学习起草：通过强化学习实现自适应推测解码

Jiebin Zhang[♠♡*] **Zhenghan Yu**[♠♡] **Liang Wang**[♢] **Nan Yang**[♢] **Eugene J. Yu**[♠] **Zheng Li**[♠]**Yifan Song**[♠] **Dawei Zhu**[♠] **Xingxing Zhang**[♢] **Furu Wei**[♢] **李素建**[♠♡]
[♠]多媒体信息处理国家重点实验室，北京大学[♡]计算机科学学院，北京大学[♢]微软亚洲研究院{zhangjiebin,lisujian}@pku.edu.cn
wangliang@microsoft.com
https://github.com/zhzihao/Learning-to-Draft

arXiv:2603.01639v1 [cs.CL] 2 Mar 2026

摘要

推测解码通过使用小型草稿模型为较大的目标模型生成候选词元以供验证，从而加速大语言模型（LLM）的推理。这项技术的效果取决于生成候选词元与验证候选词元所花费时间之间的权衡。然而，当前最先进的方法依赖静态的时间分配，而近期的动态方法则针对接受长度等代理指标进行优化，往往忽视真实的时间成本，并将草拟和验证阶段孤立处理。为了解决这些局限，我们提出了学习起草（LTD），一种直接优化每个草稿与验证周期吞吐量的新方法。我们将该问题构建为强化学习环境，并训练两种协同自适应策略，以动态协调草拟和验证阶段。这促使策略彼此适应，并明确地最大化解码效率。我们在五种不同的大语言模型和四种不同任务上进行了广泛评估。结果表明，LTD 的加速比为 2.24× 至 4.32×，最高比最先进的方法 Eagle3 高出 36.4%。

1 引言

现代大语言模型（LLM）日益广泛地用于执行审慎的多步推理，以解决复杂任务。这一范式拓展了它们在推理时的性能能力（Guo 等人，2025年；Team，2025b；Agarwal 等人，2025年）。然而，这些推理过程的复杂性会导致显著延迟，因此，如何在保持高质量输出的同时加速推理，日益成为研究重点（Sui 等人，2025年）。在这些努力中，推测解码是一种尤为有效的解决方案：它使用更小、更快的草稿模型生成词元序列，再由更大的目标模型并行验证。这种方法既能加速推理，又能保持输出分布不变（Leviathan 等人，2023年）。

草稿词元序列（简称“草稿”）的质量和结构对推测解码的整体性能至关重要。早期方法主要采用简单的链式结构草稿（Chen 等人，2023年；Leviathan 等人，2023年）。相比之下，近期研究表明，树状结构草稿能够显著增加每个草拟与验证周期中被接受的词元数量（接受长度）（Miao 等人，2023年；Li 等人，2024；Cai 等人，2024）。Eagle3（Li 等人，2025年）等最先进的方法采用静态的“一刀切”策略，通过人工调整来设定草稿树的大小。虽然一些研究尝试动态调整草稿树的大小（Brown 等人，2024；Zhang 等人，2024；Huo 等人，2025年；Huang 等人，2025年），但其方法通常依赖简单的启发式方法，或训练模型来优化接受长度这一指标——即每个验证步骤中平均正确猜中的词元数量。然而，最大化接受长度并不一定能加快整体生成速度，因为这完全忽略了草拟和验证过程本身的关键时间成本。真正的目标应该是最大化模型的吞吐量。否则，追求更长的

[*] Jiebin 在微软亚洲研究院实习期间完成的工作。李素建为通讯作者。

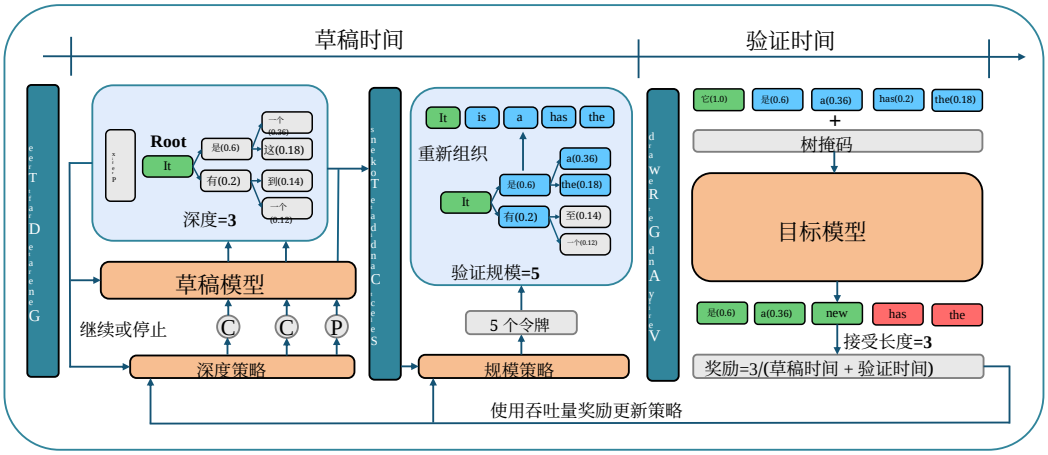


图 1：LTD 方法概览。我们将草稿模型与目标模型之间的交互建模为强化学习环境。我们采用两个策略，通过控制草稿树的深度和验证规模，动态管理草稿生成与目标验证之间的分配。这些策略利用基于每个草稿与验证周期吞吐量的奖励信号进行联合优化。

接受长度可能导致草稿树过大，其生成和验证所耗费的时间超过所节省的时间，最终拖慢整个过程。与此同时，现有方法的一个关键局限在于，草稿树的草拟与验证是孤立优化的。

本文认为，草拟与验证相互依存，应当协同优化。真正最优的策略必须兼顾时间，并能适应这两个阶段之间的相互作用。例如，在一个简单的文本生成场景中，如果只优化草拟阶段，而验证阶段保持不变，就会导致次优结果。即使草稿模型能够轻松生成较长且正确的序列，也不得不截断输出，以适应有限的验证成本。这会浪费起草能力，使其无法充分发挥加速潜力。在理想的联合优化中，草稿阶段会生成更长、更完整的序列，同时考虑到验证阶段会动态调整其资源。反过来，验证阶段在得知草稿较长后，也会分配足够的计算能力，高效地验证这些词元。

基于上述想法，我们提出 LTD（学习起草），这是一种新方法，通过将吞吐量作为直接优化目标，协同优化两个阶段。具体而言，我们将草稿模型与目标模型之间的交互构建为一个强化学习（RL）环境。如图 1 所示，我们训练两种不同的协同自适应策略：一种深度策略，通过确定草稿树的深度来控制草稿成本；另一种规模策略，通过为目标模型选择最佳候选词元数量来管理验证成本。关键在于，这些策略以每个草稿与验证周期的吞吐量作为直接奖励信号进行训练。此外，我们迭代训练这两种策略，使其能够针对彼此不断演进的策略进行协同适应。这样，它们便能学会如何自适应地管理接受长度与时间成本之间的权衡，并动态协调两个阶段，从而最大限度地提高推理效率。

我们以最先进的 Eagle3 框架为基础，通过训练策略，以动态且具备上下文感知能力的决策过程取代其静态启发式方法。正如我们在五种不同的大语言模型和四项基准测试中开展的大量评估所展示的那样，我们的方法取得了显著的性能提升。在贪心解码过程中，与经过高度优化的 Eagle3 基线相比，我们的方法显著提升了吞吐量：在 Qwen3-32B 上提升 36.4%，在 Deepseek-Distilled-Llama-8B 上提升 9.5%，在 Llama-3-8B 上提升 6.5%，在 Vicuna-1.3-13B 上提升 5%，在 Qwen3-14B 上提升 4%。值得注意的是，我们的方法在高温场景下表现稳健，吞吐量提升约 5%，而大多数其他动态推测方法在此类场景中都会失效。我们的贡献总结如下：

• 直接吞吐量优化：我们提出一种时间感知优化方法，将推测解码过程建模为强化学习环境。通过将每个起草与验证周期的吞吐量最大化作为直接奖励，我们的方法学会了在接受长度与时间成本之间进行权衡。

• 协同自适应策略框架：我们不再将草拟阶段和验证阶段视为彼此独立的问题，而是提出一种联合训练的协同自适应策略框架，使两个策略学会动态协同，以适应各自所处的阶段。

• 最先进水平的性能与鲁棒性：与强大的 Eagle3 基线相比，我们的方法在贪心解码中取得了显著的性能提升（最高达 36.4%）。与以往工作相比，该方法在高温采样场景下仍能保持有效性。

2 相关工作

2.1 推测解码

原始推测解码采用链式结构生成候选词元（Stern 等人，2018；Chen 等人，2023；Leviathan 等人，2023）。在这些早期链式方法的基础上，SpecInfer（Miao 等人，2023）提出了基于树的起草架构。通过引入树注意力机制，该方法能够并行验证多个候选序列，形成了一种高效范式，此后得到广泛采用（Cai 等人，2024；Li 等人，2024）。后续研究主要聚焦于通过若干关键策略改进草稿模型，例如利用知识蒸馏（Liu 等人，2023；Zhou 等人，2023）、采用更有效的特征进行草稿词元预测（Cai 等人，2024；Li 等人，2025a；Ankner 等人，2024；Zhang 等人，2024a；Li 等人，2025b；Du 等人，2024），以及采用自推测，即将目标模型的组件重新用于草拟（Zhang 等人，2023；Elhoushi 等人，2024；Hooper 等人，2025）。

2.2 推测解码的自适应调整

尽管基于树的验证显著提升了吞吐量，大多数方法仍为每个草稿与验证周期采用静态调度（Cai 等人，2024；Li 等人，2025a），无法根据输入上下文复杂度的变化进行动态调整。认识到这一局限后，近期研究日益聚焦于动态方法，即根据生成上下文调整推测策略。这些工作大致可分为两类：通过控制生成长度或草稿深度动态调整草稿时间（Brown 等人，2024；Mamou 等人，2024；Zhang 等人，2024c;b；Huang 等人，2025a；Gautam 等人，2025），或通过为目标模型选择候选词元动态调整验证时间（Zhong 等人，2024；Wang 等人，2025；Huo 等人，2025）。这些方法通常利用词元概率（Brown 等人，2024）、熵（Mamou 等人，2024）或隐藏状态（Zhang 等人，2024b；Huang 等人，2024）等上下文信息来调整推测解码，以最大化期望接受长度。后续研究还探索了其他优化方向。一些方法着重调整草稿树的宽度（Qin 等人，2025；Xiong 等人，2025）。另一些工作则探索自推测解码的动态方法（Zarch 等人，2025）。还有研究致力于改进整体服务级别目标（Huang 等人，2025b），或优化草拟与验证阶段之间的并行执行（Liu 等人，2025）。

