于此时输出质量一般，本文更期望可以在有限的反向过程中可以通过奖励或值函数拿到 RL 的信号改进扩散模型。为此，对 Dream 模型以及 Qwen 模型进行 GRPO 训练，奖励模型由前所述，在当前 VRAM 可支持的情况下，Batch Size 为 2，对每一个样本生成的 6 个采样数据计算当前组内的平均奖励，在更新参考模型前进行 500 轮训练后更新对其参考模型和策略模型。在计算 GRPO 的 Loss 值的时候，令 KL 散度的惩罚系数为 1，裁剪系数为 0.1。在进行策略模型更新时，学习率设成 $5e^{-6}$ 。训练数据来自 Chen 等人^[41]在 2024 年的工作，提出了一个从 GPT-4o 蒸馏出的医学推理数据。训练绘制的平均奖励如下图所示，

由图可知，在设定最大 Token 生成数为 500 时的 Dream 模型在显存完全消耗完前的训练过程中获得的奖励无法使其收敛，但 Qwen 模型在只输出 128 个新的 Token（可能被截断）时却观察到了奖励值不断增高的轨迹。为此，本文分析了两模型在 GRPO 训练中产生的采样数据。发现 Qwen 模型基本能很好地利用 128 个 Token 的容量进行充分推理，并很好在训练的过程中不断学会保留完整的特殊字符对。但是 Dream 模型无法很好利用提供的 500 个上下文空间进行推理，由于反向过程的时间步过少，多数潜在的 Token 在去噪过程中都只能被采样出

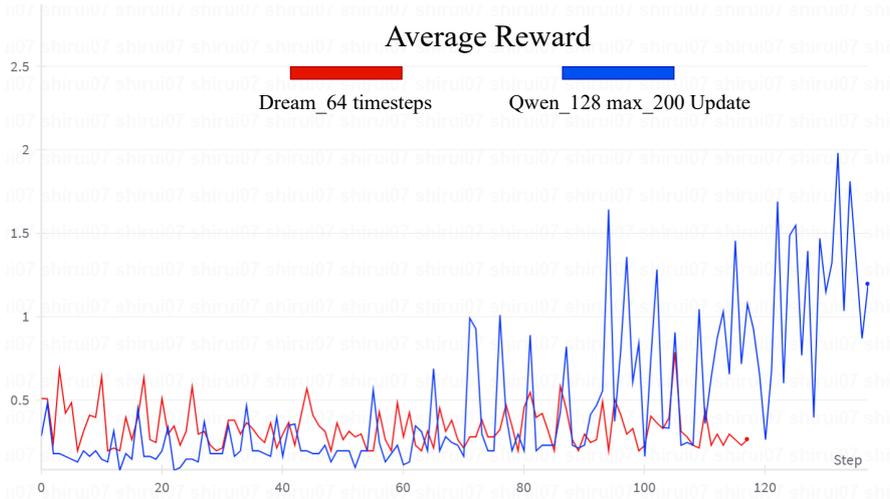


图 4-1 GRPO 训练中平均奖励的变化曲线

[MASK], 进而只能生成少量的论证而无深度推理。同时其输出质量也不高, 多数输出都在重复某些特定的 Token, 抑或是其他的一些标点符号。图 4-2 是两个模型在训练过程中的 GRPO 损失变化, 由组合替代损失 (Surrogate Loss) 和 KL 散度组成, 发现 Dream 模型的 GRPO 损失极度不稳定。在检查 GRPO 损失各组成部分的数值情况后, 发现是 KL 散度的极度不稳定影响了整体的 GRPO 损失, 也即 Dream 模型的输出不具有稳定性, 这也进一步导致策略模型无法很好根据奖励信息进行更新。具体输出示例与分析请详见附录附录 C。

