arXiv:2602.01762v1 [cs.AI] 2 Feb 2026

PRISM：面向推测采样草稿模型的参数化推理重构

Xuliang Wang [*1 2] **Yuetao Chen** [*3] **Maochan Zhen** [2] **Fang Liu** [2] **Xinzhou Zheng** [4] **Xingwu Liu** [5 2] **Hong Xu** [3] **Ming Li** [1 2]

摘要

受自回归特性限制，大语言模型（LLM）的解码速度较慢。推测解码方法已成为加速大语言模型解码的一种有前景的解决方案，受到系统与人工智能研究领域的关注。近期，对更高草稿质量的追求推动了草稿模型向参数规模更大的方向发展，这不可避免地带来了大量计算开销。现有工作试图平衡预测准确率与计算延迟之间的权衡，而我们通过架构创新解决这一根本难题。

我们提出了 PRISM，将每个预测步骤的计算分配到不同的参数集中，重构草稿模型的计算路径，从而成功地将模型容量与推理成本解耦。通过大量实验，我们证明 PRISM 优于所有现有草稿架构，在保持极低草稿延迟的同时实现了出色的接受长度，从而获得更优的端到端加速。我们还借助 PRISM 重新审视了缩放定律，发现与其他草稿架构相比，随着数据量增加，PRISM 的扩展效果更佳。通过严格、公平的比较，我们表明，PRISM 可使已高度优化的推理引擎的解码吞吐量提升 2.6× 倍以上。

1 引言

大语言模型（LLM）已成为一项基础性技术。OpenAI 的 ChatGPT（OpenAI，2022）、Meta 的 LLaMA（Touvron 等，2023）以及 Google 的 Gemini（Google，2023）在各种任务中展现了卓越能力，证明了其在推动人机交互、优化工业工作流程以及开创科学研究新范式等新型应用方面的潜力。然而，在实际应用中实现这些潜力，往往受到生成效率这一关键问题的制约。

效率问题根植于大语言模型推理的自回归特性。自回归生成通常采用顺序方案，每生成一个词元，都要对其参数进行一次完整的前向传播，因此计算成本高昂，吞吐量低。

为缓解这一问题，研究界提出了多种加速技术。尽管高效的 KV缓存管理（Kwon 等人，2023；Qin 等人，2025）、连续批处理（Yu 等人，2022）、并行优化（Li 等人，2023；Butler 等人，2024）以及预填充与解码分离方案（Zhong 等人，2024b；a；Patel 等人，2025）带来了显著的加速效果，但它们并未改变逐词元生成这一基本范式。作为一种互补方法，推测解码（Leviathan 等人，2023）有效减少了大语言模型前向传播的次数。

推测解码，亦称草稿验证，其灵感来自推测执行，这是现代CPU设计中一项成熟的优化技术。它主要由两个阶段组成：在第一阶段，一个更小、更快的草稿模型提出一串候选词元序列；在随后的阶段，所起草的序列由更大、更慢的目标模型进行验证。直观而言，当多个起草词元被接受时，推测解码使目标模型能够在单次前向传播中生成多个词元，而朴素的自回归解码每次前向传播仅生成一个词元。因此，推测解码的成功在很大程度上依赖于草稿质量，即

[*]同等贡献 [1]滑铁卢大学计算机科学系，大卫·R·切里顿，滑铁卢，安大略省，加拿大 [2]华中人工智能研究院，中国河南郑州 [3]香港中文大学，中国香港特别行政区 [4]中国科学技术大学，中国安徽合肥 [5]大连理工大学，中国辽宁大连。通讯作者：Ming Li<mli@uwaterloo.ca>。

预印本。

表 *1.* 对相关工作中所用草稿模型相对规模的比较，以起草模型参数量除以目标模型规模来衡量。请注意，这些统计数据包含前向传播中激活的全部参数，且所提及的大语言模型为聊天/指令版本。该表显示出采用相对更大的草稿模型的增长趋势。

论文	基础大语言模型	起草模型	相对规模
Leviathan 等人	T5-XXL-11B	T5-SMALL (77M)	0.68%
Miao 等人	LLaMA-2-7B	LLaMA-68M	0.95%
	LLaMA-2-65B	LLaMA-68M	0.10%
Li 等人	LLaMA2-7B	EAGLE-2（0.48B）	6.86%
	LLaMA2-70B	EAGLE-2（1.23B）	1.76%
Li 等人	LLaMA3.1-8B	EAGLE-3（1.03B）	12.88%
	LLaMA3.3-70B	EAGLE-3（2.19B）	3.13%
Tang 等人*	LLaMA-3.3-8B	缩放EAGLE-2（1.59B）	19.88%
	Llama-3.3-70B	缩放EAGLE-2（4.35B）	6.21%
Yan 等人*	LLaMA-2-7B	缩放EAGLE-2（1.26B）	18.00%
	LLaMA-2-13B	缩放EAGLE-2（1.78B）	13.69%

*由于这些行对应的起草模型尚未公开，我们据此计算其起草模型的规模。

草稿模型每次草拟提出的词元中，预计有多少会被目标模型接受。

为提升草稿质量，许多起草模型沿用 EAGLE（Li 等人，2024a）的做法，将先前验证步骤中的目标模型中间隐藏状态作为额外输入。与仅依赖词元的起草模型相比，这种方法能够生成更准确的草稿。许多这类草稿模型只采用一个 Transformer 块，以尽量降低草拟过程的计算开销。

然而，单 Transformer 配置限制了草稿模型的容量，最终也制约了接受率。正如表 1所示，鉴于这一局限，近期研究开始探索采用更大型草稿模型的可行性（Li 等人，2025a；Yan 等人，2025；Tang 等人，2025）。这些研究在保持架构相近的同时，堆叠了更多 Transformer 层。尽管更大型的起草模型确实会带来更多计算开销，但研究者认为，接受率提升所带来的收益足以抵消这部分成本。

在本文中，我们从不同的视角探讨了推测解码中草稿质量与计算开销之间的根本权衡。我们的核心洞见是，虽然需要更大的模型来提升起草模型的预测性能，但同时也能保持较低的草稿开销。我们通过 PRISM 展示了这一点：PRISM 识别出草稿步骤之间的内在差异，并利用条件计算。具体而言，我们重构推理计算路径，将计算分配给不同草稿步骤中的不同参数集，这类似于棱镜将白光分散成光谱。因此，起草模型的总参数量增加，而每个草稿步骤中激活的参数数量保持不变。激活参数较少能够加速推理和训练。关键的是，模型的表征能力仍会提升：通过大量实验，我们表明，PRISM 的架构不仅保留了良好的缩放特性，其扩展效果实际上还优于朴素参数缩放方法，例如纵向堆叠 Transformer 层。

总之，本文通过引入 PRISM 这一新颖的草稿模型架构，解决了推测解码中起草模型预测质量与效率之间的核心权衡。我们的贡献包括：

- **一种将草稿模型容量与草稿计算成本解耦的新颖架构**。 我们提出 PRISM，这是首个将条件计算范式专门应用于自回归生成步骤以扩展模型容量的架构。与容量和计算开销相互纠缠的传统草稿模型不同，PRISM 通过对不同草稿步骤的计算进行参数化解聚，成功地将二者解耦。这一新颖架构为起草模型的有效性-效率权衡提出了新视角，展示了起草模型设计的新路径。

- **重新审视草稿模型的缩放定律**。 我们的工作提供了首个实证证据，表明草稿模型的预测能力可以在不增加激活参数量的情况下有效扩展。这一发现为草稿模型确立了一种新的、更高效的缩放范式。