3 模型概述

3.1 初步研究：时间成本与接受长度

我们在 Llama 和 Vicuna 模型上进行了初步实验，以验证我们的想法：在草稿与验证推理中，动态调整关键参数（草稿深度和验证规模）对于通过最大化吞吐量来实现最佳加速是必要的。这里，每个草稿与验证周期的吞吐量（λ_c）定义为已接受词元数（L_A）除以草稿与验证周期的总实际运行时间（T_{total}）：

$$\lambda_c = L_A / T_{\text{total}} \tag{1}$$

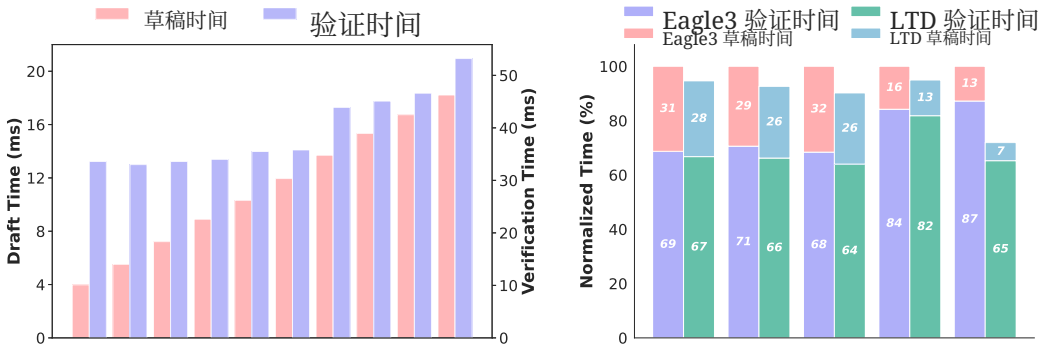


图 2：草稿时间与验证时间分析。左：Vicuna-13B 在基准测试上的平均草稿延迟和验证延迟随草稿深度 D 和验证规模 V 的变化。右：Eagle3 和 LTD 在基准测试上的总推理时间构成，其中 Eagle3 的总时间归一化为 100%，作为基线。

T_{total} 可分解为草稿时间 T_{draft}和验证时间 T_{verify}：

$$T_{\text{total}} = T_{\text{draft}} + T_{\text{verify}} \tag{2}$$

草稿时间 T_{draft} 主要由草稿深度（D）决定，即草稿模型依次执行的前向传播次数（也就是生成候选词元的步数）。较大的 D 会生成更多候选词元，但所需时间也会线性增加，如图 2 （左）中的红色柱状图所示。相反，验证时间主要由验证规模（ V ）决定，即大型目标模型在一次前向传播中处理的候选词元总数。由于模型的计算开销以及输入序列较长，随着 V增大，验证时间会呈阶梯式增长，如图 2（左）中的蓝色柱状图所示。 V可能很大（例如， $D = 10$时最多可达 1000 个词元），因此是影响验证时间的关键因素。采用我们的 LTD策略后，验证时间可显著缩短，如图 2（右）所示。

除了时间延迟外，图 5 还表明，在给定草稿模型和目标模型的情况下，草稿深度和验证规模会显著影响平均接受长度（见附录）。增加草稿深度（D）和验证规模（V）可以生成更多候选词元，从而有望提高每周期接受长度（L_A，即吞吐量公式中的分子）。然而，增加 D 和 V不可避免地会导致草稿时间和验证时间同时增加（即分母 T_{total}增大）。因此，我们的目标并非盲目地最大化 D 和 V，而是找到一种最佳平衡，以最大化总体吞吐量，其定义为 $\lambda_c = L_A/T_{\text{total}}$。

3.2 草稿与验证周期

我们的工作建立在当前最先进的基于树的方法之上，即 Eagle3（Li 等人，2025a）。本节将正式定义单个草稿与验证周期的运作机制，以及我们的策略在引导这一过程中的作用。如图 1 所示，草稿与验证周期可分为三个部分：草稿树生成、候选词元选择和草稿树验证。

草稿树生成 草拟过程始于目标模型输出紧接最后一个已接受词元之后的那个词元；该词元将作为新草稿树的根。该树通过由两个关键超参数控制的迭代式、类似束搜索的扩展过程来构建：束宽 W与草稿深度 D。构建过程从根词元的初始扩展开始。在这第一步中，草稿模型执行一次前向传播，生成后续词元上的概率分布，并选出概率最高的前W 个候选词元，构成树的第一层叶节点。此后，该树会进行 $D-2$ 轮后续的迭代生长。在每一轮中，草稿模型以上一层的 W个叶节点作为输入。对于这 W 个节点中的每一个，它都会生成 W 个新的子

候选，从而形成一个包含 W^2 个新节点的合并池。随后对该池执行束搜索选择，从中选出预测累积路径概率最高的前W 个候选，作为树的新叶节点，为下一轮扩展做好准备。Eagle3的草稿深度默认设为8，而在LTD中，我们的深度策略会在草稿模型每次前向传播之后决定是继续或停止。一旦深度策略决定停止扩展，整棵草稿树将交给规模策略进行候选词元选择。

候选词元选择 在草稿深度为 D 时，最终草稿树共包含$V_{all} = 1 + W + (D - 2)$ 个候选词元，每个词元都对应一个预测概率。生成草稿树后，选取其中整体概率最高的 V 个候选词元进行验证。Eagle3 使用固定的验证规模 V，而 LTD 则利用规模策略，根据生成的草稿树的属性动态确定该值。随后，将这些 V 个选定的词元展平为一个序列，并连同相应的树注意力掩码一起传递给目标模型。

草稿树验证 目标模型通过一次前向传播并行验证这些 V 个词元。它会接受序列中从起始位置到自身预测与候选词元首次不匹配处之前的所有词元。在首次出错的位置，目标模型会丢弃剩余的候选词元，计算正确的词元，并将其追加到已接受令牌序列中。这个新确认的词元随后成为下一个草稿与验证周期中草稿树的根。最后，将一个周期内已接受词元的数量 L_A （称为接受长度）除以草稿时间与验证时间之和，得到该周期的吞吐量。该吞吐量用作奖励信号，以优化我们的深度策略和规模策略。

4 方法

我们在强化学习（RL）框架下对草稿模型与目标模型之间的交互进行建模，并采用两种协同自适应策略来应对接受长度与延迟之间固有的权衡：一种是**深度策略**，π_D，用于确定草稿深度；另一种是**规模策略**，π_V，用于确定验证规模。深度策略的动作空间是二元的，即 $\mathcal{A}_{\pi_D} = \{0, 1\}$，用于决定每次前向传播后是否继续草拟；而规模策略的动作空间 $\mathcal{A}_{\pi_V} \in [20, 240]$ 是一个离散的验证规模范围。为指导决策，这些策略在每个时间步 t观测一个状态向量 $s_t \in \mathcal{S}$，其定义如下

$$s_t = [D, L, P_1; \ldots; P_{V_{all}}] \tag{3}$$

其中，D 是当前草稿深度，L 是上下文长度，而 P_1、.....、$P_{V_{\text{all}}}$ 是草稿树中所有词元的预测概率。该状态表示旨在提供丰富的信息，因为词元概率是接受长度的强预测指标（Li 等人，2025a），而草稿深度和序列长度则是决定延迟的重要因素（图 2）。为了优化决策延迟，规模策略 π_V 观测完整状态，而深度策略 π_D 仅使用一个最小子集：草稿树最后一层词元的概率，以及 D 和L。两种策略均由轻量级多层感知机（MLP）实现，以确保决策过程引入的计算开销可以忽略不计。关于这两种策略的更多细节，请参阅附录 A.2。

这些策略使用近端策略优化（PPO）算法（Schul- man 等，2017）进行联合优化，以最大化奖励信号 R_t，该信号表示每个周期的吞吐量：

$$R_t = L_A/(T_{draft} + T_{verify}) \tag{4}$$

该奖励函数直接激励策略掌握这一复杂权衡：生成更多可能正确的词元，同时控制由此产生的时间成本。我们通过以下目标函数更新策略网络 π_θ：

$$J(\theta) = \hat{\mathbb{E}}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t,\ \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right], \tag{5}$$

ˆ 其中，$\pi_{\theta_{\text{old}}}$ 是行为策略，A_t 是优势估计（Schulman等人，2017）。每次优化步骤完成后，旧策略参数都会更新为新的参数（$\theta_{\text{old}} \leftarrow \theta$），

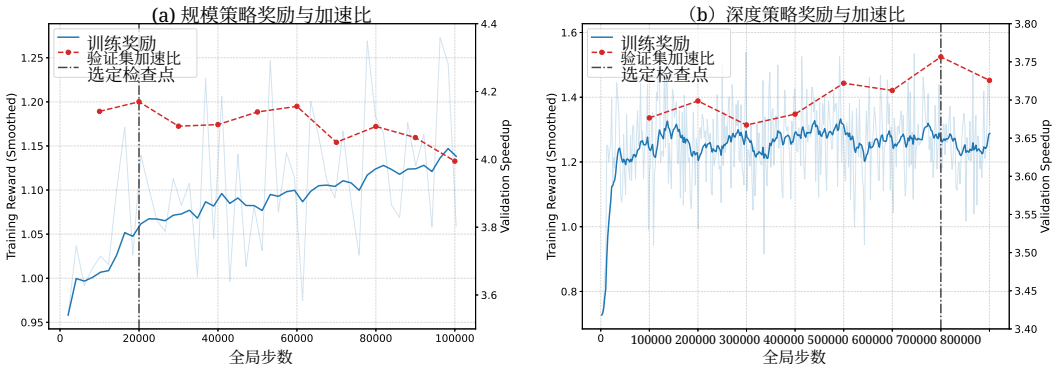


图 3：我们报告了 Vicuna-13B 模型的规模策略和深度策略的训练奖励曲线，以及验证集上加速比的变化。

然后重复数据收集和优化过程。为确保稳定收敛，我们采用两阶段训练流程。第一阶段分别训练初始深度策略和大小策略。第二阶段则以这些初始策略为起点进行迭代训练，使它们能够协同适应，并收敛到协调一致的策略。

初始策略训练。我们使用 HumanEval 数据集（Chen 等人，2021）进行训练，因为其中多样且复杂的代码片段为策略提供了丰富的环境，使其能够制定各种自适应策略。在这一初始阶段，我们分别训练每个策略，同时让另一个策略采用固定的基于启发式的策略。具体而言，规模策略 π_V 使用从范围 $[1, 12]$ 中随机采样的草稿深度进行 100k 步训练。相反，深度策略 π_D 使用恒定的验证规模 60 进行 1M 步训练，与 Eagle3 基线采用的启发式方法一致。这一初始训练构建了稳健且独立的策略，为后续的协同适应阶段奠定了稳定基础。我们在图 3 中展示了 Vicuna-13B 策略在训练期间的奖励变化，以及验证集上观察到的加速比。我们选择了在验证集上性能最高的检查点作为后续训练的基础。如图所示，训练步数足以确保模型收敛。其他模型的训练曲线与 Vicuna-13B 的曲线表现出相似的模式。