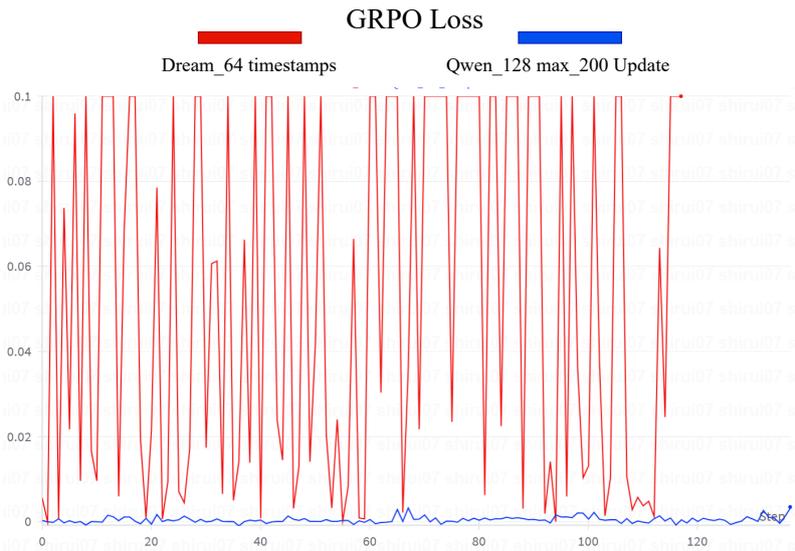


图 4-2 GRPO 训练中 Loss 的变化曲线

图 4-3 是 Qwen 模型在最大生成 Token 数为 256 时的奖励收敛情况, 相比最

大生成 Token 数为 128 时，更长的上下文输出对应的 GRPO 损失更小，策略模型更新更加稳定并更快收敛。

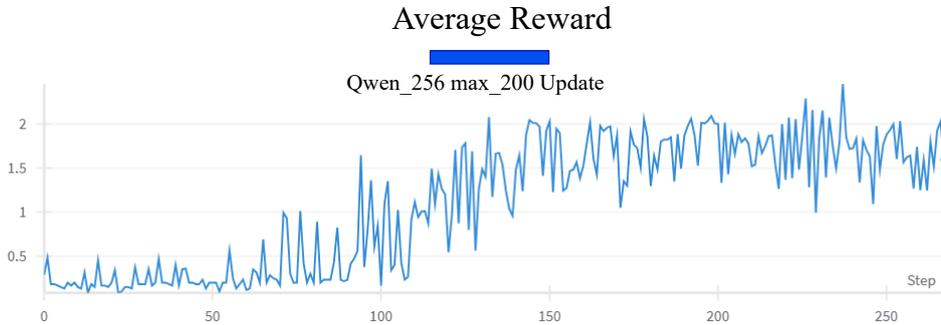


图 4-3 Qwen-2.5-7B-Instruct 在 256 个新 Token 的设定下平均奖励的变化曲线

在验证自然语言推理任务上的表现外，本文还使用了 LLaDA 8B 的开源训练数据对 GRPO-diffu Dream 7B 模型进行了后训练，并对常见的数学基准 GSM8K、MATH-500，以及多学科基准 GPQA 进行了测试。

GSM8K: 该数据集包含 8500 道小学至初中阶段的文字数学题，每题需 2–8 步基本算术操作得出最终数值答案。数据以 jsonl 形式提供，每条记录包括 question 与 answer 字段。本试验仅使用了其中 1000 道测试题进行测试，

MATH-500: 该数据集中包含 500 道来自高中数学竞赛水平的题目，主题覆盖代数、数论、几何、概率等。每题以 LaTeX 格式给出，模型需按照要求输出完整解题过程，并将最终答案直接给出。

GPQA: 该数据集一共 448 题，由博士或博士候选人在生物、化学、物理领域撰写，专为评估模型在 ‘Google-proof’ 条件下的推理能力而设计。每题为多项选择题格式，正确答案由领域专家确定。

Model	GRPO-diffu Dream 7B*	Dream 7B	LLaDA 8B	Qwen 2.5 7B
GSM8K	79.0*	77.2	70.9	78.9
MATH500	40.8*	39.6	30.7	41.1
GPQA	35.8	36.6*	30.5	35.5
Countdown	25.8*	16.0	13.2	6.2

表 4-1 模型在数学基准上的表现。序列长度为 128

可以发现尽管 GRPO-diffu Dream 在自然语言推理任务的训练和生成中无法稳定，并无法获得和自回归模型同水平的结果，但是强调推理能力，但在输出格

式较为简单的问答题测试中，GRPO-diffu Dream 超越了同量级的 Qwen 2.5 7B，并在一众扩散语言模型中有着极强的竞争力。

4.2 扩散模型生成速度与质量

在实验中作者发现扩散模型的去噪过程极其缓慢，和 Dream 模型^[6]中在 Countdown 任务^[42]上进行评测的结果差异很大。为此，作者在 MedCoT 任务上评估了 Qwen 模型，以及 Dream 模型在思维链任务上的生成速率以及生成质量，具体结果如图 4-4 所示。作者对扩散时间步设置成 1,2,4,8,16,32,128 时在 MedCoT 上每秒输出响应数量 (instance/s) 以及最终的问题回答准确率进行了评测，发现在执行了 128 步降噪后，Dream 模型依然和 Qwen2.5 模型差距显著。同时只有在去噪步骤小于 16 步时，Dream 模型才对 Qwen 模型有生成速度优势。

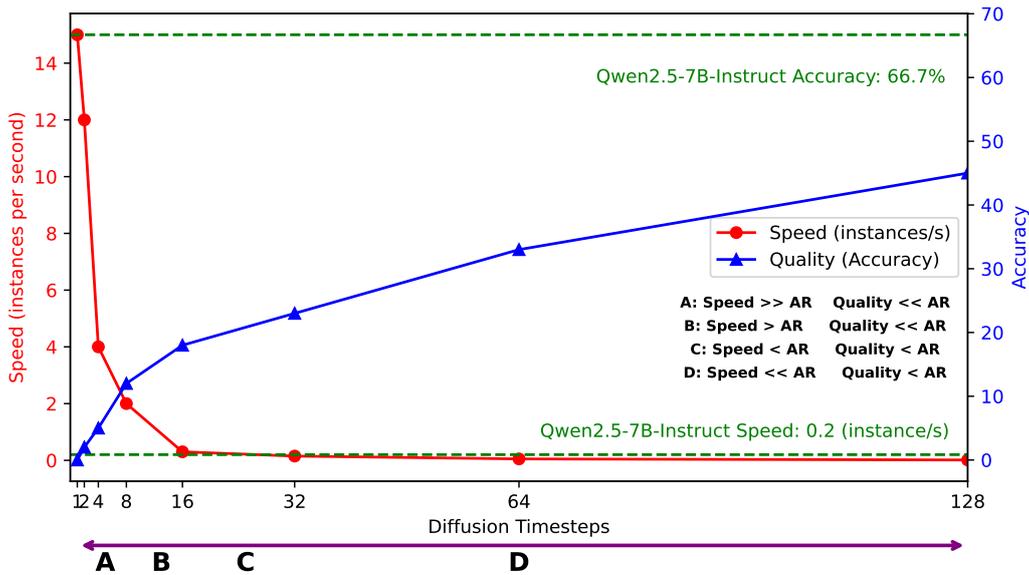


图 4-4 Med CoT: 医学思维链推理任务中的输出质量与速度

可以见得在该类复杂的，需要长思维链推理的任务中，以 Dream 模型为代表的 MDM 相比以 Qwen 模型为代表的自回归模型相比没有优势。但是在数学任务、编程任务等需要强大规划能力以及推理灵活度的任务中，MDM 依然具有很强的竞争力^[6]。

4.3 有效性验证与消融实验

为了实验和结论的全面性，本文也尝试实现以下三个实验：Python 代码补全、块级强化学习建模的有效性验证，以及奖励函数的消融实验。但截止论文提交时并未跑通，后续跑通代码后会提交到摘要中给出的 github 仓库。

