FLOWCAST：面向零成本推测式流匹配的轨迹预测

Divya Jyoti Bajpai[1]*, **Shubham Agarwal**[2]†,**Apoorv Saxena**[3]†,**Kuldeep Kulkarni**[4],**Subrata Mitra**[4], **Manjesh K. Hanawal**[1]
[1]印度理工学院孟买分校[2]加州大学伯克利分校[3]Inception Labs[4]Adobe Research India

arXiv:2602.01329v1 [cs.CV] 1 Feb 2026

摘要

流匹配（FM）近来已成为一种高质量视觉生成的强大方法。然而，由于去噪步骤数量众多，其推理速度慢得难以接受，限制了它在实时或交互式应用中的潜力。现有加速方法，如蒸馏、截断或一致性训练，要么会降低质量，要么需要耗费大量成本重新训练，要么泛化能力不足。我们提出 FlowCast，这是一种无需训练的推测生成框架，通过利用流匹配模型经过训练后能够保持恒定速度这一特性来加速推理。FlowCast 通过外推当前速度来预测未来速度，无需额外耗时；若预测误差在均方误差阈值以内，则予以接受。这种恒定速度预测使得稳定区域中的冗余步骤可以大幅跳过，同时在复杂区域保持精度。
FlowCast 是一个即插即用的框架，可无缝集成到任意流匹配模型中，且无需辅助网络。我们还给出了理论分析，并界定了推测轨迹与完整流匹配轨迹之间的最坏情况偏差。实证评估表明，FlowCast 在图像生成、视频生成和编辑任务中实现了 $> 2.5\times$加速比，优于现有基线，且与标准完整生成相比没有质量损失。

1 引言

流匹配（FM）Lipman 等（2022）提出的流匹配（FM）近来已成为一种强大且有理论依据的生成建模方法 Bond-Taylor 等（2021）。通过学习一个随时间变化的速度场，将基础分布中的样本连续变换为目标分布，流匹配模型通过求解**常微分方程（ODE）**来生成数据。与基于扩散的方法 Croitoru 等（2023）相比，该框架具有收敛更快、训练稳定以及能够直接控制样本轨迹等优势。

然而，尽管具备这些优势，**流匹配模型的推理成本高昂，已成为瓶颈**Kornilov 等（2024）。ODE积分本质上是顺序执行的——每一步都依赖前一步的输出——因此过程缓慢且难以并行化。要实现高保真生成，需要大量细粒度的步数，这会进一步增加延迟成本，使其难以用于实时或交互式场景。随着数据维度和模型复杂度的增加，推理负担也随之加重。在视频生成 Kong 等（2024）中，这一问题尤为突出，因为模型需要处理高维时空数据。顺序执行的 ODE积分会在各帧间累积，因此，对于长视频而言，高效推理至关重要 Liu 等（2025a）。

为降低推理延迟，人们提出了蒸馏Luhman & Luhman (2021); Yan 等（2024）; Kornilov 等（2024）、轨迹截断 Dhariwal & Nichol (2021); Lu 等人 (2022); Liu 等（2025a），以及一致性训练Yang 等人 (2024); Zhang & Zhou (2025); Dao 等人 (2025) 等静态缩减策略。然而，这些方法往往会降低输出

*本工作是在 Adobe Research India 实习期间完成的。
†同等贡献，按字母顺序排列。

质量，导致纹理模糊、语义漂移或空间连贯性受损。在视频领域，这类方法往往会造成时间闪烁和运动不一致，尽管单帧在视觉上看似合理，整体真实感仍会严重下降。此外，许多此类方法需要通过辅助目标或手工设计的损失函数重新训练模型，这不仅增加了工程负担，也限制了其在不同模型系列间的迁移能力。因此，这些方法的模块化程度较低，难以扩展，而且往往无法部署在对时间敏感或交互式的工作流中。

一种颇具前景的语言模型推理加速方法是**推测解码（SD）**Xia 等人（2024）：并行提出多个草稿，再有选择地进行验证，从而降低延迟。然而，这一思路无法直接扩展到流匹配（FM）等图像和视频生成模型。这些模型面临的一些独特挑战包括：(1) 与语言模型不同，目前没有现成可用的草稿模型。那么，如何设计一种轨迹与完整模型紧密耦合的草稿模型？(2) 流匹配（FM）的性能对速度输出高度敏感。速度上的微小扰动也可能沿积分路径不断累积，导致严重的生成错误。那么，如何评估草稿模型的速度，并控制轨迹漂移？我们利用流匹配（FM）轨迹的特性来应对这些挑战。
我们的关键洞见。我们发现，流匹配（FM）轨迹通常局部平滑且变化缓慢Lipman 等（2022），尤其是在训练良好的模型中。利用这一特性，我们提出了推测式框架 FlowCast，让模型自身过去的速度预测充当未来步数的零成本草稿。我们通过均方误差检查有选择地验证这些草稿：如果误差在阈值范围内，就完全跳过重新计算；否则，退回主干网络。该机制带来的额外开销可以忽略不计，无需重新训练或辅助模型，并且与现有流匹配流程完全兼容。

重要性。 与以往需要重新训练或牺牲生成保真度的加速技术不同，我们的方法即插即用，能够适应轨迹的复杂程度，并利用流匹配固有的平滑性。在动力学稳定的区域，它会积极跳过步数以加快推理；而在复杂区域，则会退回到细粒度积分以保证准确性。这种设计在**不影响输出质量**的同时实现了显著加速。该方法的一项关键优势是完全无需训练，且可直接应用于现有方法，从而进一步提升当前加速策略的效果。

FlowCast 与具体任务无关，可无缝应用于图像生成、图像编辑和视频生成，并确保编辑结果忠实且高质量。这种广泛的适用性凸显了推测生成作为加速生成模型的统一框架的价值。我们比较 FlowCast 的轨迹与完整常微分方程所得轨迹，据此推导出二者偏差的理论界。由此得到的误差界保证了我们方法的稳定性和保真度，确保推测生成在加快推理的同时，仍接近原始流匹配解。
总之，我们的主要贡献如下：

• 我们提出了一种推测式生成框架，即 FlowCast，用于基于流匹配的生成模型，从而实现自适应的部分并行推理。
• FlowCast 使用来自模型自身速度预测的零成本草稿，从而避免重新训练并保持轨迹保真度。