- **使用推理引擎进行详细评估**。 我们在 SGLang（一款先进的推理引擎）中实现并验证了 PRISM，从而弥合了 AI 学术界与系统学术界在推测解码优化方面的差距。与许多仅在基于 PyTorch 的环境中评估的先前起草模型不同，我们的实验通过高度优化的系统级推理引擎，有力地证明了 PRISM 的有效性。

2 背景

本节简要概述大语言模型推理、推测解码以及最先进的起草模型架构。

大语言模型推理。 大语言模型推理本质上是自回归的，会以概率方式生成词元序列 $T = (t_1, t_2, \ldots\ldots, t_n)$，其中每个词元 t_i 都从由模型权重 θ 和前序词元 $p(t_i|t_1, \ldots\ldots, t_{i-1}; \theta)$ 参数化的条件概率分布中采样得到。这种序列依赖性使得迭代生成过程成为必要。

从计算特性来看，大语言模型推理分为预填充和解码两个不同阶段。

预填充阶段是指处理输入提示的初始阶段。模型通过高度并行的前向传播，同时计算所有提示词元的注意力机制键值（KV）对。该阶段受计算能力限制，原因是需要对整个输入序列执行大量矩阵-矩阵乘法。

预填充后，模型进入迭代解码阶段。与预填充阶段不同，解码阶段受内存带宽限制，因为每一步都需要将完整的模型参数集从片外高带宽内存（HBM）加载到片上处理单元。对于涉及长上下文的场景，这一内存带宽瓶颈会进一步加剧：每生成一个词元，模型还必须读取所有先前词元对应的注意力键值缓存。

推测解码。推测解码的核心原理是使用参数规模较小、速度较快的草稿模型 M_{draft}，根据当前上下文生成候选词元序列，再由参数规模较大、速度较慢的目标模型 M_{target}通过单次前向传播并行验证这些词元。

形式化地，令 $p_{target}(t|c)$ 表示给定上下文 c时下一个词元 t 的概率分布，其定义如 M_{target}所示。类似地，令 $p_{draft}(t|c)$ 表示来自 M_{draft}的分布。在草拟阶段，给定上下文 c，调用 M_{draft} 生成一个包含 γ 个词元的候选序列 $\tilde{T} = (\tilde{t}_1, \tilde{t}_2, ..., \tilde{t}_\gamma)$。随后在验证阶段，$M_{target}$ 对拼接后的输入 $\tilde{}(c, T)$ 执行一次前向传播，以高效计算每个位置的真实概率分布：$p_{target}(\cdot|c)$、$p_{target}(\cdot|c, \tilde{t}_1)$、.....、$p_{target}(\cdot|c, \tilde{t}_1, ..., \tilde{t}_{\gamma-1})$。草稿词元采用修改后的拒绝采样方案依次进行验证：对于 $\tilde{}$，每个草稿词元 t_i 在 i 的初始值为 1 时，若随机抽取的数值 $r \in$ 满足 Uniform(0，1)，则该词元被接受

$$r < \min(1, \frac{p_{target}(\tilde{t}_i|c, \tilde{t}_{<i})}{p_{draft}(\tilde{t}_i|c, \tilde{t}_{<i})})$$

该过程会遍历 i，直到接受所有词元 $\tilde{}$ $\hat{}$，或词元 t_k 遭到拒绝。所有已接受的词元 T 都会保留。此外，还可以从修正分布 $p'_{target} \propto \max(0, p_{target}(\cdot|c, \hat{}T) - p_{draft}(\cdot|c, \hat{}T))$ 中采样一个额外词元。随后，模型会使用新生成的词元更新上下文，丢弃与任何被拒绝候选词元对应的 KV缓存条目，并开始新的草稿生成周期。

推测解码一个根本且关键的特性是无损性。这一点已得到形式化证明并有保证：推测过程生成的最终输出序列，与逐个词元生成时所遵循的概率分布完全相同，后者由 M_{target} 单独（Miao 等人，2024）。

从系统角度看，推测解码的效用在于能够摊销受内存带宽限制的解码步骤所带来的高昂成本。传统的每次前向传播生成一个词元的自回归方案严重受内存带宽限制，导致计算单元因等待数据而停滞，GPU 利用率低下。推测解码通过单次前向传播并行验证多个词元，从而改变了工作负载。当草稿模型的预测与目标模型高度一致、接受率较高时，每个词元的有效成本会大幅降低。这一机制从根本上改变了解码阶段的运行特性，提高了计算与内存的比率。

最先进的草稿模型。早期的推测解码方法主要使用目标大语言模型的轻量版本作为草稿模型。EAGLE（Li 等人，2024a）带来了重大突破：它首次不仅利用前文中的上下文词元，还利用目标模型在最近一次验证步骤中生成的隐藏状态。通过以这种信息量更丰富的输入作为条件，EAGLE 的草稿模型能够更准确地预测目标模型的分布，从而显著提高候选词元的接受率。利用目标模型提供的信息丰富的中间表示这一方法成效显著，因此后续研究也采用了这一方法（Li 等人，2024b；2025a；Huang 等人，2025；Zhang 等人，2025a）。

大多数最先进的草稿模型也采用基于树的草拟与验证方法，而非朴素的基于序列的方法。Specinfer（Miao 等人，2024）最初提出了这一方法，其核心观察是：即使最有可能的草稿词元被拒绝，同一生成概率分布中概率较低的备选候选项仍可能正确。通过允许在同一个生成步骤中验证多个词元，草稿令牌会形成不同的候选路径，并可组织成树结构。在验证阶段，会应用一种特殊的树形注意力掩码，以强制草稿令牌之间遵循因果关系。基于树的草拟与验证显著提高了接受率。

草稿模型的规模扩展。过去，人们对草稿模型的预期一直受制于尽量降低计算开销这一原则，因此起草模型的参数规模较小。然而，这种范式从根本上限制了模型的容量，以及其从大规模训练数据中获益的能力。随着人们逐渐认识到这一局限，近期研究开始转向在草稿模型复杂度与计算延迟之间权衡。目标是找到一个最佳平衡点：采用能力更强、尽管成本更高的草稿模型，并使其达到足够高的接受率，从而实现显著的端到端加速。

EAGLE-3（Li 等人，2025a）率先研究了有意扩大草稿模型规模的问题，展示了增加可训练参数并改进其他架构后，在使用更多数据进行训练时能够显著提升性能；Scylla（Yan 等人，2025）将 EAGLE-2 架构扩展至多达五个可训练的 Transformer 层，并在规模大得多的数据集上对其进行预训练和微调，从而取得了业界领先的接受长度。与此同时，Meta（Tang 等人，2025）探索了不同的扩展维度，研究了参数扩展方法，例如增加模型深度，以及用混合专家（MoE）层替代标准前馈网络（FFN）。这些努力共同表明，人们对草稿模型的预期正在转变：从追求最低计算开销，转向兼顾性能、以实现系统吞吐量最大化。

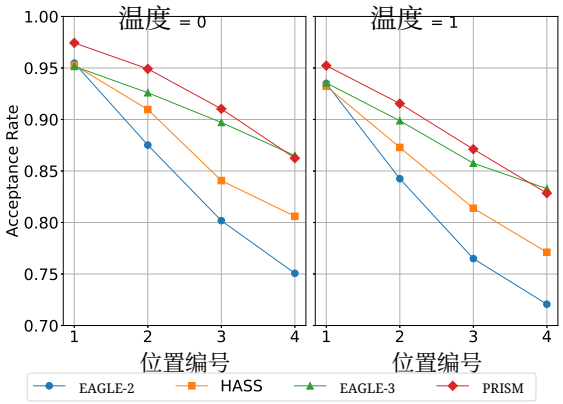

图 *1*。展示 LLaMA-3-8B 逐步平均接受率的实证结果。

3 方法

3.1 动机

随着扩展草稿模型以实现高接受率的可行性逐渐显现，使用显著更多的参数训练草稿模型颇具吸引力。为了在不引入额外开销的情况下扩展模型容量，我们从混合专家模型中获得启发：这类模型通过将计算路由至不同的子网络来发挥专业化优势。我们观察到，推测解码中也存在类似的专业化机会：推测解码中的预测任务在难度上并不均匀。经验证据清晰地表明，序列中越靠后的草稿步骤越难。这一点始终可以从后续词元接受率的急剧下降中得到体现（见图 1）。这一洞见直接促成了我们提出的架构 PRISM。PRISM 不再依赖单一的、适用于所有情况的模型，而是通过为不同的草稿步骤应用不同的参数集合，采用专业化原则。这种设计直观上带来两项优势：