Python 代码补全

数据集：CodeSearchNet 的 Python 代码补全子集。来自 Husain 等人^[43]在 2019 年提出的基准数据集工作；

数据格式：详见附录 B

该代码还未跑通，调用 Python 解释器出现问题，可能可以考虑调用沙盒 (Sandbox) 来进行验证并拿到奖励（类似 LLM Agent）。

块级强化学习建模的有效性

全新设计的块级强化学习中设定的不同是否可以缓解直接使用自回归模型强化学习算法所导致的 KL 散度不稳定的现象？进而使得扩散模型也可以获得稳定的强化学习训练呢？这需要后续的实验在现有 LLM 强化学习算法建模的基础上迁移到上述块级强化学习建模中去，尤其是对 KL 散度进行特别设计，允许扩散模型在 KL 散度上存在震荡现象，但仍可以学到奖励设定中的偏好。

奖励函数消融实验

本实验由于训练计算资源开销过大，未对奖励函数进行消融实验。可能会因为奖励函数设置不合理或冗余出现奖励收敛较慢或不收敛的情况，并出现的可能会出现奖励黑客攻击的问题。同时可能需要针对扩散模型在有限去噪步骤时生成的框架做出一些更特殊的奖励函数设置。

4.4 结论

综合扩散掩码模型的特性及现有实验和分析，可以得到以下三个结论：

首先在**模型性能**方面，当前扩散掩码模型在众多基准集上逼近甚至超越了同参数量级的自回归模型，这往往是因为如今许多 LLM 的基准测试集的结果中都是有一个真实值，而并没有涉及到复杂的、多段落自然语言推理任务，进而使得在该类基准集上两类模型效果近似。但如若把这些任务进行分类，可以发现扩散模型通常在需要统筹规划任务（如数学、编程）任务上表现良好，但在复杂自然语言推理任务中表现不佳。故作者认为扩散掩码模型的出现是**补充而不是取代**因果建模的自回归 LLM（如 ChatGPT o4-mini-high 和 DeepSeek-R1）中的长思维链输出偏好，且扩散掩码模型对应的可调节的计算质量权衡代表了与传统最回归模型框架相比的独特优势。

其次是**建模方式**。扩散模型由于其特殊的建模方式：通过双向上下文建模抽取更丰富地集成来自两个方向的信息，增强生成文本的全局连贯性。本篇文章中引入的条件概率分层建模方式通过条件扩散模型引入其他的条件变量，或控制去噪步骤数量来迭代细化过程，使其具有更加灵活可控的生成能力。

最后在**强化学习后训练的稳定性**。扩散模型如果想要基于强化学习进行对齐，或利用价值函数评价反向过程过程中的中间状态，仍然需要较高的扩散时间步后生成稳定的去噪结果，比较稳定的损失（也通常是稳定的 KL 散度）是强化学习算法可以收敛的基础，且使用扩散模型生成的框架进行高效引导在扩散时间步过小时并不现实。为了实现并验证这一点，首先需要保证有竞争力的输出速率，可能需要更多的计算资源进行更高效的并行推理。如 Mercury Coder 部署在 NVIDIA A100 集群上，Diffusion 的去噪过程并行化系数很高，在保证推理质量的情况下速率得到了限制提升。此外，扩散模型由于其逐步优化噪声的特殊文本生成方式，使用结果奖励模型的奖励赋值方式，以及引入 KL 散度作为参考模型和策略模型间的差距并不合理，需要设计更适配于扩散掩码模型的强化学习算法。本文中引入的块级条件概率分层建模强化学习的方式为解决这一问题提出了初步看法，仍需后续实验验证其有效性。

第五章 总结与展望

5.1 不足之处

由于数据质量以及计算资源受限，本实验中未加入冷启动进行 SFT 过程，而是直接基于基座模型进行 RL 训练。故可能会产生由于策略模型初始化而导致的 RL 训练不稳定的问题。同时尽管提出了离散扩散模型的块级分层概率建模方式以进一步释放 MDM 的并行能力，但并未完成实验验证这一方式对应的强化学习的有效性。此外，现有实验中强化学习的奖励模型的设置并没有进行消融实验验证各个奖励设置的功能和有效性。尽管整体平均奖励收敛，但更好的奖励函数设置往往会让奖励收敛地更迅速，并学到更准确的奖励偏好。

5.2 强化学习在 LLM 后训练中是否真正有效？是否能通向 AGI？

无论是可验证代码生成，还是基于 Lean 做数学定理证明，**RL 是否能达到真正的可验证的 AI?** 作者认可 RL 在大多数情况下会有效果，基于 RL 的后训练要想达到真正的可验证的 AI 依然很难。作者认为如若按照现在已有的技术路径进行深入，当前 SFT 得到的模型能力依然不足以掌控强化学习算法，也即无法通过强化学习的能力真正在物理世界中通过得到的反馈自我更新。进而当前范式在跑通流水线后很有可能也是在调参，把基准上的数据多提升几个点，而不能达到 AGI 可以进行自我学习（多智能体间相互竞争并学习）的畅享。

DeeepSeek-Math 提出了 SFT 和一些经典 RL 算法之间的联系，并尝试提出了一个统一范式，如图 5-1 所示。在该公式中“Data Source”项说明了从分布 D 中采样出的训练数据；“Gradient Coefficient”项说明了算法 \mathcal{A} 在处理训练数据数据和得到奖励信号后，对这个采样轨迹进行的惩罚或者奖励。文章中也对 SFT, RFT, Online RFT, DPO, PPO, GRPO 是如何选择不同的“Data Source”、“Reward Function”，和“Gradient Coefficient”进行了说明，这里不再赘述。