• 我们推导了一个理论误差界，形式化地刻画推测式轨迹相对于完整ODE积分的偏离程度。

• 我们的方法在保持样本质量的同时实现了显著的推理加速（$> 2.5\times$），优于现有基线，尤其是在多轮编辑任务中。

2 相关工作

流匹配 Lipman 等（2022）；Dao 等人（2023）；Labs等人（2025）；Deng等人（2025）近来已成为扩散模型 Croitoru 等（2023）；Xing等人（2024）；Yang 等人（2023）；Zhu等人（2023）的一种有力替代方案，能够在噪声与数据之间建立确定性映射。这种确定性使基于流的模型适用于图像反演 Deng等人（2024）、编辑 Wang等人（2024）和视频生成 Kong 等（2024）；Wan等人（2025）等应用，因为它能减少随机性，并以更少的神经网络函数评估实现更快的采样。

诸如 FlowTok He 等人（2025）这样的多模态扩展，将文本和图像压缩到共享的词元空间，以加快推理速度；FLUX.1 Labs等人（2025）则表明，大规模流可以以更低的成本媲美扩散模型。对于视频，金字塔流匹配 Jin等人（2024）通过分层生成降低了计算开销。

然而，迭代采样仍是这些模型的一大瓶颈，限制了它们在实时和交互式场景中的部署。为加速推理，先前的工作主要关注基于重训练的解决方案 Lee 等人（2023）；Bartosh 等人（2024）。蒸馏方法 Song 等人（2023）；Luhman & Luhman（2021）；Liu 等人（2022）；Kornilov 等（2024）；Salimans & Ho（2022），例如 InstaFlow Liu 等（2023）、LeDiFlow Zwick 等人（2025）和 Diff2Flow Schusterbauer 等人（2025），将扩散先验中的知识迁移到单步或少步采样中；而 PeRFlow Yan 等（2024）则通过分段整流流拉直轨迹。用于采样修改的方法 Dhariwal & Nichol（2021）；Lu 等人（2022）；Shaul 等人（2023），例如 TeaCache Liu 等（2025a），通过利用时间步嵌入跳过冗余步数；基于一致性的方法 Yang 等人（2024）；Zhang & Zhou（2025）；Haber 等人（2025）；Dao 等人（2025）则结合对抗损失与一致性损失，实现高质量的少步生成。

我们的工作利用推测解码框架加速推理过程，使流匹配（FM）更适用于实时和交互式应用。与现有方法不同，我们的方法无需重新训练、蒸馏、轨迹拉直或跳步。我们的方法为现有流模型提供了完整的即插即用解决方案。它在推理时动态运行，通过推测未来步数并即时验证来实现加速。这种设计在大幅提升加速比的同时，保留了模型原有的生成质量。

3 设置

本节将介绍流匹配（FM）和推测解码（SD）框架的基础形式化定义。

3.1 推测解码框架

推测解码（SD）是一种可并行化的推理策略，旨在加速自回归生成，尤其是在大语言模型（LLM）中，同时无需重新训练或牺牲输出质量。该框架利用了这样一个观察：并行评估一串词元仅比评估单个词元稍慢，因此在解码过程中能够实现部分并行化。

令 $\mathcal{M}_p$ 表示目标模型（速度较慢但准确），$\mathcal{M}_q$ 表示草稿模型（速度较快但准确性较低）。在推测式解码（SD）中，目标是通过验证由 $\mathcal{M}_q$生成的推测式草稿，加速 $\mathcal{M}_p$ 的词元生成。验证速度可以快于生成速度，从而实现加速。给定一个包含 i 个词元的输入 $x_{1:i} = (x_1, x_2, \ldots\ldots, x_i)$，SD 分为两个阶段：

1）起草阶段：草稿模型 $\mathcal{M}_q$ 以自回归方式生成 K 个（推测式）词元：$(x_{i+1}^{\text{draft}}, x_{i+2}^{\text{draft}}, \ldots\ldots, x_{i+K}^{\text{draft}}) \sim \mathcal{M}_q(\cdot \mid x_{1:i})$。由于 $\mathcal{M}_q$规模较小且架构较浅，这一阶段速度很快。

2）验证阶段：完整模型 $\mathcal{M}_p$ 通过一次前向传播评估整个推测序列，计算：$\mathcal{M}_p(x_{i+k} \mid x_{1:i+k-1}^{\text{draft}})$，其中 $k = 1, \ldots\ldots, K$。将这些预测分布与草稿词元进行比较，以评估其有效性（例如，通过交叉熵或 top-k一致性）。令 j 为 $\mathcal{M}_p$的预测与草稿显著偏离的第一个索引。随后接受前缀 $x_{i+1}^{\text{draft}}, \ldots\ldots, x_{i+j-1}^{\text{draft}}$。接着，将接受的词元再次传给草稿模型，使其根据更新后的上下文生成新草稿。

这一推测验证循环会持续迭代，直至生成完整序列。该方法无需修改 $\mathcal{M}_p$，即可提升吞吐量；当草稿模型的输出与目标模型高度一致时，效果尤为显著。

在 $\mathcal{M}_q$ 与 $\mathcal{M}_p$充分对齐的假设下，推测解码过程在加速生成的同时保持输出质量。此外，该方法无需重新训练，并可应用于任意一对（$\mathcal{M}_q$，$\mathcal{M}_p$），只要 $\mathcal{M}_q$ 经过合理校准。

将推测解码适配到图像生成并非易事，因为在连续的高维潜空间中起草有效的未来状态，远比语言模型中的下一词元预测更为复杂。草稿必须在预测的噪声或速度中保持时空一致性，而若没有经过训练的代理模型，这一点很难做到。此外，微小的轨迹误差会在常微分方程/随机微分方程积分过程中累积，导致语义漂移、结构畸变或与条件化失准。这些叠加效应使推测解码在视觉领域中较为脆弱，因此需要明确尊重底层动力学几何结构与稳定性的起草与验证策略。

3.2 流匹配概述

设 π_0 和 π_1 是定义在 $\mathbb{R}^d$上的两个光滑概率密度，分别表示源分布和目标分布。**流匹配（FM）**的目标是构造一个确定且连续的映射，通过由常微分方程（ODE）控制的时间索引轨迹，将样本从 π_0 传输到 π_1 。具体而言，FM 使用时变速度场 v：$\mathbb{R}^d \times [0,1] \to \mathbb{R}^d$定义变换，从而得到以下初值问题：

$$\frac{dx_t}{dt} = v(x_t, t), \quad x_0 \sim \pi_0, \quad t \in [0,1]. \tag{1}$$