策略协同适应的迭代优化。尽管这些策略是独立训练的，但要在推理时取得最佳性能，关键在于它们协同运作。为应对这一协同适应挑战，我们提出了一种迭代优化流程。该流程交替优化各策略，同时固定另一策略。首先，我们冻结规模策略（π_V）的参数，并根据 π_V 提出的动态验证规模优化深度策略（π_D）。随后，冻结新更新的深度策略（π_D），并依据 π_D 确定的动态草稿深度优化规模策略（π_V）。这种交替优化周期使两种策略能够逐步适应彼此不断演进的策略。实验发现，仅需进行两轮迭代，就能得到高度协同的策略并显著提升性能。

5 实验

5.1 设置

模型 我们使用最先进的开源聊天与推理模型开展实验，包括 Llama-3.1-8B-Instruct（Dubey 等人，2024）、Vicuna-13B-v1.3（Chiang 等人，2023）、DeepSeek-R1-Distill-LLaMA 8B（DeepSeek-AI，2025）、Qwen3-14B（Team，2025a）和 Qwen3- 32B（Team，2025a）。Qwen3 系列的 Eagle3 模型来自 AngelSlim（Contributors，2025），因为当时尚未发布官方实现。对于所有其他模型，我们使用了 Eagle3 的官方公开版本（Li 等人，2025a）。

任务 遵循 EAGLE（Li 等人，2024）和 Spec-Bench（Xia 等人，2024），我们在四项常见任务上进行评估，并对所有任务使用相同的策略权重。对于多轮对话、数学推理、指令遵循和问答，我们分别选用 MT-bench（Zhenget al.，2023）、GSM8K（Cobbe et al.，2021）、Alpaca（Taori et al.，2023）和 Natural Questions（Kwiatkowski et al.，2019）数据集。

指标 推测解码不会改变目标模型的输出分布。因此，我们不评估生成质量，而是采用以下常用指标（Li 等人，2025a）来评估加速性能：

- **加速比：** 相对于原始自回归解码的实际测试加速比。
- **平均接受长度 τ：** 在基准数据集上，每个草稿生成与验证周期平均生成的词元数。

实现细节 我们使用 Stable-Baselines3 库（Raffin 等人，2021）实现近端策略优化（PPO）算法，并在 HumanEval 数据集（Chen 等人，2021）上训练策略。大小策略和深度策略分别训练 100k 和 1M 步。关于训练步数选择的进一步分析见附录 C.1。PPO 的配置为：轨迹缓冲区包含 2048 步，小批量大小为 256。由于这两种策略十分轻量，强化学习训练阶段效率很高，详细信息见附录 A.7。完整的超参数列表见附录 A.3。

比较 我们采用原始自回归解码作为基线，并以此作为加速比的基准（1.00x）。我们将所提出的方法应用于当前最先进的推测解码方法 Eagle3，并与近期具有代表性的动态深度方法进行比较，包括 DDD（Brown 等人，2024）、SVIP（Zhang 等人，2024c）、Gammatune（Gautam 等人，2025）、Disco（Mamou 等人，2024）和 SpecDec++ （Huang 等人，2025a），以及动态规模方法 C2T（Huo 等人，2025）。动态深度方法的默认参数设置为验证规模 60、最大深度 12；动态规模方法的默认深度设置为 8。对于 Eagle3 基线，我们将 8B 和 13B 原始大语言模型的验证规模分别设为 60 和 50，草稿深度设为 8，并按照原始实现（Li 等人，2025a）的设置，在扩展阶段选择 10 个节点。我们还采用网格搜索（GS）为 Eagle3 寻找最优超参数，以建立强有力的基线。更多细节请参见附录 A.1。

5.2 有效性

我们的方法性能详见表 1 和表5。实验结果表明，在所评估的模型、数据集和采样温度下，我们提出的方法始终优于所有竞争方法。在各种配置下观察到的稳定加速比，凸显了我们框架的强大能力。

与默认的 Eagle3 配置相比，我们的方法在 Llama-3 上平均可将推理速度提高 6.5%，在 Vicuna 上平均提高 5.1%。对于 Deepseek-R1-Distll-LLaMA 8B 这类长思考模型，我们的方法可实现 10% 的加速，这对于生成时间较长的此类模型相当有用。对于 Qwen3 系列，由于没有提供默认参数，我们将网格搜索的结果作为基线。与 14B 模型相比，我们的方法在 32B 模型上取得了显著更大的提升，证明了该方法在更大规模下的可靠性。值得注意的是，尽管我们的训练仅采用贪心解码，但如表5所示，在温度为 1.0 时，我们的方法仍保持最高的平均加速比。相比之下，其他会调整草稿深度或验证规模的动态方法虽然在贪心解码时效果良好，但在较高温度下性能会显著下降。这凸显了我们方法的强大鲁棒性。

我们的方法不一定能带来最长的接受长度；其 τ 通常短于网格搜索优化基线的接受长度。尽管如此，它仍实现了最高的总体加速比。这一结果揭示了我们方法的核心机制：强化学习框架使策略学会直接针对解码效率进行优化。对于那些虽然可能正确、却会产生不成比例的高时间成本的候选词元，该方法会策略性地放弃。通过直接优化吞吐量这一奖励信号，我们的方法能够在接受长度与时间成本之间进行自适应平衡，从而实现更优的加速比。

表 1：贪心解码期间不同方法的加速比和平均接受长度 τ 。*L31 8B*表示 Llama-3.1-8B-Instruct，*V 13B* 表示 Vicuna-13B-v1.3，*Dpsk 8B*表示 DeepSeek-R1-Distill-LLaMA 8B。对于 Qwen3-14B 和 Qwen3-32B 模型，其对应的 Eagle3 模型没有默认设置。因此，此处仅展示网格搜索的结果。GT、Spec++ 和 Gs 分别代表 Gammatune、SpecDec++ 和网格搜索。温度=1 的结果见表5。

		MT-bench		Gsm8k		Alpaca		Qa		Mean	
模型	方法	加速比	τ	加速比	τ	加速比	τ	加速比	τ	加速比	τ
					温度=0						
L31 8B	Eagle3	3.70	6.39	3.58	6.44	4.26	6.93	3.17	5.35	3.68	6.28
	Eagle3+DDD	3.67	5.41	3.55	5.48	3.97	5.58	3.07	4.21	3.57	5.17
	Eagle3+SVIP	3.65	6.21	3.47	6.21	4.04	6.73	3.12	4.78	3.57	5.98
	Eagle3+GT	3.79	5.86	3.64	5.77	4.37	6.49	3.38	4.92	3.80	5.76
	Eagle3+C2T	3.80	6.39	3.57	6.44	4.19	6.93	3.25	5.36	3.70	6.28
	Eagle3+Disco	3.70	6.11	3.56	6.18	4.14	6.51	3.22	5.05	3.66	5.96
	EAGLE3+规范++	3.82	6.09	3.65	6.08	4.29	6.70	3.28	5.23	3.76	6.03
	Eagle3+GS	3.83	**6.75**	3.78	**6.91**	4.38	**7.42**	3.33	**5.67**	3.83	**6.69**
	Eagle3+LTD	**3.96**	6.74	**3.79**	6.73	**4.53**	7.31	**3.41**	5.44	**3.92**	6.56
V 13B	Eagle3	4.48	7.22	4.36	6.58	4.16	6.47	3.43	5.12	4.11	6.35
	Eagle3+DDD	4.15	6.69	4.25	6.22	3.98	5.50	3.22	4.34	3.90	5.69
	Eagle3+SVIP	4.02	7.74	3.86	6.79	3.77	6.53	3.02	4.96	3.67	6.51
	Eagle3+GT	4.15	6.85	4.07	6.00	3.94	6.09	3.35	4.81	3.88	5.94
	Eagle3+C2T	4.33	7.22	3.97	6.58	3.93	6.47	3.23	5.12	3.87	6.35
	Eagle3+Disco	4.48	7.08	4.13	6.49	4.02	6.29	3.31	5.05	3.99	6.23
	Eagle3+Spec++	4.25	6.78	4.09	6.16	4.00	6.27	3.21	5.03	3.89	6.06
	Eagle3+GS	4.53	7.99	4.47	7.27	4.38	**7.16**	3.51	**5.48**	4.22	6.98
	Eagle3+LTD	**4.64**	**8.24**	**4.59**	**7.38**	**4.40**	7.01	**3.66**	5.37	**4.32**	**7.00**
Dpsk 8B	Eagle3	3.67	5.82	4.73	7.39	3.65	5.63	3.14	5.02	3.80	5.53
	Eagle3+GS	3.69	**6.42**	4.83	8.37	3.54	**6.24**	3.20	**5.55**	3.82	6.08
	EAGLE3+LTD	**4.05**	6.32	**4.98**	**8.53**	**3.98**	6.00	**3.62**	5.37	**4.16**	**6.08**
Qwen3 14B	EAGLE3+GS	2.34	**3.40**	2.75	**3.95**	2.15	**2.90**	1.86	**2.65**	2.28	**3.23**
	EAGLE3+LTD	**2.44**	3.34	**2.82**	3.92	**2.24**	2.87	**1.98**	2.62	**2.37**	3.19
Qwen3 32B	EAGLE3+GS	1.76	**3.30**	2.26	**4.10**	1.75	**2.96**	1.45	**2.67**	1.81	**3.26**
	EAGLE3+LTD	**2.42**	3.17	**3.02**	3.90	**2.38**	2.85	**2.06**	2.60	**2.47**	3.13

为进一步验证泛化能力，我们在大规模多任务语言理解上评估了我们的方法。该基准涵盖不同领域的 57 个子任务（结果详见附录 B 和图 7）。在 57 个子任务中的 54 个上，我们的方法优于 Eagle3，平均加速比提升了 5%。表现最佳的前 10 个类别分布广泛，不仅包括数学、逻辑，还包括人类衰老和政府政治。这些结果有力地证明了我们方法的跨领域泛化能力。

5.3 草稿时间和验证时间

为细致分析性能提升的构成，我们在图 2 右侧展示了总推理时间的相对组成，将其分解为草稿时间和验证时间。结果表明，我们的方法通过改进草拟和验证两方面实现了加速。动态深度策略会在置信度较低的场景中减少推测，从而节省计算，因此草稿时间普遍缩短。值得注意的是，我们的方法还能缩短验证时间，这意味着令牌验证负载更低。矛盾的是，如表 1 所示，对于 Llama、Vicuna 和 Dpsk 模型，更低的开销对应着更长的平均接受长度。这一结果表明，我们方法的优势不仅在于缩小草稿树的大小，还在于有策略地将候选词元分配到最有可能被接受的上下文中，从而验证了我们方法的有效性。对于 Qwen 系列模型，尽管我们的方法的接受长度低于基线，但它显著缩短了验证时间，尤其是对于 Qwen3-32B 模型，因此实现了大幅提高的加速比。