基于此，本文认为当前的 RL 算法和 SFT 一样，本质上是基于损失的梯度下

$$\nabla_{\theta} \mathcal{J}_{\mathcal{A}}(\theta) = \mathbb{E}[\underbrace{(q, o)}_{\text{Data Source}} \sim \mathcal{D}] \left(\frac{1}{|o|} \sum_{t=1}^{|o|} \underbrace{GC_{\mathcal{A}}(q, o, t, \pi_{rf})}_{\text{Gradient Coefficient}} \nabla_{\theta} \log \pi_{\theta}(o_t | q, o_{<t}) \right).$$

图 5-1 统一范式：统一 SFT, RFT, Online RFT, DPO, PPO and GRPO 的优化式

降算法 (Swamy 等人^[44]也通过实验得出了类似的结论) 调整 SFT 训练后的概率分布, 进而加上了 RL 的过程是将 LLM 的搜索空间过滤到相对简单的策略模型最优策略的一个子集之中。这一定对最终效果有帮助, 但在 LLM 无比强大的年代, 文章中体现出加入 RL 后的结果往往是减少搜索空间, 让 LLM 拿到更好的结果可能性变大, 并带来一两个百分点的提升。至少在目前 RL 所具有的探索的能力依然在很多很复杂的任务中没有体现优势。

同时 Yue 等人^[45]经过强化学习算法 (文章中称 RLVR, Reinforcement Learning with Verifiable Rewards) 微调后的模型往往更专注于高分路径, 模型在探索更广泛思路方面的表现反而弱于基线, 长期看其推理能力的广度实际上比未经 RL 的模型要窄, 这也进一步损失了模型的泛化性能。图 5-2 中为原文中的示意图, 其中展示了尽管 RLVR model 在部分样本的搜索过程中相比 Base Model 更加高效, 但其在遇到新的样本需要泛化时, 其依然偏好选择学到了的高奖励的搜索路径, 使得其推理能力的广度受到损失。

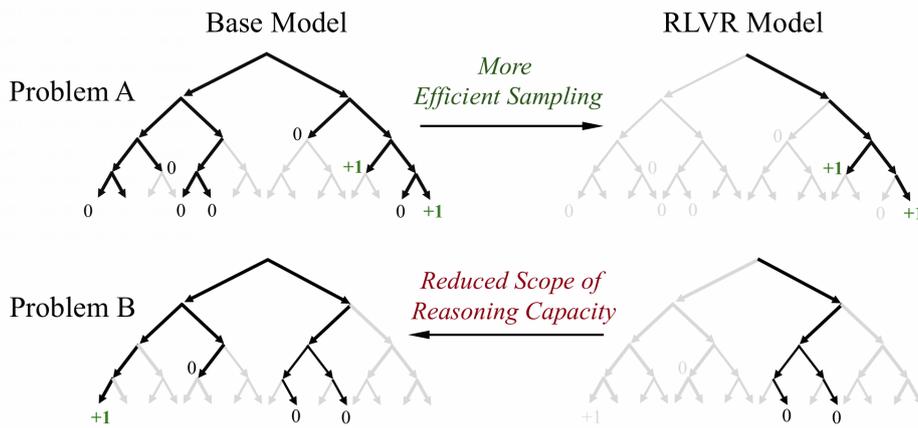


图 5-2 拓展广度：基础模型 vs. 后训练微调的模型

目前被认为非常有前景的基于强化学习后训练的推理语言模型是否是通往 AGI 或者真正 verified AI 的途径? 也许我们要利用符号主义的思想? 不光是符号执行工具, 甚至定义一个可扩放的公理系统? 作者也希望在研究生阶段可以继续

探索。

5.3 扩散模型所代表的非自回归模型能否挑战自回归模型？

掩码扩散模型通过逐步去噪来学习生成输出，这意味着他们可以非常快速地在充足的 GPU 集群中迭代优化出解决方案，并在生成过程中纠正错误。这有助于他们在编辑等任务中表现出色，包括在数学和代码的上下文中。同时其可以生成更连贯的文本。其可以一次生成整个标记块，这意味着它比自回归模型更连贯地响应用户的提示。这两个特征源自其独特的建模方式，而这是基于因果建模的自回归模型所不具备的独特优势。作者相信控制力 (Controllability)、创造力 (Creativity) 和更快的文本生成速度会是扩散模型所代表的非自回归模型挑战自回归模型的重要方面。

从香港大学团队提出的 7B Dream，到人民大学高瓴团队提出的 8B 模型 LLaDA，到 Mercury Code，以及 5 月底谷歌新提出的 Gemini-Diffusion 模型，当前扩散掩码模型对应的扩展定律 (Scaling Law) 也出现苗头。如若未来 MDM 可以在扩展定律中性能得到跃迁，则扩散掩码模型和因果建模的自回归模型的融合，甚至超越语言在多模态中的融合，将会极大消除语言、音频、视频等不同模态之间的数据差异，使得不同模态的训练数据可以共享在一个通用的训练骨架 (Backbone) 中。为此，作者相信训练的数据效率将会得到量级化的提升，并将极大提升模型的性能、感知物理世界的能力，与通过强化学习算法进行自主学习的能力。