其中，$x_t \in \mathbb{R}^d$ 表示样本在时间 t的状态，而 $v(x_t, t)$通常由一个神经网络参数化，该网络经过训练，使 $t = 1$处的边缘密度（记为 π_1）与目标分布对齐。该常微分方程定义了流映射 $\Phi_t(x_0)$，使点沿连续轨迹在 $\mathbb{R}^d$ 中移动，其中$\Phi_1(x_0) \sim \pi_1$。学习一个能够实现这种传输的有效 $v(x_t, t)$，是流匹配的核心目标。

数值积分：在实践中，常微分方程的解 x_t 必须通过数值计算得到，因为对于学习得到的速度场，闭式解通常不可得。在流匹配模型中，求解常微分方程最广泛使用的方法是**前向欧拉法**，因其简单且计算高效。时间区间 $[0,1]$ 被离散化为 K个点的序列 $\{t_0 = 0, t_1, \ldots, t_K = 1\}$，这些点的间距可以是非均匀的。从 $x_0 \sim \pi_0$开始，轨迹按如下方式迭代更新

$$x_{k+1} = x_k + \Delta t_k \cdot v(x_k, t_k), \quad \Delta t_k = t_{k+1} - t_k. \tag{2}$$

这种离散化用有限步序列来近似连续流，每次更新都使状态沿速度场规定的方向前进。欧拉法因其较低的计算开销，在大多数实现中受到青睐Deng等人（2025年）；Labs等人（2025年）。

轨迹平滑性：基于欧拉法的离散化的有效性与精度取决于速度场 $v(x_t, t)$的正则性。特别是，当常微分方程存在唯一解时，离散化误差可以被界定。更重要的是，对已训练流匹配模型的经验观察表明，速度场往往在各时间步和样本位置上逐渐变化Liu 等（2025a），尤其是在使用插值函数或合成速度目标进行训练时，这表明速度在相邻步之间不会急剧波动。

4 流模型的推测式解码

本节将介绍我们的核心贡献：建立推测式解码与流匹配模型之间的联系，提出我们的推测生成框架，并提供理论误差界，以形式化地刻画其保真度。

FM 的自回归特性： 推测解码适用于自回归模型。流匹配过程本质上具有自回归特性，因为每个去噪步骤都完全依赖于前一步的输出，从而形成严格的顺序依赖链——这与语言模型逐个生成词元的方式非常相似。不过，虽然推测解码用于自回归发生在离散空间中的语言任务（例如，来自离散集合的词元），但它也可以应用于流匹配模型，其中自回归发生在连续空间中（例如，有界区间内的速度）。

零成本草稿构建：为了在流匹配模型中实现推测解码而无需依赖独立的草稿网络，我们提出一种轻量且自包含的草稿生成策略，直接复用目标模型自身的速度预测。

回顾第 3.2 节，流匹配模型学习一个时变速度场 $v(x_t,\ t)$，通过对常微分方程进行积分，将样本从源分布 π_0 输运至目标分布 π_1：

$$\frac{dx_t}{dt} = v(x_t, t), \quad x_0 \sim \pi_0. \tag{3}$$

在训练过程中，模型 v 通常会被监督，以匹配沿插值路径的参考速度（通常随时间保持不变）。这促使模型 v 在空间和时间上都平滑地变化。利用这种平滑性，我们提出了 FlowCast，其中保持速度不变，以外推不仅是紧接着的下一个状态，还包括剩余的完整轨迹，从而生成零成本的草稿轨迹。轨迹的每一步都会由完整模型并行验证，并据此接受这些步骤或进行校正。FlowCast 包含三个组件：起草、验证和校正。其伪代码见算法 1。

算法 1 FlowCast

要求： 初始状态 x_{t_0}、时间步 $\{t_k\}_{k=0}^{K}$、阈值 ϵ、模型 v

1: 计算初始速度：$v_{t_0} \leftarrow v(x_{t_0},\ t_0)$ 2: 预测下一状态：$x_{t_1} \leftarrow x_{t_0} + (t_1 - t_0)v_{t_0}$ 3: 设置当前索引 $m \leftarrow 0$

4: **重复**

5: **(Drafting)**：生成推测状态：

$$\tilde{x}_{t_{k+1}} \leftarrow x_{t_m} + (t_k - t_m)v_{t_m} \ \forall k = m+1, \ldots, K$$

6: **(并行验证)**：对于每个 $k = m+1, \ldots, K$，计算：

$$v_{t_k} \leftarrow v(\tilde{x}_{t_k}, t_k), \quad e_k \leftarrow \text{MSE}(v_{t_m}, v_{t_k})$$

7: 找到第一个索引 $j > m$，使得 $e_j > \epsilon$

8: **如果**没有拒绝（即，对所有 k，$e_k \le \epsilon$）**则**

9: 接受完整草稿 $\{\tilde{x}_{t_k}\}_{k=m+1}^{K}$，设置 $m \leftarrow K$

10: **else**

11: **(校正)**：接受草稿 $\{\tilde{x}_{t_k}\}_{k=m+1}^{j-1}$

12: $x_{t_j} \leftarrow \tilde{x}_{t_{j-1}} + v_{t_{j-1}}(t_j - t_{j-1})$; $m \leftarrow j-1$

13: 重新计算速度：$v_{t_m} \leftarrow v(x_{t_m},\ t_m)$

14: **结束 if**

15: **直到** $m = K$

起草： 给定步骤 t_i 处的当前状态 x_{t_i}，将推测式轨迹定义为：

$$\tilde{x}_{t_i+\Delta t_k} = x_{t_i} + v(x_{t_i}, t_i) \cdot \Delta t_k \tag{4}$$

对于 $k = 1, \ldots, n$ 和 $k > i$。此处 n 表示完成轨迹所需遍历的剩余时间步数，而 $\Delta t_k = t_k - t_i$。这实际上重复使用同一个速度向量，对剩余路径进行线性外推。