6 消融研究

在本节中，我们通过一系列分析来验证 LTD 方法的有效性。首先，我们在第 6.1 节展示迭代优化过程的有效性。接下来

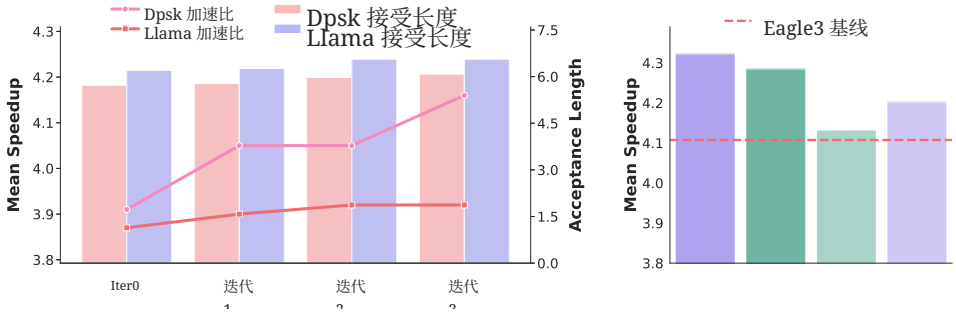


图4：左：迭代训练策略在 Llama3-8B 和 Dpsk-8B 模型上的效果。右：Vicuna-13B 模型中各组件的贡献。

我们在第 6.2 节开展消融研究，以单独评估深度和规模策略的贡献。为了论证强化学习设计的合理性，我们还在第 6.3 节测试了其他奖励信号，并在附录 A.4 中考察了其他观测空间。综合来看，这些实验凸显了 LTD 方法中各个组成部分的有效性。

6.1 迭代优化分析

我们研究了迭代优化策略在 Llama3-8B 和 Dpsk-8B 模型上的有效性。结果见图4（左），完整数据可参见附录A.5中的表 7和表 8。该过程从迭代0（Iter0）开始，采用最初训练得到的策略进行朴素组合。随后，我们按照第4节所述的交替优化计划进行优化。具体而言，我们先优化规模策略（迭代1），再优化深度策略（Iter2），最后再次优化规模策略（Iter3）。每个阶段都会冻结另一策略的参数。结果证实了这种迭代方法的优势。

每一步优化都能逐步提升平均加速比和接受长度，证实策略协同适应能够形成协同效应更强的系统。尽管最显著的提升出现在从 Iter0 到迭代1 的初始阶段，但后续迭代的效果取决于模型。对于 Llama3-8B，性能在 Iter2 后趋于饱和，Iter3 带来的收益微乎其微。相比之下，Dpsk-8B 模型在 Iter3 阶段仍持续取得显著提升。这些发现验证了迭代优化策略的有效性。

6.2 关键组件的影响

为分析我们提出的方法中各关键组件的贡献，我们在图4（右）所示的 Vicuna-13B 模型上进行了全面的消融研究。详细结果见附录A.5中的表 4。我们评估了三个主要变体：1) –规模：移除规模策略，仅依赖深度策略。2) –深度：移除深度策略，仅依赖规模策略。3) –迭代训练：直接组合第4节中的初始词元策略和深度策略，而不采用我们提出的迭代优化。

结果清楚地表明，深度策略对整体加速发挥着重要作用。尽管规模策略相较于基线也能带来加速比，但其影响不那么显著。此外，朴素组合这两种策略会导致性能欠佳。这凸显了我们迭代训练方案的有效性：该方案使策略能够协同适应，从而实现协同加速。总而言之，这些结果验证了两种策略各自的贡献，以及将它们结合起来的协同适应过程所发挥的关键作用。

6.3 奖励信号的有效性

为了验证将吞吐量用作奖励信号的有效性，我们设计并比较了三种不同的奖励函数，仅针对 Vicuna 模型优化规模策略：1）每周期接受长度；2）每周期时间成本；3）每周期吞吐量。表 2 中的结果表明，在所有数据集上，使用吞吐量作为奖励都能获得最高的加速比。

表 2：在 Vicuna-13B 模型上评估奖励信号对规模策略的有效性。

奖励	MT-bench		Gsm8k		Alpaca		Qa		Mean	
	加速比	τ	加速比	τ	加速比	τ	加速比	τ	加速比	τ
接受长度	3.77	**7.70**	3.81	**7.20**	3.71	**7.05**	3.06	**5.61**	3.59	**6.89**
时间成本	3.89	6.49	3.54	5.80	3.48	5.79	2.87	4.54	3.45	5.66
吞吐量	**4.40**	7.68	**4.43**	7.10	**4.23**	7.00	**3.47**	5.56	**4.13**	6.84

虽然使用接受长度作为奖励可使 τ 达到最大值，但其平均加速比却显著较低。这种差异是因为这种规模策略会为验证提出过多词元，导致验证时间大幅增加，抵消了接受长度增加带来的收益。相反，使用时间成本作为奖励会导致接受长度最低，因为策略会倾向于提出过少的词元，无法从目标模型的并行验证中获益，最终导致整体加速比最低。这些发现证实，吞吐量是一种更优的奖励信号，因为它能在最大化接受长度和最小化时间成本这两个相互竞争的目标之间实现平衡。

7 结论

在本文中，我们介绍了 LTD，这是一种新颖的推测解码方法，通过强化学习直接优化每个草稿与验证周期的吞吐量。通过采用两个协同自适应的策略，动态协调草拟和验证阶段，LTD 学会了管理接受长度与时间成本之间的权衡，从而实现协同加速。在四个基准测试上对五个不同的大语言模型进行的大量实验表明，无论是贪心解码还是高温采样，LTD 始终能够实现最先进的加速效果。这一优异性能凸显了我们方法的有效性和稳健性，使 LTD 成为加速大语言模型推理的有力方案。

可复现性声明

在第4节和附录 A.2 中，我们详细介绍了我们的方法和训练过程；在第 5.1 节和附录 A.3 中，我们给出了完整的训练参数和评估数据集；在附录 A.1 中，我们记录了基线方法所使用的超参数。这些细节共同支持了我们工作的可复现性。

致谢

感谢匿名审稿人对本文提出的宝贵意见。本工作部分得到国家自然科学基金项目（项目编号：92470205 和 62476010）的资助。

参考文献

Sandhini Agarwal、Lama Ahmad、Jason Ai、Sam Altman、Andy Applebaum、Edwin Arbus、Rahul K Arora、Yu Bai、Bowen Baker、Haiming Bao 等。gpt-oss-120b 与 gpt-oss-20b 模型卡片。*arXiv* 预印本 *arXiv:2508.10925*，2025年。

Zachary Ankner、Rishab Parthasarathy、Aniruddha Nrusimha、Christopher Rinard、Jonathan Ragan- Kelley 和 William Brandon。Hydra：用于 Medusa 解码的序列依赖草稿头。*arXiv* 预印本 *arXiv:2402.05109*，2024。

Oscar Brown、Zhengjie Wang、Andrea Do、Nikhil Mathew 和 Cheng Yu。动态深度解码：更快的大语言模型推测解码。*CoRR*，2024。

Tianle Cai、Yuhong Li、Zhengyang Geng、Hongwu Peng、Jason D Lee、Deming Chen 和 Tri Dao。Medusa：采用多个解码头的简易大语言模型推理加速框架。载于第*41*届国际机器学习会议论文集，第 5209–5235 页，2024。

Charlie Chen、Sebastian Borgeaud、Geoffrey Irving、Jean-Baptiste Lespiau、Laurent Sifre 和 John Jumper。通过推测采样加速大语言模型解码。*arXiv* 预印本 *arXiv:2302.01318*，2023年。

Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde De Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman 等。评估在代码上训练的大语言模型。*arXiv* 预印本 *arXiv:2107.03374*，2021年。

Wei-Lin Chiang、Zhuohan Li、Zi Lin、Ying Sheng、Zhanghao Wu、Hao Zhang、Lianmin Zheng、Siyuan Zhuang、Yonghao Zhuang、Joseph E. Gonzalez、Ion Stoica 和 Eric P. Xing。Vicuna：一款以 GPT-4 为基准、质量达到 ChatGPT 90%* 的开源聊天机器人，2023年3月。网址：https: // lmsys.org/blog/2023-03-30-vicuna/。

Karl Cobbe、Vineet Kosaraju、Mohammad Bavarian、Mark Chen、Heewoo Jun、Lukasz Kaiser、Matthias Plappert、Jerry Tworek、Jacob Hilton、Reiichiro Nakano 等。训练验证器以解答数学文字题。*arXiv* 预印本 *arXiv:2110.14168*，2021年。

腾讯 AngelSlim 项目贡献者。AngelSlim，2025年6月。网址 https://github.com/Tencent/AngelSlim。

DeepSeek-AI。Deepseek-r1：通过强化学习激励大语言模型的推理能力，2025年。网址 https://arxiv.org/abs/2501.12948。

Cunxiao Du、Jing Jiang、Xu Yuanchen、Jiawei Wu、Sicheng Yu、Yongqi Li、Shenggui Li、Kai Xu、Liqiang Nie、Zhaopeng Tu等。GLIDE with a CAPE：一种简便加速推测解码的方法。*arXiv*预印本 *arXiv:2402.02082*，2024。

Abhimanyu Dubey、Abhinav Jauhri、Abhinav Pandey、Abhishek Kadian、Ahmad Al-Dahle、Aiesha Letman、Akhil Mathur、Alan Schelten、Amy Yang、Angela Fan等。Llama 3模型群。*arXiv*预印本 *arXiv:2407.21783*，2024。

Mostafa Elhoushi、Akshat Shrivastava、Diana Liskovich、Basil Hosmer、Bram Wasti、Liangzhen Lai、Anas Mahmoud、Bilge Acun、Saurabh Agarwal、Ahmed Roman 等。LayerSkip：实现提前退出推理和自推测解码。*arXiv* 预印本 *arXiv:2404.16710*，2024。

Aayush Gautam、Susav Shrestha 和 Narasimha Reddy。令牌驱动的 gammatune：用于增强推测解码的自适应校准。*arXiv* 预印本 *arXiv:2504.00030*，2025年。

Daya Guo、Dejian Yang、Haowei Zhang、Junxiao Song、Ruoyu Zhang、Runxin Xu、Qihao Zhu、Shirong Ma、Peiyi Wang、Xiao Bi 等。DeepSeek-R1：通过强化学习激发大语言模型的推理能力。*arXiv* 预印本 *arXiv:2501.12948*，2025年。

Dan Hendrycks、Collin Burns、Steven Basart、Andy Zou、MantasMazeika、DawnSong 和 Jacob Steinhardt。《衡量大规模多任务语言理解》。*arXiv* 预印本 *arXiv:2009.03300*，2020年。