参考文献

- [1] VASWANI A, SHAZEER N, PARMAR N, USZKOREIT J, JONES L, GOMEZ A N, KAISER Ł, POLOSUKHIN I. Attention is all you need[C] // Advances in Neural Information Processing Systems: vol. 30. 2017.
- [2] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[C] // Advances in Neural Information Processing Systems: vol. 25. 2012.
- [3] YAO S. The Second Half[Z]. <https://ysmyth.github.io/The-Second-Half/>. [Online; accessed 13-May-2025]. 2025.
- [4] ZHANG Q, LYU F, SUN Z, WANG L, ZHANG W, HUA W, WU H, GUO Z, WANG Y, MUENNIGHOFF N, et al. A Survey on Test-Time Scaling in Large Language Models: What, How, Where, and How Well?[J]. arXiv preprint arXiv:2503.24235, 2025.
- [5] Qwen, : YANG A, YANG B, ZHANG B, HUI B, ZHENG B, YU B, LI C, LIU D, HUANG F, WEI H, LIN H, YANG J, TU J, ZHANG J, YANG J, YANG J, ZHOU J, LIN J, DANG K, LU K, BAO K, YANG K, YU L, LI M, XUE M, ZHANG P, ZHU Q, MEN R, LIN R, LI T, TANG T, XIA T, REN X, REN X, FAN Y, SU Y, ZHANG Y, WAN Y, LIU Y, CUI Z, ZHANG Z, QIU Z. Qwen2.5 Technical Report[EB/OL]. 2025. <https://arxiv.org/abs/2412.15115>. arXiv: 2412.15115 [cs.CL].
- [6] YE J, XIE Z, ZHENG L, GAO J, WU Z, JIANG X, LI Z, KONG L. Dream 7B [EB/OL]. 2025. <https://hkunlp.github.io/blog/2025/dream>.
- [7] SOHL-DICKSTEIN J, WEISS E, MAHESWARANATHAN N, GANGULI S. Deep unsupervised learning using nonequilibrium thermodynamics[C] // International Conference on Machine Learning. 2015: 2256-2265.

- [8] HO J, JAIN A, ABBEEL P. Denoising diffusion probabilistic models[C]// Advances in Neural Information Processing Systems: vol. 33. 2020: 6840-6851.
- [9] SONG Y, SOHL-DICKSTEIN J, KINGMA D P, KUMAR A, ERMON S, POOLE B. Score-Based Generative Modeling through Stochastic Differential Equations[C]// International Conference on Learning Representations. 2021.
- [10] ROMBACH R, BLATTMANN A, LORENZ D, ESSER P, OMMER B. High-resolution image synthesis with latent diffusion models[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 10684-10695.
- [11] RAMESH A, DHARIWAL P, NICHOL A, CHU C, CHEN M. Hierarchical text-conditional image generation with clip latents[J]. arXiv preprint arXiv:2204.06125, 2022, 1(2): 3.
- [12] SAHARIA C, CHAN W, SAXENA S, LI L, WHANG J, DENTON E L, GHASEMIPOUR K, GONTIJO LOPES R, KARAGOL AYAN B, SALIMANS T, et al. Photorealistic text-to-image diffusion models with deep language understanding[C]// Advances in Neural Information Processing Systems: vol. 35. 2022: 36479-36494.
- [13] CHEN N, ZHANG Y, ZEN H, WEISS R J, NOROUZI M, CHAN W. Wavegrad: Estimating gradients for waveform generation[J]. arXiv preprint arXiv:2009.00713, 2020.
- [14] KONG Z, PING W, HUANG J, ZHAO K, CATANZARO B. Diffwave: A versatile diffusion model for audio synthesis[J]. arXiv preprint arXiv:2009.09761, 2020.
- [15] BAR-TAL O, CHEFER H, TOV O, HERRMANN C, PAISS R, ZADA S, EPHRAT A, HUR J, LIU G, RAJ A, et al. Lumiere: A space-time diffusion model for video generation[C]// SIGGRAPH Asia 2024 Conference Papers. 2024: 1-11.
- [16] BAO F, XIANG C, YUE G, HE G, ZHU H, ZHENG K, ZHAO M, LIU S, WANG Y, ZHU J. Vidu: a highly consistent, dynamic and skilled text-to-video generator

- with diffusion models[J]. arXiv preprint arXiv:2405.04233, 2024.
- [17] YANG D, YU J, WANG H, WANG W, WENG C, ZOU Y, YU D. Diffsound: Discrete diffusion model for text-to-sound generation[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2023, 31: 1720-1733.
- [18] GRUVER N, STANTON S, FREY N, RUDNER T G, HOTZEL I, LAFRANCE-VANASSE J, RAJPAL A, CHO K, WILSON A G. Protein design with guided discrete diffusion[C] // Advances in Neural Information Processing Systems: vol. 36. 2023: 12489-12517.
- [19] LI X, THICKSTUN J, GULRAJANI I, LIANG P S, HASHIMOTO T B. Diffusion-lm improves controllable text generation[C] // Advances in Neural Information Processing Systems: vol. 35. 2022: 4328-4343.
- [20] GONG S, LI M, FENG J, WU Z, KONG L. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models[C] // International Conference on Learning Representations. 2022.
- [21] REID M, HELLENDORF V J, NEUBIG G. Diffuser: Discrete diffusion via edit-based reconstruction[J]. arXiv preprint arXiv:2210.16886, 2022.
- [22] AUSTIN J, JOHNSON D D, HO J, TARLOW D, VAN DEN BERG R. Structured denoising diffusion models in discrete state-spaces[C] // Advances in Neural Information Processing Systems: vol. 34. 2021: 17981-17993.
- [23] GONG S, AGARWAL S, ZHANG Y, YE J, ZHENG L, LI M, AN C, ZHAO P, BI W, HAN J, et al. Scaling Diffusion Language Models via Adaptation from Autoregressive Models[J]. 2024.
- [24] NIE S, ZHU F, YOU Z, ZHANG X, OU J, HU J, ZHOU J, LIN Y, WEN J R, LI C. Large Language Diffusion Models[J]. arXiv preprint arXiv:2502.09992, 2025.
- [25] SHI J, HAN K, WANG Z, DOUCET A, TITSIAS M. Simplified and generalized masked diffusion for discrete data[C] // Advances in Neural Information Processing Systems: vol. 37. 2024: 103131-103167.