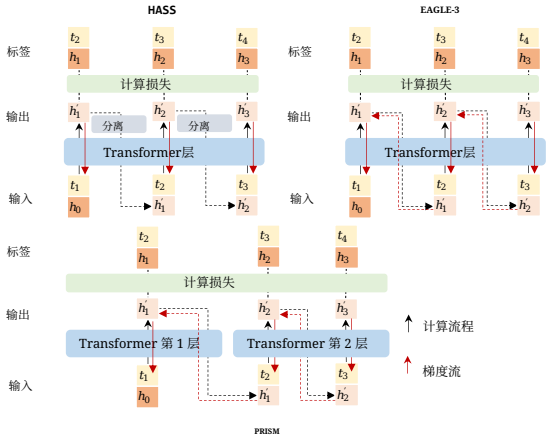

图 *2*。HASS、EAGLE-3 和 PRISM 的计算与梯度流示意图。

- **自适应表示复杂度**。 PRISM 有效地构建了一个分层模型，其中后续参数处理先前参数的输出，从而形成级联计算结构。该设计能够在各草稿步骤之间自适应分配计算资源。如图2所示，与先前所有草稿步骤均使用相同

  在各个草稿步骤中，PRISM 逐步增加有效深度。通过这种架构设计，序列后半部分较难的预测任务自然会分配到更多累积计算；此时不确定性通常会增加，准确预测词元也更具挑战性。

- **将模型容量与推理成本解耦**。关键在于，这种架构将模型的总学习容量与每步推理成本解耦。我们可以大幅增加可训练参数的总数，使模型能够学习更复杂的函数，同时生成任意单个草稿令牌的计算成本仍保持恒定且极低。

3.2 PRISM 架构

PRISM，即面向推测解码草稿模型的参数化推理重构，遵循最先进的草稿模型的惯例，采用基于 Transformer 的草稿模型。如图3所示，PRISM 由词元嵌入层、最终词汇投影头和一系列处理模块组成。每个处理模块由一个全连接融合层和一个 Transformer 层组成。

PRISM 中草稿步骤与处理模块之间的关系由一个满射定义。我们通过为每个草稿步骤分配一个专用于其计算的处理模块来建立该满射。

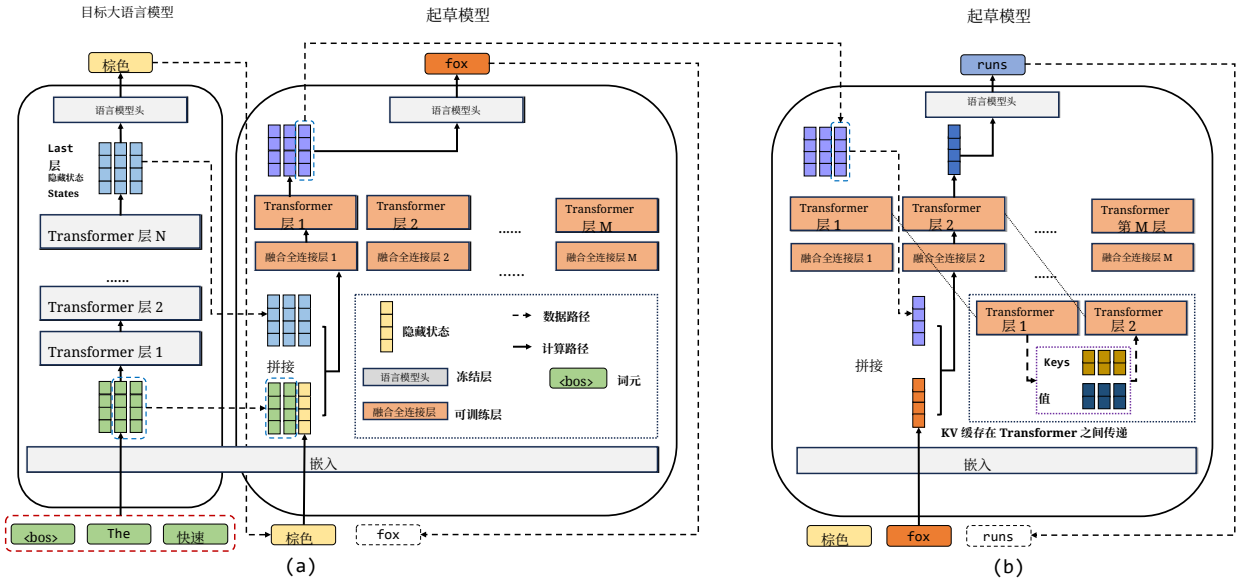

图 *3*。 PRISM 架构及其计算路径示意图。(a) 目标模型对多个词元进行预填充或验证，其最后一层Transformer的输出隐藏状态将作为草稿模型的输入，并与输入词元嵌入融合。融合后的特征被馈送至PRISM草稿模型的第一个Transformer进行预填充。(b) 将前一个生成词元的嵌入与前一个生成的隐藏状态融合后，在第一个解码步骤中，将融合后的特征应用于第二个Transformer层。请注意，Transformer 1中的KV缓存会被传输至Transformer 2。

表 *2*。 PRISM组件和变量的符号表示

符号表示	描述
S_{acc}	n 已接受词元的序列，来自先前验证：(x_1，……，x_n)
H_{base}	S_{acc} 的目标模型隐藏状态：(h_0, h_1, ..., h_{n-1})
E	词元嵌入层（冻结）
L_1, ..., L_M	M 个不同的 Transformer 层
$F_i(\cdot,\ \cdot)$	用于组合的融合函数 L_i 词元嵌入和隐藏状态
f_{map}	将草稿步骤映射为满射 K 到 M 模块：1、…… $K \to 1$、…… M
LM_{head}	词汇投影层（冻结）

这种映射允许多对一的关系：虽然每个步骤都有唯一分配的模块，但单个处理模块可以服务于多个步骤。

使用 P 起草RISM。 如图3所示，草稿过程的特点是在推理前向传播之间采用独特的主动参数切换过程。

形式上，按照表2定义模型的组件和变量。

草稿以对草稿模型进行预填充开始。对于每个接受的词元 $x_i \in S_{acc}$，通过将其嵌入与目标模型中对应的隐藏状态 h_{i-1} 相结合，并将其输入融合函数

$$H_{\text{fused}} = [F_{f_{map}(1)}(E(x_i), h_{i-1})]_{i=1}^{n}$$

随后，这一融合表示序列经由指定的Transformer层进行单次前向传播

$$H'_{out}, KV_{(1)} = L_{f_{map}(1)}(H_{fused}, KV_{(0)})$$

其中 $KV_{(0)}$ 是KV缓存的初始状态。输出由隐藏状态序列 H'_{out} 和更新后的KV缓存 $KV_{(1)}$组成。请注意，KV 缓存将在各处理模块之间共享。来自H'_{out}的最终隐藏状态，即 h'_n，成为起草模型自回归解码阶段的初始预测状态。