Coleman Hooper、Sehoon Kim、Hiva Mohammadzadeh、Hasan Genc、Kurt Keutzer、Amir Gholami 和 Yakun Sophia Shao。SPEED：用于高效解码的推测流水线执行。载于提升大语言模型性能：效能、微调与推理技术，第19–32页。施普林格，2025年。

Shijing Hu、Jingyang Li、Xingyu Xie、Zhihui Lu、Kim-Chuan Toh 和 Pan Zhou。Griffin：通过有效的令牌对齐加速推测解码。*arXiv* 预印本 *arXiv:2502.11018*，2025年。Kaixuan Huang、Xudong Guo 和 Mengdi Wang。Specdec++：通过自适应候选长度提升推测解码性能。*arXiv* 预印本 *arXiv:2405.19715*，2024年。Kaixuan Huang、Xudong Guo 和 Mengdi Wang。Specdec++：通过自适应候选长度提升推测解码性能。*arXiv* 预印本 *arXiv:2405.19715*，2025a。

Kaiyu Huang、Hao Wu、Zhubo Shi、Han Zou、Minchen Yu 和 Qingjiang Shi。《SpecServe：通过自适应推测解码实现高效且具备服务级目标感知的大语言模型服务》。*arXiv* 预印本 *arXiv:2503.05096*，2025b。

Feiye Huo、Jianchao Tan、Kefeng Zhang、Xunliang Cai 和 Shengli Sun。《C2T：推测解码中的一种基于分类器的树构造方法》。*arXiv* 预印本 *arXiv:2502.13652*，2025。Tom Kwiatkowski、Jennimaria Palomaki、Olivia Redfield、Michael Collins、Ankur Parikh、Chris Alberti、Danielle Epstein、Illia Polosukhin、Jacob Devlin、Kenton Lee 等。《Natural Questions：面向问答研究的基准测试》。计算语言学协会汇刊，7：453–466，2019。Yaniv Leviathan、Matan Kalman 和 Yossi Matias。《通过推测解码实现 Transformer 的快速推理》。载于国际机器学习大会，第 19274–19286 页。PMLR，2023。Yuhui Li、Fangyun Wei、Chao Zhang 和 Hongyang Zhang。《Eagle-2：借助动态草稿树加速语言模型推理》。载于*2024*年自然语言处理实证方法会议论文集，第 7421–7432 页，2024。Yuhui Li、Fangyun Wei、Chao Zhang 和 Hongyang Zhang。《Eagle-3：通过训练时测试扩大大语言模型推理加速规模》。*arXiv* 预印本 *arXiv:2503.01840*，2025a。Yuhui Li、Fangyun Wei、Chao Zhang 和 Hongyang Zhang。《Eagle：推测采样需要重新思考特征不确定性》，2025b。网址：https://arxiv.org/abs/2401.15077。Tianyu Liu、Yun Li、Qitan Lv、Kai Liu、Jianchen Zhu、Winston Hu 和 Xiao Sun。《Pearl：采用自适应草稿长度的并行推测解码》。*arXiv* 预印本 *arXiv:2408.11850*，2025。Xiaoxuan Liu、Lanxiang Hu、Peter Bailis、Alvin Cheung、Zhijie Deng、Ion Stoica 和 Hao Zhang。《在线推测解码》。*arXiv* 预印本 *arXiv:2310.07177*，2023。Jonathan Mamou、Oren Pereg、Daniel Korat、Moshe Berchansky、Nadav Timor、Moshe Wasserblat 和 Roy Schwartz。《动态推测前瞻加速大语言模型的推测解码》。*arXiv* 预印本 *arXiv:2405.04304*，2024。Xupeng Miao、Gabriele Oliaro、Zhihao Zhang、Xinhao Cheng、Zeyu Wang、Zhengxin Zhang、Rae Ying Yee Wong、Alan Zhu、Lijie Yang、Xiaoxiang Shi 等。《SpecInfer：通过基于树的推测推理与验证加速生成式大语言模型服务》。*arXiv* 预印本 *arXiv:2305.09781*，2023。Zongyue Qin、Zifan He、Neha Prakriya、Jason Cong 和 Yizhou Sun。《用于高效大语言模型推理的动态宽度推测束解码》。*arXiv* 预印本 *arXiv:2409.16560*，2025。Antonin Raffin、Ashley Hill、Adam Gleave、Anssi Kanervisto、Maximilian Ernestus 和 Noah Dormann。《Stable-Baselines3：可靠的强化学习实现》。机器学习研究期刊，22(268)：1–8，2021。网址：http://jmlr.org/papers/v22/ 20-1364.html。John Schulman、Filip Wolski、Prafulla Dhariwal、Alec Radford 和 Oleg Klimov。《近端策略优化算法》。*arXiv* 预印本 *arXiv:1707.06347*，2017。Mitchell Stern、Noam Shazeer 和 Jakob Uszkoreit。《深度自回归模型的分块并行解码》。神经信息处理系统进展，31，2018。Yang Sui、Yu-Neng Chuang、Guanchu Wang、Jiamu Zhang、Tianyi Zhang、Jiayi Yuan、Hongyi Liu、Andrew Wen、Shaochen Zhong、Hanjie Chen 等。《停止过度思考：大语言模型高效推理综述》。*arXiv* 预印本 *arXiv:2503.16419*，2025。Rohan Taori、Ishaan Gulrajani、Tianyi Zhang、Yann Dubois、Xuechen Li、Carlos Guestrin、Percy Liang 和 Tatsunori B Hashimoto。《Alpaca：一种强大且可复现的指令遵循模型》。斯坦福基础模型研究中心。*https://crfm. stanford. edu/2023/03/13/alpaca. html*，3(6)：7，2023。Qwen 团队。《Qwen3 技术报告》，2025a。网址：https://arxiv.org/abs/2505.09388。Qwen 团队。《QwQ-32B：拥抱强化学习的力量》，2025年3月，2025b。网址：https://qwenlm.github.io/blog/qwq-32b/。

Jikai Wang、Yi Su、Juntao Li、Qingrong Xia、Zi Ye、Xinyu Duan、Zhefeng Wang 和 Min Zhang。Opt-tree：采用自适应草稿树结构的推测解码。*arXiv* 预印本 *arXiv:2406.17276*，2025年。

Heming Xia、Zhe Yang、Qingxiu Dong、Peiyi Wang、Yongqi Li、Tao Ge、Tianyu Liu、Wenjie Li 和 Zhifang Sui。解锁大语言模型推理效率：推测解码综合综述。载于 Lun-Wei Ku、Andre Martins 和 Vivek Srikumar（编），计算语言学协会 *ACL 2024* 论文集，第 7655–7671 页，泰国曼谷，2024 年 8 月。计算语言学协会。doi：10.18653/v1/2024.findings-acl.456。URL：https://aclanthology.org/2024.findings-acl.456/。

Yunfan Xiong、Ruoyu Zhang、Yanzeng Li 和 Lei Zou。Dyspec：利用动态词元树结构实现更快的推测解码。万维网，28(3):36，2025年。

Hossein Entezari Zarch、LeiGao、Chaoyi Jiang 和 Murali Annavaram。Del：用于高效自推测解码的上下文感知动态退出层。*arXiv* 预印本 *arXiv:2504.05598*，2025年。

Jun Zhang、Jue Wang、Huan Li、Lidan Shou、Ke Chen、Gang Chen 和 Sharad Mehrotra。Draft & verify：通过自推测解码实现无损大语言模型加速。*arXiv* 预印本 *arXiv:2309.08168*，2023年。

Lefan Zhang、Xiaodan Wang、Yanhua Huang 和 Ruiwen Xu。学习协调表示以进行推测采样。*arXiv* 预印本 *arXiv:2408.15766*，2024a。

Situo Zhang、Hankun Wang、Da Ma、Zichen Zhu、Lu Chen、Kunyao Lan 和 Kai Yu。Adaeagle：通过显式建模自适应草稿结构优化推测解码。*arXiv* 预印本 *arXiv:2412.18910*，2024b。

Ziyin Zhang、Jiahao Xu、Tian Liang、Xingyu Chen、Zhiwei He、Rui Wang 和 Zhaopeng Tu。草稿模型知道何时停止：一种用于推测解码的自验证长度策略。*arXiv* 预印本 *arXiv:2411.18462*，2024c。

Lianmin Zheng、Wei-Lin Chiang、Ying Sheng、Siyuan Zhuang、Zhanghao Wu、Yonghao Zhuang、Zi Lin、Zhuohan Li、Dacheng Li、Eric Xing 等。使用 MT-bench 和 Chatbot Arena 评审大语言模型评审。神经信息处理系统进展，36:46595–46623，2023年。

Shuzhang Zhong、Zebin Yang、Meng Li、Ruihao Gong、Runsheng Wang 和 Ru Huang。Propd：用于大语言模型并行解码的动态令牌树剪枝与生成。*arXiv* 预印本 *arXiv:2402.13485*，2024。

Yongchao Zhou、Kaifeng Lyu、Ankit Singh Rawat、Aditya Krishna Menon、Afshin Rostamizadeh、Sanjiv Kumar、Jean-Franc¸ois Kagy 和 Rishabh Agarwal。《Distillspec：通过知识蒸馏改进推测解码》。*arXiv* 预印本 *arXiv:2310.08461*，2023年。

A 详细设置

A.1 基线设置

表 3 总结了我们的方法与各种动态草稿树方法之间的主要区别。对于 DDD，我们在第 5、7 和 9 层之后监测累积对数概率，并设定了 -0.3的最小阈值。在 Gammatune 中，γ_{min} 为默认深度的一半，γ_{max} 为默认深度的 1.5 倍，适应率 η 为 0.2。对于 SVIP，我们遵循原始论文，采用 0.15 的熵因子和 0.3 的最小熵。对于 DDD 和 SVIP，我们直接采用原作者提供的超参数；而对于原始论文未指定超参数的方法，我们则通过实验自行确定。

对于基于前馈神经网络的方法，我们使用默认 Eagle3 生成的 HumanEval 样本，提取隐藏状态、词元分数、熵，以及一个指示草稿词元是否被目标模型接受的二元标记。训练时正样本和负样本数量相同。C2T 使用 64 的隐藏层大小训练 50 轮，而 Disco 使用 128 的隐藏层大小训练相同轮数；SpecDec++ 使用隐藏层大小为 1024 的残差网络训练 5 轮。所有模型均使用 AdamW（$\text{lr} = 1\text{e}{-4}$）训练。有趣的是，我们发现，将熵纳入观测空间有时会降低加速比。这是因为频繁计算熵所产生的开销与前馈神经网络决策准确率之间存在权衡。根据验证实验，我们将熵用作 Disco 的观测特征，而在 C2T 中则省略了熵。
我们还在 HumanEval 基准测试上进行了网格搜索，以确定 Eagle3 模型的最佳超参数，将其作为强基线。搜索空间包括深度为 4 到 12 的草稿树，以及大小为 40 到 240 的验证规模。随后，我们选取了性能最佳的配置用于后续实验。