- [26] CAMPBELL A, BENTON J, DE BORTOLI V, RAINFORTH T, DELIGIAN-NIDIS G, DOUCET A. A continuous time framework for discrete denoising models[C]// Advances in Neural Information Processing Systems: vol. 35. 2022: 28266-28279.
- [27] SUN H, YU L, DAI B, SCHUURMANS D, DAI H. Score-based Continuous-time Discrete Diffusion Models[C]// The Eleventh International Conference on Learning Representations.
- [28] MNIH V, BADIA A P, MIRZA M, GRAVES A, LILLICRAP T, HARLEY T, SILVER D, KAVUKCUOGLU K. Asynchronous methods for deep reinforcement learning[C]// International Conference on Machine Learning. 2016: 1928-1937.
- [29] GUO S, ZHANG B, LIU T, LIU T, KHALMAN M, LLINARES F, RAMÉ A, MESNARD T, ZHAO Y, PIOT B, et al. Direct Language Model Alignment from Online AI Feedback[J]. CoRR, 2024.
- [30] ZHANG D, HUANG X, ZHOU D, LI Y, OUYANG W. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b[J]. arXiv preprint arXiv:2406.07394, 2024.
- [31] KOH J Y, MCALEER S, FRIED D, SALAKHUTDINOV R. Tree search for language model agents[J]. arXiv preprint arXiv:2407.01476, 2024.
- [32] REN Z, SHAO Z, SONG J, XIN H, WANG H, ZHAO W, ZHANG L, FU Z, ZHU Q, YANG D, et al. DeepSeek-Prover-V2: Advancing Formal Mathematical Reasoning via Reinforcement Learning for Subgoal Decomposition[J]. arXiv preprint arXiv:2504.21801, 2025.
- [33] ARRIOLA M, GOKASLAN A, CHIU J T, YANG Z, QI Z, HAN J, SAHOO S S, KULESHOV V. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models[J]. arXiv preprint arXiv:2503.09573, 2025.
- [34] LUO C. Understanding diffusion models: A unified perspective[J]. arXiv preprint arXiv:2208.11970, 2022.

- [35] BLACK K, JANNER M, DU Y, KOSTRIKOV I, LEVINE S. Training diffusion models with reinforcement learning[J]. arXiv preprint arXiv:2305.13301, 2023.
- [36] SUI Y, CHUANG Y N, WANG G, ZHANG J, ZHANG T, YUAN J, LIU H, WEN A, ZHONG S, CHEN H, et al. Stop overthinking: A survey on efficient reasoning for large language models[J]. arXiv preprint arXiv:2503.16419, 2025.
- [37] SHOJAEE P, JAIN A, TIPIRNENI S, REDDY C K. Execution-based Code Generation using Deep Reinforcement Learning[J]. Transactions on Machine Learning Research, 2023.
- [38] BAXTER I D, YAHIN A, MOURA L, SANT'ANNA M, BIER L. Clone detection using abstract syntax trees[C] // Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272). 1998: 368-377.
- [39] SHAO Z, WANG P, ZHU Q, XU R, SONG J, BI X, ZHANG H, ZHANG M, LI Y, WU Y, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models[J]. arXiv preprint arXiv:2402.03300, 2024.
- [40] XIN R, XI C, YANG J, CHEN F, WU H, XIAO X, SUN Y, ZHENG S, SHEN K. BFS-Prover: Scalable Best-First Tree Search for LLM-based Automatic Theorem Proving[J]. arXiv preprint arXiv:2502.03438, 2025.
- [41] CHEN J, CAI Z, JI K, WANG X, LIU W, WANG R, HOU J, WANG B. Huatuogpt-o1, towards medical complex reasoning with llms[J]. arXiv preprint arXiv:2412.18925, 2024.
- [42] YE X, YIN F, HE Y, ZHANG J, YEN H, GAO T, DURRETT G, CHEN D. LongProc: Benchmarking Long-Context Language Models on Long Procedural Generation[J]. arXiv preprint arXiv:2501.05414, 2025.
- [43] HUSAIN H, WU H H, GAZIT T, ALLAMANIS M, BROCKSCHMIDT M. Codesearchnet challenge: Evaluating the state of semantic code search[J]. arXiv preprint arXiv:1909.09436, 2019.
- [44] SWAMY G, CHOUDHURY S, SUN W, WU Z S, BAGNELL J A. All Roads Lead to Likelihood: The Value of Reinforcement Learning in Fine-Tuning[J].

arXiv preprint arXiv:2503.01067, 2025.

- [45] YUE Y, CHEN Z, LU R, ZHAO A, WANG Z, SONG S, HUANG G. Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?[J]. arXiv preprint arXiv:2504.13837, 2025.

致 谢

四年时间白驹过隙，让我从青涩逐渐成长起来。一路上很幸运遇到了许许多多的老师和同学，在我遇到低谷时给予我激励使我振作，在我迷茫困惑时给予我您们宝贵的人生的体验与经验。

感谢赵一铮老师与吴萱学姐让我第一次找到“知识表示与处理”这个令我兴奋的科研方向，带我第一次了解科研的全过程。感谢香港大学的齐晓娟老师在我于香港大学交换时给予我学业以及生活上的莫大帮助。在我对未来感到迷茫时分享了许多自己亲身的经历，让我逐渐明晰了不同发展路径的可能性。此外，也非常感谢耶鲁大学的徐华老师给予我远赴美国进行暑期科研的机会并在我刚到美国时给予的生活上的点滴帮助，在这难得的 6 个月中，我深深体验了一个完全不一样的生活、学术与科研氛围。也感谢徐老师的信任，让我第一次作为项目的负责人主导一个科研项目，并提供了很多难得的平台，和众多美国一流高校的教授与博士后们交流学术和未来发展！三位老师在 25 Fall 的申请季都对我的申请帮助良多，在此衷心感谢各位老师！