该方法在起草过程中无需引入辅助模型，也不会增加额外的计算成本。尽管方法简单，但它利用了流匹配模型的归纳偏置来生成连贯的轨迹，尤其适用于速度演化平滑的领域。

验证： 我们采用一种轻量且有理论依据的机制，来验证每个起草状态的有效性。对于每个外推点 $\tilde{x}_{t+\Delta t}$，我们将复用的速度 $v(x_t,\ t)$ 与模型在该外推点和步长处自身预测的速度进行比较，即 $v(\tilde{x}_{t+\Delta t},\ t+\Delta t)$。如果两者之间的均方误差（MSE）保持低于预定义阈值 ϵ，则接受该草稿，即

$$\text{MSE}(v(x_t, t),\, v(\tilde{x}_{t+\Delta t}, t+\Delta t)) < \epsilon. \tag{5}$$

校正： 如果在某一步违反了该标准，则拒绝该草稿，并丢弃后续所有草稿。随后，从发生拒绝的时间步重新启动一条新轨迹，并按照公式 4 使用模型新生成的速度预测。

这种验证策略效率高，并且与流匹配的公式直接一致，其中速度场编码了生成动力学。不同于在数据空间或潜在空间中评估有效性的传统方法，我们基于速度的判据在函数空间中运作，因此能更敏锐地捕捉局部动力学中的不一致，同时计算成本低廉。这确保积分路径忠实于学习到的流，而无需进行不必要的重新计算。

FlowCast 的工作流程如图1所示。左侧展示了常规生成过程，其中流匹配模型在轨迹的每一步依次调用。右侧展示了 FlowCast 对通过草拟步骤生成的轨迹进行处理的过程。轨迹中所有推测式生成的草稿都会并行输入流匹配模型进行处理。在常规流匹配生成过程中，模型在每一步的输出仅取决于前一步的输出。我们利用这一特性来验证草稿是否正确。具体而言，如果流匹配模型在步骤 k 的输出与步骤 $k+1$ 的推测式草稿在均方误差阈值内相匹配，则接受步骤 $k+1$ 的推测式草稿，否则拒绝。此验证通过一次

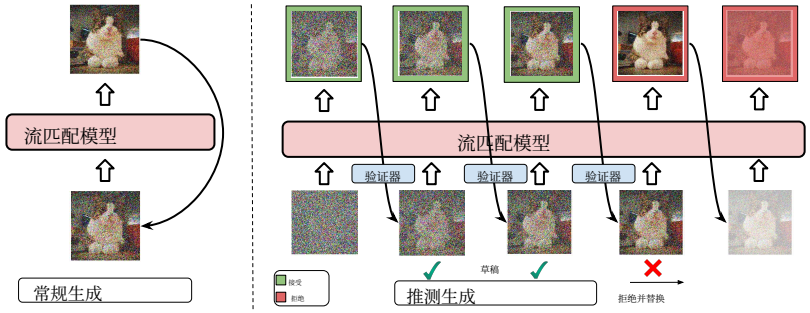


图1：常规图像生成与推测式图像生成的对比：左侧展示了流匹配模型中的常规迭代生成过程。右侧展示了我们的方法：引入中间推测草稿，由主干网络并行快速提出并验证，从而在保持质量的同时加快采样。

前向传播即可高效完成。在图1中，草稿一直被接受至步骤 3，而步骤 4 的草稿则被拒绝。一旦某一步的草稿被拒绝，之后所有草稿都会被丢弃，并从最后一个已接受的步骤开始计算轨迹。

4.1 分析

在本小节中，我们界定草稿轨迹与完整模型轨迹之间的偏差。随后，我们给出均方误差阈值的界，以保证最坏情况下的偏差。

引理*4.1*。令 $x(t)$ 为 $\frac{dx}{dt} = v(x,t)$ $x(0) = x_0$ 的解，其中 v 关于 x 是利普希茨连续的，常数为 M。假设：*1)* 对所有 $t \in [0,1]$，$\|x''(t)\| \le N$；*2)* 对所有 (x,t)，$\|\frac{\partial v}{\partial x}(x,t)\| \le M$；*3)* 使用步长为 h 的前向欧拉格式运行 k 步，其中推测速度 $\tilde{v}$ 在比例为 $p \in [0,1]$ 的步数中被接受，且满足 $\|\tilde{v}(x_k,t_k) - v(x_k,t_k)\| \le \sqrt{\epsilon}$.。则全局离散误差 $E_k = \|x(t_k) - x_k\|$ 可界定为

$$\|x(t_k) - x_k\| \;\le\; \frac{e^{Mt_k} - 1}{2M}\left(hN + 2p\sqrt{\epsilon}\right).$$

这里，第一项 $\frac{e^{Mt_k}-1}{2M}hN$ 表示标准的欧拉离散化误差，而第二项 $\frac{e^{Mt_k}-1}{M}p\sqrt{\epsilon}$ 量化了推测更新引入的额外偏差。该引理的证明见附录A.1。由此引理可得下述定理。

定理4.2. 设 $q_d > 0$ 表示用户指定的容差，即相对于完整（非推测式）生成轨迹所允许的最大偏差。若要使推测误差 $\|E_k^s\|$ （即推测生成引入的偏差）保持在 q_d以内，只需选择满足以下条件的推测阈值 ϵ：

$$\epsilon \;\le\; \left(\frac{q_d}{2A}\right)^2, \qquad A = \frac{e^M - 1}{M}.$$

利用这一结果，我们可以通过设置适当的 ϵ，从形式上控制推测轨迹与完整轨迹之间的最坏情况偏差。定理4.2提供了一种有理论依据的方法来调整 ϵ，确保推测步数导致的轨迹偏差不超过质量预算。定理4.2的证明见附录A.2。

5 实验

下面我们将详细介绍实验设置和结果。首先介绍数据集。

- **文本到图像生成**：我们使用**GenEval**数据集 Ghosh等人（2023），该数据集包含553条针对组合推理的提示词，涵盖评估物体共现、空间位置、颜色绑定和物体数量的类别。

方法	BAGEL			FLUX		
	总体 ↑	CLIPIQA ↑	加速比 ↑	总体 ↑	CLIPIQA ↑	加速比 ↑
完整模型						
50 步数	**0.78**	**0.84**	1.00×	**0.65**	**0.83**	1.0×
25步	0.77	0.82	2.0×	0.64	0.80	2.00×
10步	0.73	0.75	5.0×	0.57	0.59	5.00×
5步	0.57	0.40	10.0×	0.44	0.43	10.0×
静态基线						
InstaFlow	0.33	0.70	50.0×	–	–	–
PerFlow	0.58	0.79	5.0×	–	–	–
TeaCache	0.75	0.80	1.8×	0.64	0.81	1.84×
我们的方法：推测解码						
FlowCast-50	**0.78**	0.83	2.5×	**0.65**	**0.83**	2.4×
FlowCast-25	0.77	0.82	4.1×	0.64	0.80	4.2×
FlowCast-10	0.73	0.74	7.8×	0.57	0.60	7.6×
FlowCast-5	0.57	0.38	13.0×	0.43	0.45	12.8×