$$\hat{h}_0 = h'_n$$

解码阶段的初始输入词元$\hat{}x_1$ 同时也是$\hat{}$第一个草稿词元。它应使用分类头直接从 h_0预测得出

$$\hat{x}_1 = \arg\max(LM_{head}(\hat{h}_0))$$

剩余的草稿步骤都是重复类似流程的解码步骤：对于 $k = 2,\ 3,\ \ldots\ldots K$,

$$H_{in}^{(k)}, KV_{(k)} = F_{f_{map}(k)}(E(\hat{x}_{k-1}, \hat{h}_{k-2}), KV_{(k-1)})$$

$$\hat{h}_{k-1} = L_{f_{map}(k)}(H_{in}^{(k)})$$

$$\hat{x}_k = \arg\max(LM_{head}(\hat{h}_{k-1}))$$

整个草稿过程以输出草稿词元 ^ ^ ^(h_1， h_2，..... h_k) 结束，随后将这些词元送去验证。

值得注意的是，上述公式假设生成路径是线性的。这是为了便于清晰呈现而做出的简化。实际上，PRISM机制与许多其他起草模型采用的树结构草稿和随机采样方法完全兼容。

训练 PRISM。为缓解暴露偏差问题，许多起草模型会在训练期间采用上下文对齐技术，从而显著提高接受长度。典型示例包括 HASS 中的协调上下文对齐（Zhang 等人，2025a）、EAGLE-3 中的训练时测试（Li 等人，2025a），以及 Griffin 中的词元对齐（Hu 等人，2025）。上下文对齐会对齐训练和推理过程中的输入隐藏状态、KV 缓存，甚至输入词元。使用 PRISM 时，我们沿用了 HASS 的做法，对齐了 3 步的隐藏状态和 KV 缓存。

上下文对齐通过要求每个草稿步骤都单独进行一次前向传播，来模拟训练中的自回归推理过程。这会大幅增加训练成本：例如，训练 EAGLE-3 起草 n 个词元时，每个训练样本都需要进行 n次独立的前向传播。

PRISM 架构在进行上下文对齐时，训练效率更高。参数量相近的单体草稿模型每一步都需要在整个网络深度中传播激活值和梯度，而 PRISM 的前向传播及随后的反向传播则仅限于一个子网络。

训练 PRISM 时，我们首先进行预热阶段，只使用一个处理模块训练模型，也就是说，前向传播之间不会切换参数。训练损失由两部分组成：交叉熵损失（CEL），用于比较草稿模型输出的词元分布与真实标记；以及均方误差（MSE），用于对齐目标模型生成的隐藏状态与草稿模型生成的隐藏状态。

$$L_1 = \sum_{t=1}^{T} \text{CEL}(\text{softmax}(LM_{head}(z_i^{(t)})), y_i)$$

$$L_2 = \sum_{t=1}^{T} \text{MSE}(z_i^{(t)}, z_i)$$

$$L_{train} = \lambda_1 L_1 + \lambda_2 L_2$$

其中 z_i 是目标模型生成的第 i个隐藏状态，$z_i^{(t)}$ 是草稿模型在对该样本进行第 t次前向传播后输出的第 i个隐藏状态。 y_i 是第 i个真实词元。 λ_1 和 λ_2 是系数。

预热后，我们复制训练好的 Transformer 权重，生成 M 个训练好的 Transformer 副本

表 3. 训练数据的构成

数据集	数量	比例	主题覆盖范围
ShareGPT	68K	8.5%	问答/代码/数学/逻辑
UltraChat	463K	57.9%	QA
OpenThoughts2	269K	33.6%	数学/逻辑

然后进行第二阶段训练。此时，我们移除 MSE 损失，并在前向传播之间切换 Transformer。

3.3 与 SGLang 集成

虽然在 PyTorch（Paszke 等人，2019）中实现推测算法很方便，但由于缺少部署场景中的优化（例如 CUDA 图和连续批处理），推测解码的效果可能会被高估。因此，要切实评估其有效性，有必要将 PRISM 集成到 SGLang 这样的高效推理框架中。我们在 SGLang 引擎中实现 PRISM，并将其与 CUDA 图、连续批处理和其他推理优化协同运行。

4 实验

我们在多种设置下对 PRISM 进行了实验。

目标大语言模型。 我们在两种广泛使用的开源大语言模型上进行了大量实验，即 LLaMA-2-chat-7B 和 LLaMA-3-Instruct-8B（下文简称 LLaMA-2-7B 和 LLaMA-3-8B）。选择这些模型是因为它们是以往大多数研究中的目标大语言模型。尽管二者的隐藏状态维度相同，LLaMA-2-7B 和 LLaMA-3-8B 在许多方面却有很大差异。首先，LLaMA-3 使用字节对编码（BPE）构建词表，而 LLaMA-2 采用 SentencePiece，因此二者的词表截然不同；LLaMA-3 使用分组查询注意力（GQA），而 LLaMA-2 则不使用；二者的训练数据集和训练流程也大不相同。草稿模型对上述所有因素都很敏感。

数据集。 为验证缩放定律并评估各种草稿模型架构的缩放能力，我们收集并预处理了多个广泛使用的开源数据集，包括 ShareGPT、UltraChat（Ding 等人，2023）和 OpenThoughts2（Guha 等人，2025）。我们将每个样本截断至最长 2048 个词元，并移除 OpenThoughts2 样本中 think 标签之间的内容，以此对数据进行预处理。我们从这三个数据集中抽取了约 800,000 个样本。数据集的构成如表 3所示。为确保公平比较，本文提及的基线方法也在同一数据集上进行训练。

表*4*。英伟达 A800 图形处理器上不同方法的接受长度（接受长度，AL）和吞吐量（TPS，每秒令牌数）

			MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		自然问答	
模型	方法	Temp	AL	TPS	AL	TPS	AL	TPS	AL	TPS	AL	TPS	AL	TPS
LLaMA-2	Vanilla *	T = 0	N/A	95.66	N/A	96.10	N/A	96.51	N/A	96.75	N/A	92.77	N/A	96.62
		T = 1	N/A	95.70	N/A	96.08	N/A	96.31	N/A	96.51	N/A	92.66	N/A	96.447
	标准 **	T = 0	2.68	142.35	2.70	145.65	2.88	155.23	2.88	155.49	2.17	109.92	2.83	152.58
		T = 1	2.68	142.37	2.55	136.65	2.77	146.76	2.58	137.17	2.12	104.46	2.61	138.58
	EAGLE-2	T = 0	4.16	246.18	4.78	283.84	4.29	254.13	4.08	243.54	3.88	218.83	3.80	225.49
		T = 1	3.87	223.69	4.49	260.25	4.26	246.24	3.87	225.22	3.70	210.09	3.60	208.45
	HASS	T = 0	4.48	265.08	5.15	305.31	4.54	268.94	4.42	263.70	4.18	235.41	4.08	241.61
		T = 1	4.12	238.13	4.77	276.39	4.43	256.26	4.21	244.44	3.99	226.91	3.79	219.49
	PRISM（我们的方法）	T = 0	**4.66**	**274.32**	**5.33**	**316.16**	**4.73**	**279.14**	**4.67**	**277.35**	**4.40**	**255.08**	**4.24**	**250.31**
		T = 1	**4.30**	**246.64**	**4.90**	**284.43**	**4.68**	**269.19**	**4.39**	**254.15**	**4.15**	**231.13**	**4.02**	**229.80**
LLaMA-3	Vanilla *	T = 0	N/A	85.63	N/A	85.88	N/A	85.98	N/A	86.19	N/A	83.18	N/A	86.12
		T = 1	N/A	85.39	N/A	85.70	N/A	85.85	N/A	85.79	N/A	82.90	N/A	85.86
	标准 **	T = 0	5.06	164.13	5.77	180.66	5.47	173.28	4.56	144.51	4.43	143.18	4.30	136.81
		T = 1	4.42	142.69	5.45	169.80	5.20	163.19	4.16	131.12	4.17	124.41	3.77	119.45
	EAGLE-2	T = 0	3.67	168.26	4.57	212.36	4.24	197.66	3.67	171.89	3.59	163.34	3.14	147.74
		T = 1	3.67	168.64	4.30	182.15	4.00	172.00	3.37	146.32	3.30	137.69	2.99	129.65
	HASS	T = 0	3.93	180.50	5.12	235.38	4.73	216.75	3.94	184.39	3.90	176.16	3.36	156.67
		T = 1	3.93	181.14	4.84	206.69	4.49	191.51	3.62	155.73	3.58	147.01	3.08	133.46
	PRISM（我们的方法）	T = 0	**4.29**	**201.51**	**5.43**	**251.96**	**5.11**	**237.41**	**4.25**	**199.31**	**4.17**	**187.95**	**3.63**	**170.06**
		T = 1	**4.29**	**201.47**	**5.02**	**216.30**	**4.76**	**205.09**	**3.86**	**167.72**	**3.76**	**158.33**	**3.31**	**143.63**