感谢一路以来在科研合作中遇到的好朋友们，非常幸运在耶鲁大学暑期科研期间遇到了许许多多可爱有趣的博士后们：西安交大的贺欢博士、中科大的李安然博士、澳大利亚国立大学的秦震岳博士、华盛顿大学的周伟鹏博士、香港城市大学的陈阵博士、复旦大学的彭雪晴博士、武汉大学的谢倩倩博士（现在得叫谢教授了哈哈），以及浙江大学的马凡博士等等。他们都在耶鲁大学暑期科研期间给予我生活、科研以及申请中莫大的帮助，希望未来仍有机会一起合作玩耍！此外，感谢耶鲁大学的 Vipina K. Keloth 博士、洪娜博士、钱凌飞博士以及陈庆宇教授、哈佛大学的 Timothy A. Miller 教授、佛罗里达大学的 Jiang Bian 教授在科研合作中给予的指导，是您们的帮助和指导让项目的推动非常顺利，您们在 Clinical NLP 以及医学领域的独特见解让我知悉了纯粹的 AI 技术领域外的诸多知识！

在毕业设计的完成过程中，非常感谢葛存菁老师肯定我的毕设想法并提出

详细的建议，并在我两次更换毕业设计方向时给予建议与计算资源的支持。同时也非常感谢上海人工智能实验室的付杰老师，在我并不了解强化学习这一领域的时候给予了我耐心的指导与帮助，并在实习与毕业设计时给予我方向上的指引，让我了解到 AI 领域更多的可能性以及未来的发展方向！也感谢当前在快手实习时公司以及纪兴光 Mentor 给予充足的计算资源支持与宽松的学习环境，虽然入职较晚结果没来得及跑出来，但这对我在毕设答辩以及论文实验的完整性有很大的帮助！最后也很感谢 LUG@NJU提供的南京大学毕业设计的 Latex 模板，让我把大部分的精力放在论文内容而无需考虑格式规范性的问题。

从 Neuro Symbolic Learning and Reasoning, Lean Math Theorem Proving 到当下 RL 与 Formal Language, Diffusion Model，探索毕业论文方向的过程还是蛮曲折的。完成毕业论文的过程中作为本科教育的最后一站，我依然选择了一个自己觉得很有意义且很感兴趣，但却并不太了解的领域。在这个过程中虽然困难，但也让我学到了非常多的知识，结识了良师益友，也了解到原来 AI（尤其是 LLM）领域还有这么多值得探索的方向。“勇敢地走出舒适区，去见更大的世界”这一理念贯穿了我大学四年的求学过程，从南京大学步入 CS/AI 的殿堂，到香港大学交换、耶鲁大学暑研，到在上海 AI Lab 与快手实习。这一路虽有诸多遗憾，但我也已感这四年已足够精彩，感谢这一路爸妈的支持陪伴与一直努力的自己！

希望在未来的学习工作中我可以找到一条可以和自我价值和解的路，不再患得患失，自洽地好好生活。与您共勉！

附录 A 案例分析 1 & 2

Status	Input: LLM Prompt (without "role")	Output: Dafny Script	Feedback: Dafny Compiler Signal
Natural Language Description	Prompt (请见第 3.1 节)	<pre>// Dafny 代码示例 class Solution { method fib(n: nat) returns (res: nat){ // 对n的要求: n是自然数 (nat 即 >= 0) // 终止度量: decreases n // 后置条件: 保证返回值与斐波那契定义一致 requires n >= 0 ensures res == (if n < 2 then n else fib(n - 1) + fib(n - 2)) decreases n if n == 0 { return 0;} else if n == 1 { return 1;} else { return fib(n - 1) + fib(n - 2);}}</pre>	<p>Error 1: expression is not allowed to invoke a method (fib)</p> <p>Error 2: method call is not allowed to be used in an expression context (fib)</p>
Dafny Script with Compiler Feedbacks	<p>Error 1: expression is not allowed to invoke a method (fib)</p> <p>Error 2: method call is not allowed to be used in an expression context (fib)</p>	<pre>// 定义类 Solution class Solution { // 第一步: 用“函数”纯逻辑地定义斐波那契, 用于规范(spec)和证明 function fibSpec(n: nat): nat decreases n{ if n < 2 then n else fibSpec(n - 1) + fibSpec(n - 2)} // 第二步: 用“方法”来实际执行斐波那契的计算 method fib(n: nat) returns (res: nat){ requires n >= 0 ensures res == fibSpec(n) // 注意: ensures 中只调用 fibSpec 函数 decreases n // 声明递归调用的终止度量 if n == 0 { return 0;} else if n == 1 { return 1;} else { // 这里在方法体中, 可以调用自身方法, 因为这是在可执行上下文里 var a := fib(n - 1); var b := fib(n - 2); return a + b;}}</pre>	Dafny program successfully verified

表 A-1 Easy example with Dafny Process and Signals

Status	Input: LLM Prompt (without "role")	Output: Dafny Script	Feedback: Dafny Compiler Signal
Natural Language Description	<p>题目描述: 斐波那契数 (通常用 $F(n)$ 表示) 形成的序列称为斐波那契数列。该数列由 0 和 1 开始, 后面的每一项数字都是前面两项数字的和。也就是: $F(0) = 0$, $F(1) = 1$, $F(n) = F(n - 1) + F(n - 2)$, 其中 $n > 1$。给定 n, 请计算 $F(n)$。示例: leetcode 中共 3 个, 此处略。</p> <p>Python 代码: 该问题对应的 Python 代码实现为 (略)。</p> <p>请根据 Dafny 的语法和题目语义, 将该代码转化为可通过 Dafny 编译检查的代码</p>	<pre>// Dafny 代码示例 class Solution { method fib(n: nat) returns (res: nat){ // 对n的要求: n是自然数 (nat 即 >= 0) // 终止度量: decreases n // 后置条件: 保证返回值与斐波那契定义一致 requires n >= 0 ensures res == (if n < 2 then n else fib(n - 1) + fib(n - 2)) decreases n if n == 0 { return 0;} else if n == 1 { return 1;} else { return fib(n - 1) + fib(n - 2);}}</pre>	<p>Error 1: expression is not allowed to invoke a method (fib)</p> <p>Error 2: method call is not allowed to be used in an expression context (fib)</p>
Dafny Script with Compiler Feedbacks	<p>Error 1: expression is not allowed to invoke a method (fib)</p> <p>Error 2: method call is not allowed to be used in an expression context (fib)</p>	<pre>// 定义类 Solution class Solution { // 第一步: 用“函数”纯逻辑地定义斐波那契, 用于规范(spec)和证明 function fibSpec(n: nat): nat decreases n{ if n < 2 then n else fibSpec(n - 1) + fibSpec(n - 2)} // 第二步: 用“方法”来实际执行斐波那契的计算 method fib(n: nat) returns (res: nat){ requires n >= 0 ensures res == fibSpec(n) // 注意: ensures 中只调用 fibSpec 函数 decreases n // 声明递归调用的终止度量 if n == 0 { return 0;} else if n == 1 { return 1;} else { // 这里在方法体中, 可以调用自身方法, 因为这是在可执行上下文里 var a := fib(n - 1); var b := fib(n - 2); return a + b;}}</pre>	<p>Dafny program successfully verified</p>