表 1：**BAGEL 和 FLUX 模型在图像生成上的比较**。我们报告总体 GenEval、CLIPIQA 图像质量分数，以及相对于 50 步完整生成的加速比。

• **图像编辑**：我们采用**GEdit** 基准 Liu等人（2025b），该基准提供606条真实世界编辑指令，涵盖物体操作、颜色修改等。

• **多轮编辑**：我们进一步基于 EditBench 构建了一个辅助数据集，其中 GPT-4.1 Achiam 等（2023）生成三次渐进式编辑，随后进行三次反向编辑以还原原始图像。通过最终输出与原始图像之间的峰值信噪比（PSNR）量化重建保真度。

• **视频生成**：我们在**Vbench**数据集 Huang 等（2024）上进行评估，从 16 个评估维度中分别均匀抽取 5 个提示，共采样 80 个提示，以确保对运动、持续性、摄像机控制和场景构图的均衡覆盖。

基线： 我们将所提出的方法与以下基线进行比较：

完整生成。 这是标准的采样流程，模型会完整执行去噪轨迹，不进行任何加速。它提供了保真度方面的上界，并作为所有加速方法的参考基准。

TeaCache。 Liu 等（2025a）通过缓存中间表示并在不同时间步之间重复使用，且以时间步嵌入为指导，从而加快生成速度。通过省去计算，它实现了更快的推理，但重复使用近似状态会导致保真度下降。

InstaFlow。Liu 等（2023）提出了一种基于流匹配的采样器，经过训练后可在 Stable-Diffusion-v-1.5 模型中实现极快生成（最快仅需一个步数）。虽然效率很高，但与完整采样相比，它牺牲了保真度。我们在已发布的 Stable-Diffusion-v1.5 权重上评估 InstaFlow。

PeRFlow（分段整流流）。 PeRFlow Yan 等（2024）通过对分段时间步进行分段线性整流，使扩散轨迹变得更直，从而能够在 Stable-Diffusion-xl 模型上实现少步生成，并取得出色的质量与效率权衡。我们在已发布的 Stable-Diffusion-xl 检查点上评估 PeRFlow。

FlowCast（我们的方法）。我们的方法通过让轻量级草稿过程提出多个未来步数，再由完整模型并行验证这些步数，从而加速推理。我们将其记为 *FlowCast-*n，其中 n 表示推测步数。

我们的方法与 PeRFlow 和 TeaCache 等现有加速技术互为补充，即使在已针对效率进行优化的框架中，也能进一步减少冗余步数。相应结果见表 4 和表 5。

对于所有基线方法，我们都采用其代码库中提供的官方超参数。在表 1 中，TeaCache 按发布版本使用；在图 2 中，我们还在不同时间步下对其进行评估，以

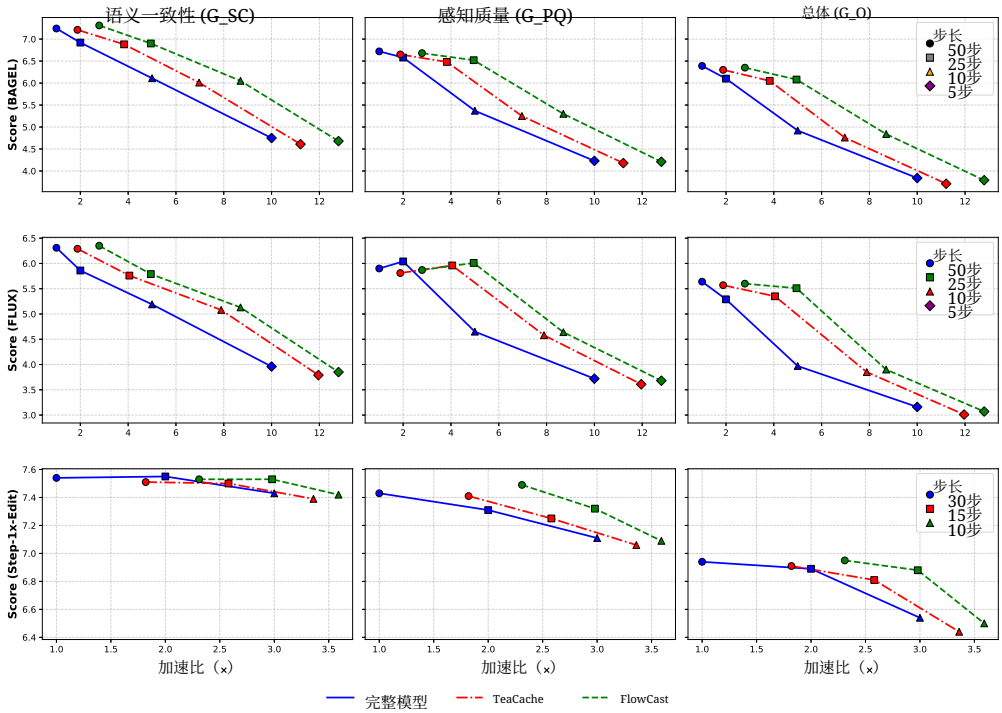


图 2：三个模型（BAGEL、FLUX 和 Step-Edit）的 GEdit 分数。每个子图报告了语义一致性（G SC）、感知质量（G PQ）和总分（G O）与加速比之间的关系。