* Vanilla 指标准自回归解码。因此，Vanilla 解码没有接受长度。** 标准表示使用同一系列模型的轻量版本作为起草模型。对于 LLaMA-2-7B，我们使用 LLaMA-160M 模型；对于 LLaMA-3-8B，我们使用 LLaMA-3.1-1B 模型。4090 GPU 上的实验也采用相同设置。

基准测试。 遵循以往研究，我们在六个基准测试上进行评估，即 MT-bench（Chen 等人，2025）、HumanEval（Chen 等人，2021）、GSM8K（Cobbe 等人，2021）、Alpaca（Taori 等人，2023）、CNN/Daily Mail（See 等人，2017）和 Natural Questions（Kwiatkowski 等人，2019），每个基准测试分别代表截然不同的工作负载。MT-bench 用于多轮对话；HumanEval 用于代码补全；GSM8K 用于评估大语言模型的小学数学能力；Alpaca 是一个用于指令遵循的数据集；CNN/Daily Mail 用于摘要任务，而 Natural Questions 由用户在 Google 上搜索的真实问题构成。

指标。 我们重点关注以下指标来评估草稿模型：

- **接受长度**。接受长度是目标模型在每个验证步骤中接受的草稿令牌的期望数量。它反映了草稿模型预测目标模型输出的能力。

- **吞吐量**。吞吐量以每秒令牌数衡量，表示推测解码系统端到端生成词元的速率。它不仅受接受长度影响，也受计算开销影响。

基线。 我们将 PRISM 与多种基线草稿模型进行比较，包括标准推测解码、EAGLE-2、HASS 和 EAGLE-3。

训练设置。 我们使用规模从 100,000 到 800,000 个样本的数据集训练 PRISM 和所有基线起草模型，使用 8 张英伟达 A100 40G 图形处理器和 4 张英伟达 A100 80G 图形处理器。通常，较小的数据集会训练更多轮次，较大的数据集则训练较少轮次。训练耗时因数据集规模而异，从 1 天到 2 周不等。

请注意，主实验中评估的 PRISM 由两个处理模块组成。第一个模块专门用于处理第一个预填充步骤，第二个模块则负责后续所有解码步骤。

4.1 有效性

表4展示了我们的主要结果：在六种工作负载、两种目标模型以及贪心和非贪心采样设置下，将 PRISM 与基线方法进行了比较。实验模拟了使用 1 个 NVIDIA A800 80G GPU 进行目标大语言模型推测采样生成的情况。另有一组使用 2 个 NVIDIA 4090 24G GPU 的实验，见表5。所有实验均采用批大小为 1。树形草稿和验证使用一个 6 步、4 分支的树结构。每次验证最多验证 16 个词元。在所有测试设置中，PRISM 相较于所有基线方法均表现出稳定优势。与标准的原始自回归解码相比，PRISM 始终实现超过 2.4 倍的加速。与推理时激活相同数量参数的其他基线起草模型相比，

表5. 英伟达 4090 GPU 上不同方法的接受长度（AL）和吞吐量（TPS，每秒令牌数）

			MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		自然问答	
模型	方法	Temp	AL	TPS	AL	TPS	AL	TPS	AL	TPS	AL	TPS	AL	TPS
LLaMA-2	Vanilla	T = 0	N/A	101.30	N/A	103.57	N/A	104.25	N/A	103.89	N/A	99.73	N/A	104.52
		T = 1	N/A	102.27	N/A	103.27	N/A	102.54	N/A	103.80	N/A	99.60	N/A	103.38
	标准	T = 0	2.77	134.85	2.72	133.32	2.88	140.77	2.90	143.16	2.05	96.59	2.84	136.31
		T = 1	2.78	136.56	2.62	129.76	2.85	138.21	2.85	137.00	2.02	94.20	2.79	134.66
	EAGLE-2	T = 0	4.12	223.32	4.75	251.94	4.26	222.95	4.09	217.64	3.89	197.64	3.84	201.01
		T = 1	4.12	223.28	4.48	218.69	4.24	205.44	3.92	192.05	3.71	174.54	3.64	173.72
	HASS	T = 0	4.42	238.65	5.12	271.59	4.51	238.27	4.40	233.42	4.20	213.89	4.08	217.04
		T = 1	4.43	241.22	4.74	233.54	4.50	218.77	4.18	207.23	4.01	190.18	3.78	179.97
	PRISM（我们的方法）	T = 0	**4.69**	**254.18**	**5.26**	**287.77**	**4.68**	**256.28**	**4.62**	**256.10**	**4.35**	**217.02**	**4.24**	**232.58**
		T = 1	**4.65**	**253.61**	**4.88**	**250.55**	**4.63**	**234.73**	**4.34**	**223.28**	**4.18**	**194.20**	**3.96**	**195.25**
LLaMA-3	Vanilla	T = 0	N/A	91.60	N/A	91.51	N/A	91.42	N/A	91.32	N/A	88.51	N/A	90.88
		T = 1	N/A	91.72	N/A	91.02	N/A	90.83	N/A	91.01	N/A	88.46	N/A	90.97
	标准	T = 0	5.08	149.47	5.93	171.76	5.75	166.41	4.80	139.88	4.75	135.04	4.48	130.60
		T = 1	5.08	150.61	5.36	154.60	5.31	152.54	4.28	124.80	4.30	121.33	3.93	115.29
	EAGLE-2	T = 0	3.76	166.75	4.57	198.19	4.26	186.82	3.63	160.91	3.58	153.15	3.24	142.68
		T = 1	3.77	168.10	4.35	154.91	4.02	142.84	3.37	120.82	3.34	115.65	0.98	107.00
	HASS	T = 0	4.01	184.12	5.16	230.42	4.74	212.79	3.91	178.80	3.93	170.92	3.41	155.38
		T = 1	4.01	183.37	4.75	169.27	4.45	157.93	3.58	127.26	3.58	124.07	3.13	112.40
	PRISM（我们的方法）	T = 0	**4.27**	**198.21**	**5.44**	**252.18**	**5.06**	**235.85**	**4.21**	**198.16**	**4.15**	**178.01**	**3.58**	**168.44**
		T = 1	**4.30**	**198.65**	**5.08**	**182.53**	**4.71**	**170.26**	**3.85**	**139.94**	**3.71**	**125.98**	**3.21**	**117.48**