Table 2: Easy example with Dafny Process and Signals

图 A-1 Intractable example, which could not pass Dafny’s verification

附录 B 数据格式

MedCot-reasoning-SFT

Question: A 36-year-old man was found to have a transient, low-volume, right-to-left shunt through the atrial septum during coughing, indicating a failure in the closure process similar to another embryologic condition. What condition, characterized by a similar embryonic development failure, involves the incomplete closure along the genital tubercle structure in males?

Complex_CoT: *Okay, let's think about this.* We've got a guy who's 36 years old, and during a cough, he's showing a bit of a shunt through the atrial septum. That sounds like what's known as a patent foramen ovale, or PFO, where the septum doesn't quite get the memo to fully close during development. This PFO is literally a leftover flap between the atria, and it takes me back to embryology. *Are there any similar stories of closure failures in other parts of the body?*

Wait a minute, what about something related to the genital region in males? Hmm, yes, hypospadias comes to mind. It's that condition where the urethral opening isn't quite where you'd expect it to be, because again, things didn't close up as they were supposed to. *So, just to wrap it up nicely,* the condition in males analogous to the incomplete atrial closure we see in PFO, due to similar developmental hiccups, is hypospadias. They both share that underlying theme of incomplete closure. *Yep, that fits well.*

Response: The condition in males that involves an incomplete closure along the genital tubercle, similar to the embryologic development failure seen in a patent foramen ovale (PFO), is hypospadias. Both conditions result from a failure of certain structures to close properly during embryonic development. In hypospadias, the urethral opening is not at the typical location on the tip of the penis due to incomplete fusion of the urogenital folds.

Answer: Hypospadias

以下是 CodeSearchNet 数据集中 Python Code Completion 的数据结构。

```
{'code': 'def get_vid_from_url(url):\n'\n    '    """Extracts video ID from URL.\n'\n    '    """\n'\n    '    return match1(url, r'youtu\\.be/([^/?/]+)') or \\n"\n    '    match1(url, r'youtube\\.com/embed/([^/?/]+)') or \\n"\n    '    match1(url, r'youtube\\.com/v/([^/?/]+)') or \\n"\n    '    match1(url, r'youtube\\.com/watch/([^/?/]+)') or \\n"\n    '    parse_query_param(url, 'v') or \\n"\n    '    parse_query_param(parse_query_param(url, 'u'), 'v')",\n'code_tokens': ['def',\n                'get_vid_from_url',\n                '(',\n                'url',\n                ')',\n                ':',\n                'return',\n                'match1',\n                '(',\n                'url',\n                ',',\n                'r'youtu\\.be/([^/?/]+)',\n                ')',\n                'or',\n                'match1',\n                '(',\n                'url',\n                ',',\n                'r'youtube\\.com/embed/([^/?/]+)',\n                ')',\n                'or',\n                'match1',\n                '(',\n                'url',\n                ',',\n                'r'youtube\\.com/v/([^/?/]+)',\n                ')',\n                'or',\n                'match1',\n                '(',\n                'url',\n                ',',\n                'r'youtube\\.com/watch/([^/?/]+)',\n                ')',\n                'or',\n                'parse_query_param',\n                '(',
```

```

        'url',
        ', ',
        "'v'",
        ') ',
        'or',
        'parse_query_param',
        '(',
        'parse_query_param',
        '(',
        'url',
        ', ',
        "'u'",
        ') ',
        ', ',
        "'v'",
        ') ]',
'docstring': 'Extracts video ID from URL.',
'docstring_tokens': ['Extracts', 'video', 'ID', 'from', 'URL', '.'],
'func_name': 'YouTube.get_vid_from_url',
'language': 'python',
'original_string': 'def get_vid_from_url(url):\n'
                    '    """Extracts video ID from URL.\n'
                    '    """\n'
                    '    return match1(url, r'youtu\\.be/([^?/]+)') or \\n"\n'
                    '    match1(url, r'youtube\\.com/embed/([^/?]+)') or "\n'
                    '    match1(url, r'youtube\\.com/v/([^/?]+)') or \\n"\n'
                    '    match1(url, r'youtube\\.com/watch/([^/?]+)') or "\n'
                    '    parse_query_param(url, 'v') or \\n"\n'
                    '    parse_query_param(parse_query_param(url, 'u'), "\n'
                    '    'v')',
'partition': 'test',
'path': 'src/you_get/extractors/youtube.py',
'repo': 'soimort/you-get',
'sha': 'b746ac01c9f39de94cac2d56f665285b0523b974',
'url': 'https://github.com/soimort/you-get/blob/\n'
       'b746ac01c9f39de94cac2d56f665285b0523b974/src/you_get/extractors/youtube.py#L135-\n'
       'L143'}

```