A.2 LTD

A.2.1 规模策略

规模策略决定验证规模 V——即从推测树中传递给目标模型的候选词元数量。其目标是评估草稿树的整体质量，并选择一个 V 值，以最优方式平衡验证的计算成本与可能接受的词元数量。

- **架构：** 策略网络和价值网络均采用两层前馈网络（FFN）实现，隐藏层维度为 [1024, 256]。

- **训练：** 为增强模型对不同草稿树结构的适应能力，策略通过多个回合进行训练，其中生成深度采用随机化设置，并从 1 到 12 的整数中均匀采样。

- **观测空间：** 提供给智能体的状态表示旨在涵盖草稿阶段的关键结果。它是一个固定大小的向量，主要由生成树中所有候选词元的对数概率分数组成。对于深度为D的树，其中包含最多 $1 + W + (D - 2)$ 个 $\times W^2$ 词元的分数，并通过填充确保向量长度一致。随后，将此分数向量与表示最终草稿深度（D）和输入序列总长度（L）的标量特征拼接起来。

- **动作空间：** 智能体的动作空间是离散的，包含 12 个动作，这些动作线性映射到范围为 20 至 240 的验证规模 V。

A.2.2 深度策略

深度策略按顺序执行，在每个深度层级决定是继续扩展草稿树（CONTINUE），还是终止草稿阶段并进入验证阶段（STOP）。鉴于该策略会在单个生成步骤中多次执行，其推理延迟是影响性能的关键因素。

- **架构：** 为尽量降低延迟，我们采用了轻量级架构。我们发现，单层前馈网络（隐藏层维度为 1024）能够在预测性能与推理速度之间实现有效平衡。

表 3：不同动态草稿树方法的比较

方法名称	预测策略	观测空间	动态大小或深度	训练方法	目标
DDD（Brown 等人，2024）	启发式方法	概率	深度	-	L_A
SVIP（Zhang 等人，2024c）	启发式方法	熵	深度	-	L_A
Gammatune（Gautam 等人，2025年）	启发式方法	令牌接受率	深度	-	L_A
Disco（Mamou 等人，2024）	FFN	概率、熵、草稿深度	深度	FT	L_A
SpecDec++（Huang 等人，2025a）	FFN	隐藏状态	深度	FT	L_A
C2T（Huo 等人，2025年）	FFN	概率、熵、草稿深度	Size	FT	L_A
LTD	FFN	概率、总长度、草稿深度	Both	RL	吞吐量

• **观测空间：** 在这一序贯决策过程中，状态会在每个深度层级更新。观测向量主要由当前扩展前沿生成的候选词元的对数概率构成。对于宽度为W的束搜索，这包括 W^2 个候选分数。随后，将该概率向量与两个标量特征拼接：当前草稿深度（D）和输入序列的总长度（L）。

• **训练策略：** 为了建立稳定的训练课程，我们首先使用固定的 $V = 60$ 词元验证规模训练了一个初始深度策略，这与 Eagle3 基线所采用的启发式方法一致。

• **动作与奖励：** 智能体在每一步都会输出一个二元动作。只有在选择停止动作时，才会根据该回合的最终吞吐量发放数值奖励。中间的继续动作不会带来奖励。为避免策略可能偏向过早终止，我们采用较高的折扣因子（$\gamma = 0.999$），从而鼓励探索更深的草稿树，以期获得更多奖励。

表 4：在 Vicuna-13B 上评估的 LTD 各组成部分的贡献。

	MT-bench		Gsm8k		Alpaca		QA		Mean	
方法	**加速比**	τ	**加速比**	τ	**加速比**	τ	**加速比**	τ	**加速比**	τ
Eagle3	4.48	7.22	4.36	6.58	4.16	6.47	3.43	5.12	4.11	6.35
-深度	4.40	7.68	4.43	7.10	4.23	7.00	3.47	**5.56**	4.13	6.84
-大小	**4.77**	7.75	4.46	6.82	4.36	6.59	3.55	5.12	4.29	6.57
-迭代训练	4.54	**8.29**	4.48	**7.41**	4.28	**7.08**	3.51	5.45	4.20	**7.06**
LTD	4.64	8.24	**4.59**	7.38	**4.40**	7.01	**3.66**	5.37	**4.32**	7.00

A.3 LTD 训练详情

策略（演员）和价值（评论家）函数均使用多层感知机（MLP）实现。

规模策略（π_V）规模部分的策略网络和价值网络采用相同的架构，均由两个隐藏层组成，分别包含 1024 个和 256 个单元，并使用 ReLU 激活函数进行激活。该策略使用标准的折扣因子 $\gamma = 0.9$进行训练。

深度策略（π_D）为最大限度地降低决策延迟，深度策略采用了更轻量的架构，仅包含一个由 1024 个单元组成的隐藏层，而价值网络则与规模部分保持一致。其主要区别在于使用了较高的折扣因子（$\gamma = 0.999$）。选择该数值是为了减轻策略倾向于过早终止的潜在偏差，从而通过优先考虑长期奖励，鼓励探索更深的草稿树。

共享参数 对于两种策略，PPO 代理损失在每个更新周期中优化 20 轮次，并采用 0.01 的熵系数来促进探索。我们使用 1e-3 的学习率，该学习率在训练步数最初 1% 的预热阶段后线性衰减至零。其余所有超参数均保持 Stable-Baselines3 库提供的默认值。

表5：不同方法在温度=1下的加速比和平均接受长度 τ。*L31 8B* 代表 Llama-3.1-8B-Instruct，*V 13B* 代表 Vicuna-13B-v1.3，*Dpsk 8B* 代表 DeepSeek-R1-Distill-LLaMA 8B。对于 Qwen3-14B 和 Qwen3-32B 模型，其对应的 Eagle3 模型没有默认设置。因此，此处仅展示网格搜索的结果。GT、Spec++ 和 Gs 分别代表 Gammatune、SpecDec++ 和网格搜索。

		MT-bench		Gsm8k		Alpaca		Qa		Mean	
模型	**方法**	**加速比**	τ	**加速比**	τ	**加速比**	τ	**加速比**	τ	**加速比**	τ
					温度=1						
L31 8B	Eagle3	2.85	4.11	**3.43**	5.24	3.38	5.47	2.27	3.06	2.98	4.47
	Eagle3+DDD	**2.89**	3.40	3.35	4.09	3.38	4.59	2.36	2.95	3.00	3.76
	Eagle3+SVIP	2.74	3.41	3.21	4.92	3.19	4.95	2.31	3.20	2.86	4.12
	Eagle3+GT	2.75	3.54	3.28	4.27	**3.51**	5.12	**2.46**	3.24	3.00	4.04
	Eagle3+C2T	2.78	**4.11**	3.28	5.24	3.24	5.47	2.27	3.06	2.89	4.47
	Eagle3+Disco	2.82	3.82	3.26	5.08	3.30	5.18	2.20	2.88	2.90	4.24
	Eagle3+Spec++	2.92	4.20	3.33	4.74	3.42	5.51	2.35	3.28	3.01	4.43
	Eagle3+GS	2.79	4.08	3.36	**5.50**	3.46	**5.79**	2.31	**3.40**	2.98	**4.69**
	Eagle3+LTD	2.85	4.02	3.35	5.09	3.43	5.68	2.39	3.26	**3.01**	4.51
V 13B	Eagle3	3.60	5.55	3.68	5.64	3.40	5.49	3.08	4.64	3.44	5.33
	Eagle3+DDD	3.40	5.05	3.58	5.46	3.43	5.00	2.96	4.11	3.34	4.91
	Eagle3+Svip	3.18	5.73	3.26	5.78	3.21	5.39	2.61	4.44	3.07	5.34
	Eagle3+GT	3.43	5.36	3.32	5.26	3.39	4.95	2.99	4.44	3.28	5.00
	Eagle3+C2T	3.38	5.66	3.41	5.63	3.27	5.54	2.83	4.65	3.22	5.37
	Eagle3+Disco	3.48	5.55	3.67	5.74	3.36	5.51	2.88	4.59	3.35	5.35
	Eagle3+Spec++	3.61	5.58	3.64	5.63	3.34	5.39	3.02	4.55	3.40	5.29
	Eagle3+GS	3.58	6.30	**3.81**	**6.39**	3.61	6.01	3.00	4.93	3.50	5.91
	Eagle3+LTD	**3.82**	**6.40**	3.75	6.38	**3.65**	**6.20**	**3.26**	**4.97**	**3.62**	**5.99**
Dpsk 8B	Eagle3	3.07	4.74	3.97	6.89	2.79	4.53	2.44	4.06	3.07	4.66
	Eagle3+GS	3.05	5.19	**4.14**	7.64	2.76	**4.83**	2.33	**4.44**	3.07	5.03
	Eagle3+LTD	**3.18**	**5.20**	4.02	**7.95**	**2.83**	4.76	**2.57**	4.39	**3.15**	**5.09**
Qwen3 14B	EAGLE3+GS	2.05	**3.13**	2.62	**3.83**	2.04	**2.78**	1.68	**2.57**	2.10	**3.08**
	EAGLE3+LTD	**2.16**	3.11	**2.73**	3.85	**2.11**	2.76	**1.82**	2.55	**2.21**	3.07
Qwen3 32B	EAGLE3+GS	1.56	**2.89**	2.10	**3.95**	1.63	**2.73**	1.33	**2.46**	1.66	**3.01**
	EAGLE3+LTD	**2.17**	2.78	**2.83**	3.76	**2.16**	2.59	**1.88**	2.41	**2.26**	2.89

表6：在 Vicuna-13B 上评估规模策略的观测空间消融研究。

奖励	MT-bench		Gsm8k		Alpaca		Qa		Mean	
	加速比	τ	**加速比**	τ	**加速比**	τ	**加速比**	τ	**加速比**	τ
P+L	4.35	7.68	4.30	7.10	4.18	7.00	3.35	5.55	4.05	6.83
P+D	4.31	7.68	4.35	7.10	4.22	7.00	3.41	5.56	4.07	6.84
P+D+L（LTD）	**4.40**	7.68	**4.43**	7.10	4.23	7.00	**3.47**	5.56	**4.13**	6.84
P+D+L+H	4.40	7.53	4.32	6.97	4.22	6.94	3.44	5.51	4.10	6.74
P+D+L+E	4.36	7.68	4.36	7.10	**4.28**	7.00	3.43	5.56	4.11	6.84

A.4 观测状态消融研究

我们进行了消融研究，以确定策略的最佳观测状态构成。以往的方法会将预测概率（P）、隐藏状态（H）、熵（E）、输入长度（L）和草稿深度（D）等变量纳入状态表示，但只有当由此带来的准确率提升超过相应的延迟成本时，纳入这些计算成本高昂的特征才有意义。