PRISM 的接受长度平均比 EAGLE-2 和 HASS 分别提高 14.09% 和 5.69%，每秒生成的词元数分别增加 14.21% 和 6.10%。推测解码的性能取决于具体任务。在 CNN/Daily Mail 工作负载上，与其他基线相比，PRISM 表现尤为出色，优于 HASS；这或许表明，步骤拆分架构可能更适用于长上下文任务。

4.2 重新审视 PRISM 的缩放定律

我们进行实验，以验证先前工作中观察到的起草模型缩放现象（Li 等人，2025a；Yan 等人，2025）。我们还通过这些实验证明，即使不增加每次预测所激活的参数数量，随着训练数据量的增长，模型的能力也会增强。这种增强体现在预测性能的显著提升上。为了更清楚地展示不同训练数据规模下各类草稿模型预测能力的上限，我们在实验中采用了更极端的树结构：一个包含 6 步、每步 10 个分支的树，每一步最多验证 60 个词元。

首批结果见图4，其中将 PRISM 与推理期间激活参数量相近的起草模型进行比较，即 EAGLE-2 和 HASS。在所有测试的数据规模下，PRISM 的接受长度均显著优于 EAGLE-2 和 HASS。

我们进一步分析这些曲线。首先，在低数据量阶段当训练样本少于 200,000 个时，单变换器模型的表示能力尚未达到饱和。这表现为随着数据量增加，其性能仍在持续提升。然而，在这一规模下，PRISM 已经超越了两个基线模型。这支持了我们的假设：随着数据经过其堆叠的处理模块，PRISM 的层次化结构逐步生成信息量更丰富的表示，从而带来更准确的预测。其次，在高数据量阶段，即曲线的后半段，EAGLE-2 和 HASS 都明显出现了缩放瓶颈。尽管 PRISM 在样本量超过 600,000 个时仍展现出扩展潜力，EAGLE-2 和 HASS 的性能分别在达到 400,000 个样本后开始趋于平稳。这表明，与 EAGLE-2 和 HASS 等单变换器草稿模型相比，PRISM 具有更强的扩展能力。

我们还将 PRISM 与专为扩展而设计的架构 EAGLE-3 进行比较。EAGLE-3 增加了注意力层参数，并利用目标大语言模型生成的多个隐藏状态，而 PRISM 并未采用这些设计。

图5 比较了 PRISM 和 EAGLE-3 在作为 LLaMA-3-8B 的起草模型进行训练时的平均扩展能力。PRISM 展现出更高的数据效率，在使用相对少量的数据训练时优于 EAGLE-3。随着训练数据量增加，两种方法的接受长度趋于一致。我们认为，这是由于数据集的复杂度有限。尽管二者的预测性能相近，PRISM 和

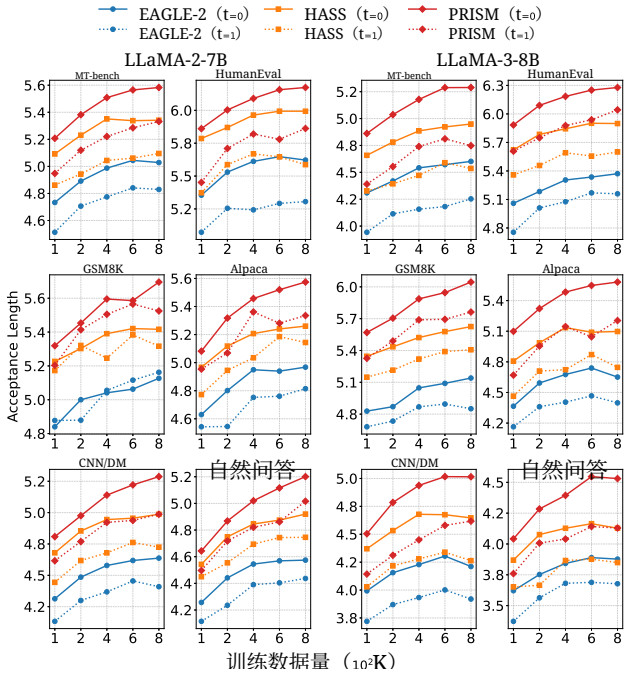

图*4.* PRISM 与对比草稿模型在 6 个基准测试以及贪心和非贪心采样下的扩展曲线。

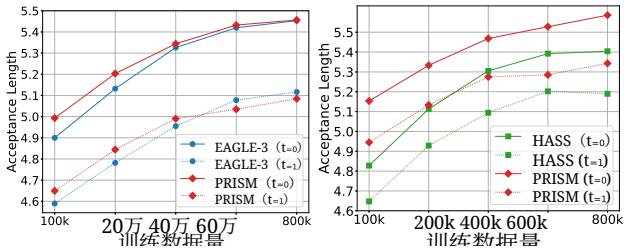

图*5.* PRISM 与 EAGLE-3 图*6.* PRISM 的消融实验

EAGLE-3 在逐步表现上截然不同。如图 1所示，PRISM 在初始步骤的预测更好，而 EAGLE-3 在后续步骤的预测更好。与 EAGLE-2 和 HASS 相比，两种方法的斜率都更平缓。对于 EAGLE-3，这可归因于其训练时测试上下文对齐以及信息更丰富的输入；对于 PRISM，则归因于我们的逐步专用参数。

我们还进行了消融实验，以证明扩展效果直接源自 PRISM 的核心创新：将草稿步骤计算分配给不同的参数组。我们通过将 PRISM 与一个直接堆叠两个 Transformer 的模型进行比较来开展该实验。我们使用与 PRISM 实例完全相同的上下文对齐和数据集训练对比模型。结果如图 6 所示。缺少这项核心创新的消融模型，尽管拥有更多参数，其预测效果和扩展表现仍远逊于 PRISM。

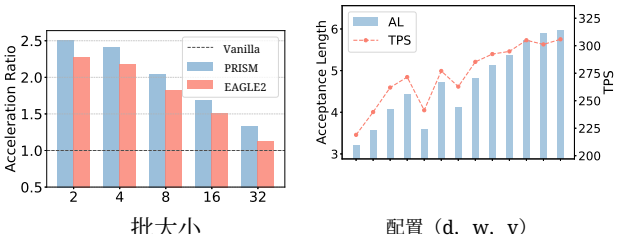

图 *7.*PRISM 与 EAGLE-2 的加速对比 图 *8.* 不同树拓扑结构下，批次长度增大时 PRISM 的接受率和吞吐量。

4.3 超参数

推测解码是一个复杂的系统，许多超参数都会影响端到端加速。本节将展示这些影响，并揭示如何实现更高端到端吞吐量的相关见解。

批大小。随着批大小增加，推测采样的优势会逐渐减弱，因为较大的批大小会使大语言模型解码更加依赖计算，而较少受内存带宽限制。采用推测采样还是使用较大的批大小，往往需要权衡。尽管如此，PRISM 在相对较大的批大小下评估时，仍能提供可观的加速效果，如图 7所示。

树拓扑结构。在推测采样中，树拓扑结构决定了草稿模型生成的候选词元树的深度 (d) 和宽度 (w)。为使验证步骤的计算成本保持在可控范围内，已验证节点的总数 (v) 通常是固定的。然而，最优树拓扑结构在很大程度上取决于草稿模型的预测特性。例如，能够准确进行长程预测的草稿模型，采用更深、更窄的树可能效果最佳。相反，预测准确率下降较快的模型，则可能更适合采用更宽、更浅的树，以探索更多并行分支。在图 8 中，我们探究了树拓扑结构对 PRISM 的影响。

实验结果表明，采用更大的树通常可以提升 PRISM 的性能。然而，随着树深度和宽度增加，速度提升会逐渐趋于饱和。这是因为更大的树会增加草稿阶段和验证阶段的开销。具体而言，在待验证词元数量固定的情况下，较深且较窄的树结构优于较浅且较宽的树结构。