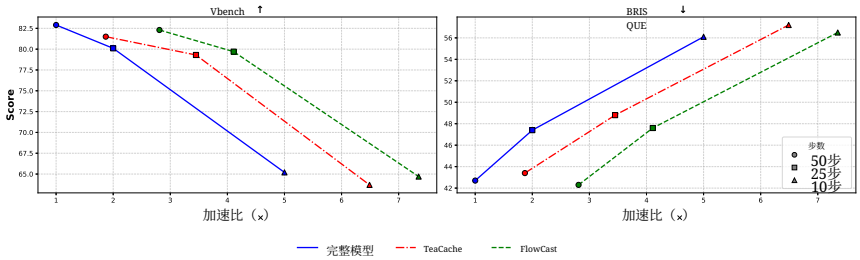


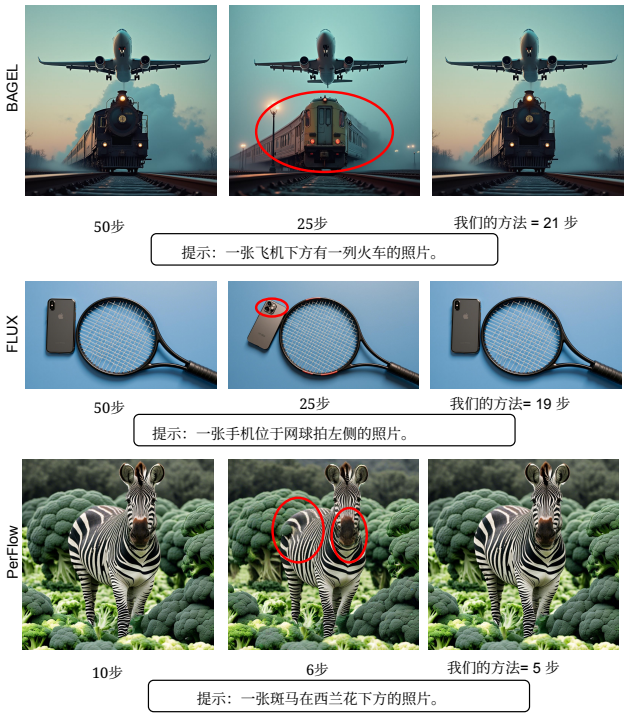
图3：在混元视频模型上的结果。我们报告 Vbench 分数，并特别使用 BRISQUE 指标评估单帧的质量。

强调其即插即用的灵活性。对于 FlowCast 中 ϵ 的选择，根据定理4.2，我们发现，图像生成（所有模型）采用 $\epsilon \in [0.01, 0.02]$、图像编辑（所有模型）采用 $\epsilon \in [0.07, 0.08]$、视频生成采用 $\epsilon \in [0.001, 0.002]$ 时，都能在性能与完整模型相近的情况下实现较高的加速比。附录 8 中给出了针对 ϵ 在多个步数下的消融研究。

模型。 为展示我们方法的通用性，我们在多个最先进模型上进行了评估：图像生成方面使用 BAGEL（Deng等人（2025））、Flux-Kontext（Labs等人（2025））和 PeRFlow（Yan 等（2024））；图像编辑方面使用 BAGEL、Flux-Kontext 和 Step-1X-Edit（Liu等人（2025b））；视频生成方面则使用混元视频（Kong 等（2024））。

5.1 结果

图像生成： 表 1 报告了 BAGEL 与 FLUX 在 GenEval 基准上的结果，表 4 则展示了对优化后的 PerFlow 模型的额外评估。GenEval 分数捕捉语义维度（详见第 2、3 表），而保真度通过 CLIPIQA 评估。静态



Figu图 4：推测步数缩减在不同模型上均能保持图像质量，而直接缩减步
数则会导致明显的质量下降。

缩减方法（例如固定步数截断）会造成显著的质量损失，而我们的自适应方法实现了最低的质量下降，并保持了语义对齐。

图像编辑： 图 2 展示了使用 BAGEL、FLUX 和 Step-1X-Edit 进行的 GEdit 结果，并报告了 G SC（语义一致性）、G PQ（感知质量）和 G O_(总体可编辑性）。静态步数缩减会降低可编辑性和质量，而我们的自适应策略始终能达到与全步数生成相当的效果。多次迭代的结果见图 7，相关讨论见附录A.3。

视频生成： 在图3中，我们使用 Hunyuan-Video 模型将分析扩展到视频生成。我们使用 vBench 指标报告性能，该指标评估时间连贯性、背景稳定性、闪烁伪影等。我们还使用 BRISQUE 指标报告逐帧质量分数。静态缩减方法会引入明显的时间不一致和运动伪影，而 FlowCast 方案能够保持视频的高质量和时间稳定性。

定性分析： 我们对不同任务中的静态与自适应步数缩减进行了定性比较。在图像生成中（图4），静态缩减常会引入语义不一致和伪影，而我们的自适应方法只跳过冗余步数，生成的结果与完整步数生成的结果高度一致。在编辑任务中（图5、图11），静态缩减会导致编辑不完整或出现非预期结果，并加重残余噪声；而我们的方法能够保持语义准确性和视觉保真度，即使在多轮设置下也是如此。在视频生成中（图9），静态缩减会造成时间断裂和帧丢失，而 FlowCast 则能保持流畅度和保真度。

总结： 在图像生成、编辑、多轮迭代编辑和视频生成等多种任务中，现有加速技术始终会导致保真度和对齐度下降，凸显了其通用性有限。相比之下，FlowCast 是一种通用的自适应框架，可加速基于流的模型，并根据生成复杂度进行调整（见附录A.5）。与会造成性能下降的启发式截断不同，FlowCast 利用模型提供的信息进行验证，只跳过冗余更新。因此，输出在实现显著加速的同时，仍与全步数生成难以区分。这些结果表明，FlowCast 是跨模态实现高效、可靠且广泛可迁移的步数缩减的首选方案。