表6中的结果表明，我们提出的观测空间配置在这四个基准测试中实现了最高的平均加速比。值得注意的是，我们发现许多竞争方法的接受长度相近。这表明，尽管它们提出的验证规模足以覆盖真实词元序列，但往往需要付出验证时间过长的代价。我们的方法之所以更优，源于两大优势。首先，与依赖熵或完整隐藏状态等计算密集型输入的方法相比，我们的方法在构建观测状态时开销显著更低。其次，与省略草稿深度或总长度信息的基线相比，我们的模型学会了提出更

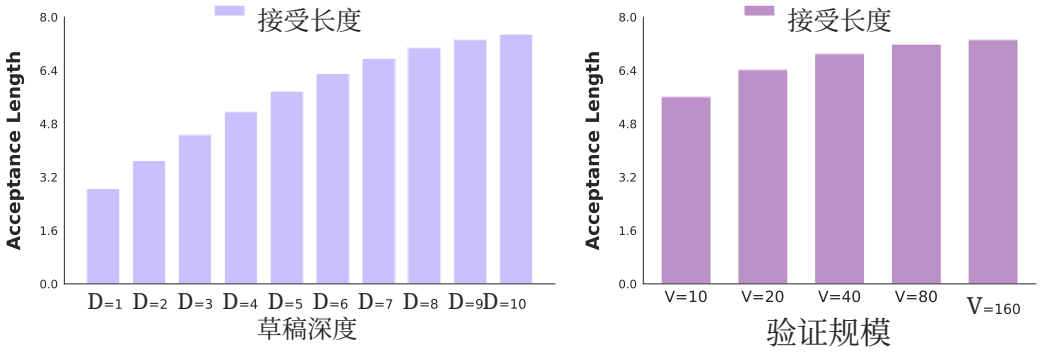


图 5：在 HumanEval 数据集上使用 Eagle3 的 Llama-3.1-8B-Instruct 的平均接受长度。左：在验证规模固定为 60 的情况下改变草稿深度。右：在草稿深度固定为 8 的情况下改变验证规模。

为保守的候选词元数量。这种谨慎策略对于降低验证延迟至关重要。

A.5 消融研究的详细结果

本节展示了消融研究的详细结果。其中包含表 4 中 LTD 各组件贡献的结果，以及表 7 和表 8 中迭代优化的结果。

表 7：Llama3.1-8B 迭代优化

	MT-bench		Gsm8k		Alpaca		Qa		Mean	
方法	加速比	τ	加速比	τ	加速比	τ	加速比	τ	加速比	τ
Eagle3	3.70	6.39	3.58	6.44	4.26	6.93	3.17	5.35	3.68	6.28
Iter0	3.88	6.46	3.76	6.40	4.34	6.82	3.50	5.12	3.87	6.20
迭代1	3.95	6.50	**3.80**	6.47	4.36	6.89	**3.50**	5.18	3.90	6.26
迭代2	**3.96**	6.74	3.79	6.73	4.53	7.31	3.41	5.44	3.92	6.56
迭代3	3.93	**6.74**	3.73	**6.73**	**4.57**	**7.31**	3.44	**5.44**	**3.92**	**6.56**

表 8：Dpsk-8B 迭代优化

	MT-bench		Gsm8k		Alpaca		Qa		Mean	
方法	加速比	τ	加速比	τ	加速比	τ	加速比	τ	加速比	τ
Eagle3	3.67	5.82	4.73	7.39	3.65	5.63	3.14	5.02	3.80	5.53
Iter0	3.88	5.95	4.69	8.03	3.68	5.63	3.37	5.07	3.91	5.72
迭代1	3.96	5.98	4.92	8.14	3.86	5.67	3.46	5.08	4.05	5.78
迭代2	4.02	6.22	4.94	8.36	3.71	5.91	3.52	5.29	4.05	5.97
迭代3	**4.05**	**6.32**	**4.98**	**8.53**	**3.98**	**6.00**	**3.62**	**5.37**	**4.16**	**6.08**

A.6 交互 深度与大小策略之间的作用 ICY

为了验证迭代训练的有效性，我们分析了初始独立训练策略（‘Iter0’）与最终迭代训练的 LTD 策略的测试集分布，如图 6 所示。具体而言，我们考察了在给定接受长度的条件下，候选词元数量与草稿深度之间的关系。观察发现，无论接受长度如何，Iter0 都呈现出固定模式，主要集中在草稿深度为 < 6且词元数量为 < 60的区域。深度较浅（< 6）但词元预算较大的情况（> 60）几乎可以忽略。相比之下，LTD 在不同接受长度下表现出更强的适应性。对于较大的接受长度，相当大比例（43%）的情况同时采用较大的词元预算和更深的树（两者均为 > 60）。此外，当接受长度较小时——这表明生成

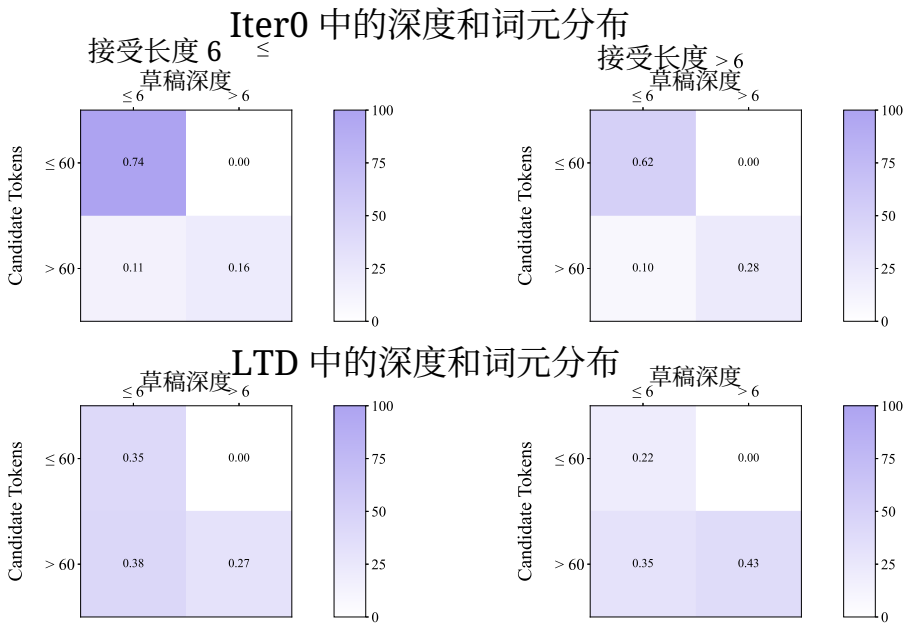


图 6：我们报告了 Llama3-8B 上深度策略与规模策略在 Iter0 和 LTD 中的交互。

难度较高——LTD 经常采用“浅而宽”的策略（深度 < 6，词元 > 60）。这种部署更多候选词元的策略有效缓解了难度增加所导致的接受长度下降。

A.7 深度策略与规模策略的延迟

我们分析了深度策略和规模策略在 Llama-3.1-8B-Instruct 与 Vicuna-13B-v1.3 上的延迟开销。表 9 报告了每个样本的平均生成时间，以及规模策略和深度策略所占的时间成本比例。结果表明，我们的策略引入的计算开销极小：在 Llama 上总开销低于 1.5%，在更大的 Vicuna 模型上甚至更低（1.2%）。值得注意的是，深度策略的延迟大约是规模策略的两倍。尽管我们采用了轻量级单层网络，但由于深度策略的调用频率高于规模策略，其总成本更高。

A.8 规模策略的多层感知机大小选择

多层感知机设计中存在固有的权衡：我们希望更大的多层感知机带来的准确率提升能够超过其增加的推理成本。为验证我们的设计选择，我们在 Llama-3.1-8B-Instruct 上针对规模策略的三种架构大小（隐藏维度：[1024], [1024, 256], 和 [1024, 512, 256]）进行了消融研究，结果见表10。结果显示，层数从 1 增至 2 时，性能显著提升。然而，从 2 层增至 3 层的边际收益微乎其微。因此，我们为规模策略选择了 2 层架构（[1024, 256]），以在训练简便性与推理效率之间实现最佳平衡。

B 训练数据泛化

为评估我们的模型在不同领域的泛化能力，我们采用了大规模多任务语言理解（Hendrycks 等，2020）基准测试，这是一个被广泛采用的数据集，涵盖从初等数学和美国历史到计算机科学和法律的 57 项任务。我们比较了加速比

表 9：Vicuna-13B 和 Llama31-8B 上动态策略的延迟与开销分析。策略延迟以毫秒（ms）计，总生成延迟以秒（s）计，开销以百分比（%）计。

每个样本的时间	MT-bench	Alpaca	Qa	Gsm8K	Mean
Llama-3.1-8B-Instruct					
尺寸策略延迟（ms）	21.4	6.22	6.36	5.83	9.96
深度策略延迟（ms）	40.8	12.4	10.2	11.9	18.8
每个样本的总延迟（s）	4.48	1.28	1.27	1.24	2.07
尺寸策略开销（%）	0.478	0.485	0.500	0.469	0.483
深度策略开销（%）	0.910	0.969	0.803	0.955	0.909
总开销（%）	1.39	1.45	1.30	1.42	1.39
Vicuna-13B-v1.3					
尺寸策略延迟 (ms)	13.7	4.92	4.90	5.56	7.26
深度策略延迟 (ms)	27.9	9.61	8.39	12.0	14.5
每个样本的总延迟 (s)	3.59	1.16	1.10	1.37	1.81
尺寸策略开销 (%)	0.381	0.423	0.444	0.406	0.414
深度策略开销 (%)	0.777	0.826	0.761	0.874	0.810
总开销（%）	1.16	1.25	1.20	1.28	1.22

Table 10：Llama-3.1-8 上规模策略在三种架构配置下的加速比
uBt.指令

配置	MT-bench	Gsm8k	Alpaca	Qa	Mean
[1024]	3.84	3.99	4.39	3.22	3.86
[1024, 256]	3.95	4.03	4.44	3.24	3.92
[1024, 512, 256]	3.96	4.07	4.43	3.25	3.93

图 7 展示了我们提出的方法 LTD 相对于 Llama-3.1-8B-Instruct 模型上 Eagle3 基线的加速比。结果表明，LTD 在 57 个领域中的 54 个领域优于 Eagle3，凸显了我们的方法在不同领域中强大的泛化能力。具体而言，LTD 在数学和逻辑相关任务中的加速比提升超过 10%。值得注意的是，在 LTD 相较于 Eagle3 提升最显著的前五个领域中，基线的加速比表现低于平均水平。这表明，我们的方法有效弥补了 Eagle3 在特定领域的不足。此外，我们的方法并不只在 Eagle3 表现欠佳的情境下有效；即使在 Human Aging 和天文学等 Eagle3 表现已显著高于平均水平的领域，LTD 仍能保持更高的加速效果。