5 相关工作

应用推测采样技术以加速大语言模型最早由 Leviathan 等人 2023提出。自那以后，

系统和人工智能研究界都已做出大量努力来改进该框架。

已有研究开始探索大语言模型服务系统中的推测采样。SpecInfer（Miao 等人，2024）首次引入了基于树的推测推理和词元验证机制，以替代增量解码器，从而显著降低验证开销。Swift-Spec（Zhang 等人，2025b）发现，在草稿模型和目标模型采用相同张量并行配置这一常见情况下，二者的计算需求并不均衡。该方法以异步、解耦的方式重新设计推测解码流水线，从而根据需求分配计算资源。AdaServe（Li 等人，2025b）通过面向服务等级目标定制的推测解码，支持多种服务等级目标需求。它借助硬件感知算法和推测-选择-验证流水线，提高了有效吞吐量并减少了服务等级目标违约。此外，由于包括展开在内的强化学习已成为增强大语言模型的关键方法，而展开长度的不均衡会导致图形处理器利用率低下，推测采样也是缓解这一问题并加速强化学习训练的有前景方法（He 等人，2025）。

另一项重要研究方向是通过增加每一步接受的词元数量，提升推测解码的效率。Medusa（Cai 等人，2024）开创了用于草稿生成的多头方法。Medusa 并未使用独立的草稿模型，而是通过在预训练主干网络的最后一层附加多个轻量级解码头来对其进行扩展。EAGLE 系列（Li 等人，2024a；b；2025a）则通过将目标模型中更丰富的特征表示馈入草稿头，进一步提升接受长度。上下文的增强提高了接受率，从而带来更大的端到端加速。EAGLE-2 引入了动态树构建算法，而 EAGLE-3 则通过在训练期间融入上下文对齐技术，并利用目标模型的额外隐藏状态，进一步提升了性能。HASS（Zhang 等人，2025a）独立开发了自己的上下文对齐方案，提出了一种新颖的两阶段训练方案及专门设计的损失函数。针对同样的上下文不匹配问题，Griffin（Hu 等人，2025）提出，在训练和推理期间，不仅草稿模型与目标模型的隐藏状态需要同步，输入词元本身也应保持同步。Jakiro（Huang 等人，2025）则转向另一研究重点；该方法在非贪心采样场景中表现突出，并将混合专家和对比学习引入草稿模型训练。

最后，Scylla（Yan 等人，2025）探讨了扩展草稿模型所面临的挑战，研究了增加起草模型参数量的影响。Meta（Tang 等人，2025）同期独立开展了类似的实验，证实了能力更强、规模更大的起草模型能够带来显著的性能提升。

6 结论

本文提出 PRISM，一种面向推测采样草稿模型的新型架构。PRISM 应对草稿模型有效性与效率之间的核心权衡。通过对草稿步骤的计算进行参数化拆分，草稿模型容量得以提升，而每一步的计算成本保持不变。我们开展了大量实验，在验证草稿模型缩放定律的同时，证明了 PRISM 架构的有效缩放。我们还将 PRISM 集成到 SGLang 中并证明了其有效性，从而展示了 PRISM 在实际场景中的可行性。

参考文献

Butler, B.、Yu, S.、Mazaheri, A. 与 Jannesari, A. Pipeinfer：利用异步流水线推测加速大语言模型推理。载于《国际高性能计算、网络、存储与分析会议论文集》，SC '24。IEEE Press，2024。ISBN 9798350352917。doi: 10.1109/SC41406. 2024.00046。URL https://doi.org/10.1109/SC41406.2024.00046。

Cai, T.、Li, Y.、Geng, Z.、Peng, H.、Lee, J. D.、Chen, D. 和 Dao, T. Medusa：采用多个解码头的简单大语言模型推理加速框架。载于 Salakhutdinov, R.、Kolter, Z.、Heller, K.、Weller, A.、Oliver, N.、Scarlett, J. 和 Berkenkamp, F.（编），第 *41* 届国际机器学习会议论文集，机器学习研究论文集第 *235* 卷，第 5209–5235 页。PMLR，2024 年 7 月 21–27 日。网址：https://proceedings.mlr.press/v235/cai24b.html。

Chen, J.、Feng, A.、Zhao, Z.、Garza, J.、Nurbek, G.、Qin, C.、Maatouk, A.、Tassiulas, L.、Gao, Y. 和 Ying, R. MTBench：用于时间推理和问答的多模态时间序列基准测试，2025 年。网址：https://arxiv.org/abs/2503.16858。

Chen, M., Tworek, J., Jun, H.,Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavar- ian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert- Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saun- ders, W., Hesse, C., Carr, A. N., Leike, J., Achiam,

J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. 对基于代码训练的大语言模型的评估，2021。网址 https://arxiv.org/abs/2107.03374。

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C. 和 Schulman, J. 训练验证器以解决数学文字问题，2021 年。链接https://arxiv.org/abs/2110.14168。

Ding, N., Chen, Y., Xu, B., Qin, Y., Zheng, Z., Hu, S., Liu, Z., Sun, M. 和 Zhou, B. 通过扩展高质量指令对话来增强聊天语言模型，2023 年。链接 https://arxiv.org/abs/2305.14233。

Google, G. T. Gemini：一系列能力强大的多模态模型。*arXiv* 预印本 *arXiv:2312.11805*，2023 年。

Guha, E., Marten, R., Keh, S., Raoof, N., Smyrnis, G., Bansal, H., Nezhurina, M., Mercat, J., Vu, T., Sprague, Z., Suvarna, A., Feuer, B., Chen, L., Khan, Z., Frankel, E., Grover, S., Choi, C., Muennighoff, N., Su, S., Zhao, W., Yang, J., Pimpalgaonkar, S., Sharma, K., Ji, C.C.-J., Deng, Y., Pratt, S., Ramanujan, V., Saad-Falcon, J., Li, J., Dave, A., Albalak, A., Arora, K., Wulfe, B., Hegde, C., Durrett, G., Oh, S., Bansal, M., Gabriel, S., Grover, A., Chang, K.-W., Shankar, V., Gokaslan, A., Merrill, M. A., Hashimoto, T., Choi, Y., Jitsev, J., Heckel, R., Sathiamoorthy, M., Dimakis, A. G., and Schmidt, L. OpenThoughts：推理模型的数据配方，2025。URL https://arxiv.org/abs/2506.04178。

He, J., Li, T., Feng, E., Du, D., Liu, Q., Liu, T., Xia, Y. 和 Chen, H. 历史押韵：使用 Rhymerl 加速大语言模型强化学习，2025 年。URL https://arxiv.org/abs/2508.18588。

Hu, S., Li, J., Xie, X., Lu, Z., Toh, K.-C. 和 Zhou, P. Griffin：用于更快推测解码的有效词元对齐，2025 年。URLhttps://arxiv.org/abs/2502.11018。

Huang, H., Yang, F., Liu, Z., Xu, Y., Li, J., Liu, Y., Yin, X., Li, D., Ren, P. 和 Barsoum, E. Jakiro：通过混合专家实现的解耦多头增强推测解码，2025 年。URL https://arxiv.org/abs/2502.06282。

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., De- vlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural Questions：用于问答研究的基准测试。计算语言学协会会刊计算语言学，7:452–466，2019。doi: 10.1162/tacl a-00276。网址 https://aclanthology.org/Q19-1026/。

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. 基于分页注意力的大语言模型服务高效内存管理。载于 第*29*届操作系统原理研讨会论文集（*SOSP '23*），第61–626页，2023。doi: 10.1145/ 3600006.3613165。网址 https://doi.org/10.1145/3600006.3613165。