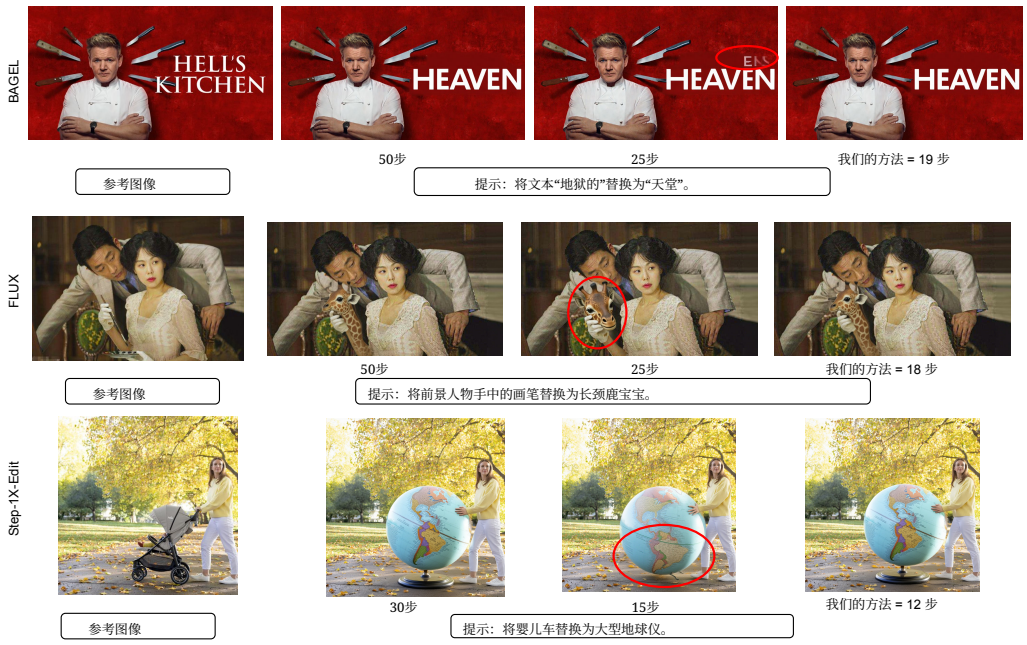


图5：与直接步数缩减相比，推测式步数缩减能够更好地保持编辑保真度和一致性。

6 结论

我们提出了一种自适应推理策略，通过消除真正冗余的计算来加速生成。我们的方法扩展了推测解码，利用先前的速度预测直接生成零成本草稿，再通过轻量级均方误差准则并行验证这些草稿。除了方法简单之外，我们还为偏离原始轨迹的程度建立了理论保证，从而确保其可靠性和严谨性。在多种模型和任务上进行的大量实验表明，我们的方法能够实现稳定加速，同时不牺牲性能，凸显了其作为通用框架实现更快、更可靠推理的潜力。
FlowCast 的主要局限在于依赖并行草稿评估，这需要充足的计算资源；减少草稿数量可以降低开销，但也会削弱加速效果。

7 伦理与可复现性声明

我们已阅读并遵守本次会议详细说明的伦理准则。据我们所知，本研究为原创成果，且已恰当引用所有相关工作。所有作者均为本研究做出了贡献，并对其内容负责。除这些模型已作为我们技术方法的一部分，或其使用情况已披露的情形外，我们未以任何未声明的方式使用大语言模型（LLMs）。
我们详细描述了复现本研究的算法。我们使用了公开可用的代码、模型和数据集。此外，我们计划提供用于复现本研究结果的源代码。相关工作的结果取自其官方论文和／或官方源代码仓库（每项工作均另行注明）。

参考文献

Josh Achiam、Steven Adler、Sandhini Agarwal、Lama Ahmad、Ilge Akkaya、Florencia Leoni Ale- 男人、DiogoAlmeida、JankoAltenschmidt、Sam Altman、Shyamal Anadkat 等。Gpt-4 技术报告。*arXiv* 预印本 *arXiv:2303.08774*，2023年。

Grigory Bartosh、Dmitry PVetrov 和 Christian Andersson Naesseth。神经流扩散模型：用于改进扩散建模的可学习前向过程。神经信息处理系统进展，37:73952–73985，2024年。

Sam Bond-Taylor、Adam Leach、Yang Long 和 Chris G Willcocks。深度生成建模：对变分自编码器、生成对抗网络、归一化流、基于能量的模型和自回归模型的比较综述。*IEEE*模式分析与机器智能汇刊，44(11):7327–7347，2021年。

Florinel-Alin Croitoru、Vlad Hondru、Radu Tudor Ionescu 和 Mubarak Shah。视觉中的扩散模型：综述。*IEEE*模式分析与机器智能汇刊，45(9): 10850–10869，2023年。

Quan Dao、Hao Phung、Binh Nguyen 和 Anh Tran。潜在空间中的流匹配。*arXiv* 预印本 *arXiv:2307.08698*，2023年。

Quan Dao、Hao Phung、Trung Tuan Dao、Dimitris N Metaxas 和 Anh Tran。用于一致单步和少步图像生成的自校正流蒸馏。载于*AAAI*人工智能大会论文集，第39卷，第2654–2662页，2025年。

Chaorui Deng、Deyao Zhu、Kunchang Li、Chenhui Gou、Feng Li、Zeyu Wang、Shu Zhong、Weihao Yu、Xiaonan Nie、Ziang Song 等。统一多模态预训练中的新兴特性。*arXiv* 预印本 *arXiv:2505.14683*，2025年。

Yingying Deng、Xiangyu He、Changwang Mei、Peisong Wang 和 Fan Tang。FireFlow：用于图像语义编辑的整流流快速反演。*arXiv* 预印本 *arXiv:2412.07517*，2024年。

Prafulla Dhariwal 和 AlexanderNichol。扩散模型在图像合成方面胜过生成对抗网络。神经信息处理系统进展，34：8780–8794，2021年。

Dhruba Ghosh、HannanehHajishirzi 和 Ludwig Schmidt。GenEval：用于评估文本到图像对齐的面向对象的框架。神经信息处理系统进展，36：52132–52152，2023年。

EldadHaber、ShadabAhamed、Md ShahriarRahimSiddiqui、Niloufar Zakariaei 和 Moshe Eliasof。用于增强生成模型的迭代流匹配——路径校正与渐进式细化。*arXiv* 预印本 *arXiv:2502.16445*，2025年。

Ju He、Qihang Yu、Qihao Liu 和 Liang-Chieh Chen。Flowtok：在文本与图像标记之间无缝流动。*arXiv* 预印本 *arXiv:2503.10772*，2025年。

Ziqi Huang、Yinan He、Jiashuo Yu、Fan Zhang、Chenyang Si、Yuming Jiang、Yuanhan Zhang、Tianx- ing Wu、Qingyang Jin、Nattapol Chanpaisit 等。Vbench：面向视频生成模型的综合基准测试套件。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 21807–21818 页，2024年。

Yang Jin、Zhicheng Sun、Ningyuan Li、Kun Xu、Hao Jiang、Nan Zhuang、Quzhe Huang、Yang Song、YadongMu 和 ZhouchenLin。用于高效视频生成建模的金字塔流匹配。*arXiv* 预印本 *arXiv:2410.05954*，2024年。

Weijie Kong、Qi Tian、Zijian Zhang、Rox Min、Zuozhuo Dai、Jin Zhou、Jiangfeng Xiong、Xin Li、Bo Wu、Jianwei Zhang 等。混元视频：大型视频生成模型的系统性框架。*arXiv* 预印本 *arXiv:2412.03603*，2024年。