C 强化学习详情

C.1 训练步数分析

为验证训练预算的合理性，我们分析了总训练时间步与数据集（HumanEval）覆盖率之间的关系。以Llama模型为例，HumanEval 数据集共包含约 $N_{\text{total}} = 27829$ 个新词元（真实标签）。

C.1.1 深度策略的步数

对于深度策略，总训练预算设为 $B_{\text{depth}} = 1{,}000{,}000$步。在我们的强化学习表述中，一步对应于一次草稿动作或一次验证动作。

然而，推测解码具有随机性，并非所有起草的词元都会被接受。为了估算有效的数据集遍历次数（轮次），我们必须考虑接受率 α。假设接受率适中（例如 $\alpha \approx 0.7$），生成完整数据集所需的草稿步数

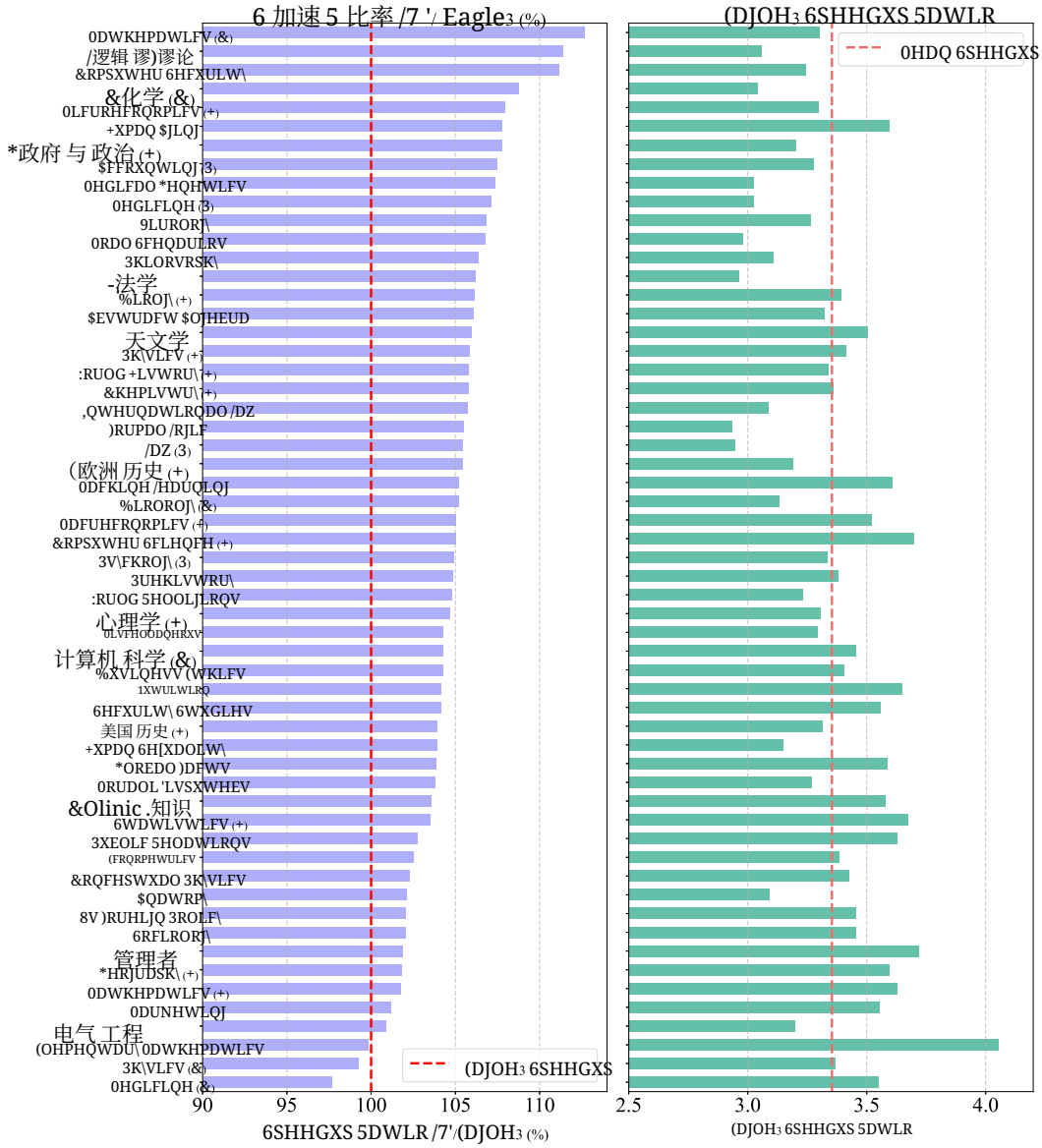


图 7：LTD 与 Eagle3 在 Llama-3.1-8B-Instruct 模型上的加速比对比。我们使用 MMLU 中的 57 个子任务（例如数学、历史、法律），每个子任务抽取 80 个样本进行评估。领域级别缩写为 H（高中）、C（大学）和 P（专业级）。左图：LTD 相对于 Eagle3 的加速比，并以 Eagle3 为标准化基准；数值超过 100% 表示 LTD 的性能优于 Eagle3。右图：Eagle3 基线在不同子任务中的绝对加速比。

大于已接受词元的数量：

$$\text{Step}_{\text{epoch}} \approx \frac{N_{\text{total}}}{\alpha} = \frac{27,829}{0.7} \approx 39,756 \text{ steps} \tag{6}$$

因此，数据集遍历次数 K_{depth} 计算如下：

$$K_{\text{depth}} = \frac{B_{\text{depth}}}{\text{Step}_{\text{epoch}}} \approx \frac{1,000,000}{39,756} \approx 25.1 \text{ passes} \tag{7}$$

在 $\alpha \to 1$ 的理想情形下（或者假设每个新词元 ≈ 1 需要 1 步，并将其作为步数的下界估计），覆盖率约为 1，000，000/27，829 ≈ 35.9 次遍历。因此，我们设置的 1M 步大致相当于对数据集进行**25–40 次遍历**。

鉴于深度策略必须学会在不同上下文中识别最优终止点，并且近端策略优化是一种需要反复采样轨迹的同策略算法，这种程度的探索很常见，也是实现收敛所必需的，并非过度。

C.1.2 规模策略的步数

对于规模策略，预算设为 $B_{\text{token}} = 100{,}000$ 步。该策略决定总草稿预算（动作空间大小：12 种配置，范围从 20 到 240 个词元）。每个推测周期仅做出一次决策。

Ass假设每个周期的平均接受长度为 5 个词元，则每隔一次调用该策略 5
gen生成的词元。遍历一次数据集所需的步数为：

$$\text{Step}_{\text{epoch}} \approx \frac{N_{\text{total}}}{\tau} = \frac{27,829}{5} \approx 5,566 \text{ steps} \tag{8}$$

数据集遍历总次数 K_{token} 为：

$$K_{\text{token}} = \frac{B_{\text{token}}}{\text{Step}_{\text{epoch}}} \approx \frac{100,000}{5,566} \approx 18.0 \text{ passes} \tag{9}$$

由此可得约 **18–20次遍历**。考虑到策略必须探索由 12 种不同配置组成的离散动作空间，以优化延迟与生成长度之间的权衡，这一采样量较为保守，也足以高效地稳定策略梯度。

C.2 基于吞吐量的奖励的收敛

为明确展示训练期间基于吞吐量的奖励动态变化，我们报告了 Vicuna-13B 规模策略和深度策略的奖励曲线与验证集加速比，如图 3 所示。可以看到，随着训练步数增加，两种策略的奖励均稳步提升，且分配的训练预算足以确保收敛。就验证集上的性能而言，我们观察到性能先有所提升，随后呈下降趋势。为防止对训练集过拟合，我们选取在验证集上取得最高加速比的检查点作为最终模型。

C.3 训练耗时

由于这两种策略都较为轻量，强化学习训练阶段效率很高。例如，训练规模策略（100k 步）时，Llama3-8B、Vicuna-13B 和 Qwen3-32B 分别仅需 2.0、2.5 和 3.5 小时。同样，在这些模型上训练深度策略（1M 步）分别需要 5、5 和 7 小时。所有实验均在单张 NVIDIA A100 GPU 上完成。即使进行三轮训练，总计算成本仍低于 30 GPU 小时。在实际应用中，这些策略只需训练一次，而解码则占据实际部署中的主要运行时间；因此，改进推理效率所带来的收益足以抵消额外训练开销。鉴于 LTD 相较 Eagle-3 的加速比最高提升 36.4%，对于延迟敏感型应用而言，这一额外训练成本所换来的权衡极具优势。

C.4 近端策略优化选择

我们特意选择近端策略优化，因为它能够自然地融入在线推测解码流程，使我们可以在标准推理步骤中记录奖励。其他强化学习方法（例如目前广受欢迎的 GRPO）并不适用于我们的任务。我们知道，GRPO 需要从同一状态重复采样，以计算基线。在推测解码场景中，这意味着要么为不同样本反复重建 KV 缓存（计算开销很大），要么依次生成多个续写（这会导致与真实推理环境之间出现时序不匹配）。因此，我们选择近端策略优化来训练策略。

D LTD 在 Griffin 上的有效性

为进一步验证我们的方法在不同推测解码方法中的有效性，我们选择了 Griffin（Hu 等人，2025），这是另一种基于动态树框架的方法。Griffin

表11：在 Griffin 上使用 Meta-Llama-3-8B-Instruct 进行贪心解码时，不同方法的加速比和平均接受长度 τ

	MT-bench		Gsm8k		Alpaca		Qa		Mean	
方法	**加速比**	τ	**加速比**	τ	**加速比**	τ	**加速比**	τ	**加速比**	τ
Griffin	1.83	4.82	3.18	5.16	2.98	4.79	2.20	3.96	2.55	4.68
Griffin+GS	2.81	5.11	**3.19**	5.47	2.95	5.04	2.18	4.10	2.78	4.93
Griffin+LTD	**2.88**	4.98	3.13	5.45	**2.98**	5.01	**2.23**	4.10	**2.81**	4.89

与 Eagle3 相比，它采用了不同的训练损失和草稿模型架构，但加速比略低。我们使用 Meta-Llama-3-8B-Instruct（Dubey 等人，2024），比较了 GRIFFIN 默认设置与我们的 LTD 方法的加速比。表11中的结果表明，我们的方法在 Griffin 上同样实现了最先进的加速效果。

E 声明

大语言模型的使用

大语言模型仅用于语言润色，例如改进语法、优化句子结构以及提升文本的整体可读性。作者对大语言模型提出的所有建议和修改都进行了严格审阅和人工编辑。此过程确保最终文本准确、忠实地表达作者的原创思想、论点和研究发现。本研究的核心智力贡献，包括研究构思、实验设计、数据收集、分析和结果解释，均由人类作者独立完成。大语言模型未参与这些关键研究活动。作者对本文呈现的所有内容承担全部责任，包括数据的准确性和结论的有效性。