Leviathan, Y.、Kalman, M. 和 Matias, Y. 通过推测解码实现 Transformer 的快速推理。载于 Krause, A.、Brunskill, E.、Cho, K.、Engelhardt, B.、Sabato, S. 和 Scarlett, J.（编），第 *40* 届国际机器学习会议论文集，第 202 卷，机器学习研究论文集，第 19274–19286 页。PMLR，2023 年 7 月 23–29 日。网址：
https://proceedings.mlr.press/v202/leviathan23a.html。

Li, Y.、Wei, F.、Zhang, C. 和 Zhang, H. Eagle：推测采样需要重新思考特征不确定性，2024a。网址：https://arxiv.org/abs/2401.15077。

Li, Y.、Wei, F.、Zhang, C. 和 Zhang, H. Eagle-2：利用动态草稿树加速语言模型推理，2024b。网址：https://arxiv.org/abs/2406.16858。

Li, Y., Wei, F., Zhang, C. 和 Zhang, H. Eagle-3：通过训练时测试扩大大语言模型推理加速规模，2025a。网址https://arxiv.org/abs/2503.01840。

Li, Z., Zheng, L., Zhong, Y., Liu, V., Sheng, Y., Jin, X., Huang, Y., Chen, Z., Zhang, H., Gonzalez, J. E. 和 Stoica, I. AlpaServe：通过模型并行实现深度学习服务的统计复用。发表于第 *17* 届 *USENIX* 操作系统设计与实现研讨会（*OSDI 23*），第 663–679 页，美国马萨诸塞州波士顿，2023年7月。USENIX 协会。ISBN 978-1-939133-34-2。网址
https://www.usenix.org/conference/osdi23/presentation/li-zhouhan。

Li, Z., Chen, Z., Delacourt, R., Oliaro, G., Wang, Z., Chen, Q., Lin, S., Yang, A., Zhang, Z., Chen, Z., Lai, S., Cheng, X., Miao, X. 和 Jia, Z. AdaServe：通过 SLO 定制化推测解码加速多 SLO 大语言模型服务，2025b。网址 https://arxiv.org/abs/2501.12162。

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., Shi,C., Chen, Z., Arfeen, D., Abhyankar,R.，以及

Jia, Z. Specinfer：通过基于树的推测推理与验证加速大语言模型服务。载于 第 *29* 届 *ACM* 编程语言与操作系统体系结构支持国际会议论文集，第 *3* 卷，ASPLOS '24，第 932–949 页，纽约，NY，美国，2024 年。计算机协会。ISBN 9798400703867。doi: 10.1145/3620666.3651335。URLhttps://doi.org/10.1145/3620666.3651335。

OpenAI。《Introducing ChatGPT》。https://openai.com/index/chatgpt/，2022年11月。

Paszke, A., Gross, S., Massa, F., Lerer,A., Bradbury, J.,Chanan, G., Killeen, T., Lin,Z.,Gimelshein, N., Antiga,L.,Desmaison, A., K¨opf, A., Yang, E., DeVito, Z., Rai- son, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch：一种命令式风格、高性能的深度学习库，2019。链接：https://arxiv.org/abs/1912.01703。

Patel, P., Choukse, E., Zhang, C., Shah, A., Goiri, I. n., Maleki, S., and Bianchini, R. Splitwise：通过阶段拆分实现高效生成式大语言模型推理。载于第*51*届年度国际计算机架构研讨会论文集，ISCA '24，第118–132页。IEEE Press，2025。ISBN 9798350326581。doi：10.1109/ISCA59077.2024.00019。链接：https://doi.org/10.1109/ISCA59077.2024.00019。

Qin, R.、Li, Z.、He, W.、Cui, J.、Ren, F.、Zhang, M.、Wu, Y.、Zheng, W.和Xu, X. Mooncake：以更多存储换取更少计算——一种以KVCache为中心的大语言模型聊天机器人服务架构。见第*23*届*USENIX*文件与存储技术会议（*FAST 25*），第155–170页，美国加利福尼亚州圣克拉拉，2025年2月。USENIX 协会。ISBN 978-1-939133-45-8。URL
h
t
tps://www.usenix.org/conference/fast25/presentation/qin。

See, A.、Liu, P. J.和Manning, C. D. 直击要点：基于指针生成网络的摘要。见计算语言学协会第*55*届年会论文集（第*1*卷：长论文），第1073–1083页，加拿大温哥华，2017年7月。计算语言学协会。doi: 10.18653/ v1/P17-1099。
U
R
Lhttps://www.aclweb.org/anthology/P17-1099。

Tang, B., Fu, C. C., Kou, F., Sizov, G., Zhang, H., Park, J., Liu, J., You, J., Yang, Q., Mehta, S., Cai, S., Wang, X., Liu, X., Li, Y., Zhou, Y., Wei, W., Zhao, Z., Qi, Z., Victoria, A., Ibrahim, A., Wasti, B., Kim, C., Haz- iza, D., Sun, F., Delfin, G., Guo, E., Ouyang, J., Lee, J., Huang, J., Reizenstein, J., Fang, L., Zhu, Q., Verma, R., Mihailescu, V., Guo, X., Cui, Y., Hu, Y., and Lee,Y.

面向大规模 LLaMA 的高效推测解码：挑战与解决方案，2025。网址 https://arxiv.org/abs/2508.08192。

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P. 和 Hashimoto, T. B. 斯坦福 Alpaca：一个指令跟随型 Llama 模型。https://github.com/tatsu-lab/stanford_alpaca，2023。

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozi`ere, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. 和 Lam- ple, G. LLaMA：开放且高效的基础语言模型。*arXiv* 预印本 *arXiv:2302.13971*，2023。

Yan, S., Zhu, M., qing Jiang, G., Wang, J., Chen, J., Zhang, W., Liao, X., Cui, X., Zhang, C., Song, Z. 和 Zhu, R. 推测解码缩放定律，2025。网址 https://arxiv.org/abs/2505.07858。

Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.- G. Orca：一种用于基于 Transformer 的生成式模型的分布式服务系统。发表于 第 *16* 届 *USENIX* 操作系统设计与实现研讨会（*OSDI 22*），第 521–538 页，美国加利福尼亚州卡尔斯巴德，2022 年 7 月。USENIX 协会。ISBN 978-1-939133-28-1。网址：
h
ttps://www.usenix.org/conference/osdi22/presentation/yu。

Zhang, L., Wang, X., Huang, Y., and Xu, R. 面向推测采样的协调表征学习。发表于国际学习表征会议，2025a。

Zhang, Z., Jiang, Z., Jiang, C., Yu, M., Zheng, S., Lin, H., Hoffmann, H., and Liu, X. Swiftspec：通过扩展异步推测解码实现超低延迟大语言模型解码，2025b。网址：https://arxiv.org/abs/2506.11309。

Zhong, Y., Liu, S., Chen, J., Hu, J., Zhu, Y., Liu, X., Ji n, X., and Zhang, H. Distserve：通过拆分预填充和解码，实现吞吐量优化的大语言模型服务。载于第*18*届 *USENIX* 操作系统设计与实现会议论文集，OSDI'24，美国，2024a。USENIX 协会。ISBN 978-1-939133-40-3。

Zhong, Y., Yang, Z., Li, Z., Chen, R., Yu, Z., Zheng, L., Ho, Q., Gonzalez, J. E., Stoica, I., Zhang, H., and Jin, X. TetriInfer：弹性冷存储 GPU 推理服务案例。载于第*21*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 24*），第 907–925 页，美国加利福尼亚州圣克拉拉，2024年7月。USENIX 协会。网址：
h
t
t
p
s://
www.usenix.org/conference/osdi24/presentation/zhong-yinmin。