Nikita Kornilov、Petr Mokrov、Alexander Gasnikov 和 Aleksandr Korotin。最优流匹配：仅用一步学习直线轨迹。神经信息处理系统进展，37：104180–104204，2024年。

Black Forest Labs、Stephen Batifol、AndreasBlattmann、FredericBoesel、Saksham Consul、Cyril Diagne、Tim Dockhorn、Jack English、Zion English、Patrick Esser 等。Flux. 1 kontext：用于潜在空间中上下文图像生成与编辑的流匹配。*arXiv* 预印本 *arXiv:2506.15742*，2025年。

Sangyun Lee、Beomsu Kim 和 Jong Chul Ye。最小化基于常微分方程的生成模型的轨迹曲率。发表于国际机器学习会议，第 18957–18973 页。PMLR，2023年。Yaron Lipman、Ricky TQ Chen、Heli Ben-Hamu、Maximilian Nickel 和 Matt Le。用于生成建模的流匹配。*arXiv* 预印本 *arXiv:2210.02747*，2022年。Feng Liu、Shiwei Zhang、Xiaofeng Wang、Yujie Wei、Haonan Qiu、Yuzhong Zhao、Yingya Zhang、Qixiang Ye 和 Fang Wan。时间步嵌入揭示：视频扩散模型到了缓存的时机。发表于计算机视觉与模式识别会议论文集，第 7353–7363 页，2025a。Shiyu Liu、Yucheng Han、Peng Xing、Fukun Yin、Rui Wang、Wei Cheng、Jiaqi Liao、Yingming Wang、Honghao Fu、Chunrui Han 等。Step1x-edit：通用图像编辑的实用框架。*arXiv* 预印本 *arXiv:2504.17761*，2025b。Xingchao Liu、Chengyue Gong 和 Qiang Liu。流动更直、更快速：学习使用整流流生成和迁移数据。*arXiv* 预印本 *arXiv:2209.03003*，2022年。Xingchao Liu、Xiwen Zhang、Jianzhu Ma、Jian Peng 等。InstaFlow：一步即可实现高质量的基于扩散的文本到图像生成。发表于第十二届国际学习表征会议，2023年。Cheng Lu、Yuhao Zhou、Fan Bao、Jianfei Chen、Chongxuan Li 和 Jun Zhu。DPM-Solver：一种快速常微分方程求解器，可在约10步内完成扩散概率模型采样。神经信息处理系统进展，35：5775–5787，2022年。Eric Luhman 和 Troy Luhman。迭代生成模型中的知识蒸馏：提升采样速度。*arXiv* 预印本 *arXiv:2101.02388*，2021年。Tim Salimans 和 Jonathan Ho。用于扩散模型快速采样的渐进式蒸馏。*arXiv* 预印本 *arXiv:2202.00512*，2022年。Johannes Schusterbauer、Ming Gui、Frank Fundel 和 Bj¨orn Ommer。Diff2Flow：通过扩散模型对齐训练流匹配模型。发表于计算机视觉与模式识别会议论文集，第 28347–28357 页，2025年。Neta Shaul、Juan Perez、Ricky TQ Chen、Ali Thabet、Albert Pumarola 和 Yaron Lipman。生成流模型的定制求解器。*arXiv* 预印本 *arXiv:2310.19075*，2023年。Yang Song、Prafulla Dhariwal、Mark Chen 和 Ilya Sutskever。一致性模型。2023年。Wan 团队、Ang Wang、Baole Ai、Bin Wen、Chaojie Mao、Chen-Wei Xie、Di Chen、Feiwu Yu、Haiming Zhao、Jianxiao Yang 等。Wan：开放且先进的大规模视频生成模型。*arXiv* 预印本 *arXiv:2503.20314*，2025年。Jiangshan Wang、Junfu Pu、Zhongang Qi、Jiayi Guo、Yue Ma、Nisha Huang、Yuxin Chen、Xiu Li 和 Ying Shan。驯服整流流以实现反演与编辑。*arXiv* 预印本 *arXiv:2411.04746*，2024年。Heming Xia、Zhe Yang、Qingxiu Dong、Peiyi Wang、Yongqi Li、Tao Ge、Tianyu Liu、Wenjie Li 和 Zhifang Sui。释放大语言模型推理效率：推测解码综合综述。*arXiv* 预印本 *arXiv:2401.07851*，2024年。Zhen Xing、Qijun Feng、Haoran Chen、Qi Dai、Han Hu、Hang Xu、Zuxuan Wu 和 Yu-Gang Jiang。视频扩散模型综述。*ACM* 计算调查，57(2)：1–42，2024年。Hanshu Yan、Xingchao Liu、Jiachun Pan、Jun Hao Liew、Qiang Liu 和 Jiashi Feng。PeRFlow：作为通用即插即用加速器的分段整流流。神经信息处理系统进展，37：78630–78652，2024年。Ling Yang、Zhilong Zhang、Yang Song、Shenda Hong、Runsheng Xu、Yue Zhao、Wentao Zhang、Bin Cui 和 Ming-Hsuan Yang。扩散模型：方法与应用综合综述。*ACM* 计算调查，56(4)：1–39，2023年。

Ling Yang、Zixiang Zhang、Zhilong Zhang、Xingchao Liu、Minkai Xu、Wentao Zhang、Chenlin Meng、Stefano Ermon 和 Bin Cui。一致性流匹配：通过速度一致性定义直线流。*arXiv* 预印本 *arXiv:2407.02398*，2024 年。

Yuchen Zhang 和 Jian Zhou。逆向流与一致性模型。载于第四十二届国际机器学习大会，2025年。

Yuanzhi Zhu、Zhaohai Li、Tianwei Wang、Mengchao He 和 Cong Yao。基于扩散模型的条件文本图像生成。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 14235–14245 页，2023年。

Pascal Zwick、Nils Friederich、Maximilian Beichter、Lennart Hilbert、Ralf Mikut 和 Oliver Bringmann。LeDiFlow：学习分布引导流匹配以加速图像生成。*arXiv* 预印本 *arXiv:2505.20723*，2025年。

A 附录

A.1 引理 4.1 的证明

证明。 定义步骤 $k+1$ 处的误差为

$$E_{k+1} := x(t_{k+1}) - x_{k+1}.$$

根据欧拉更新,

$$x_{k+1} = x_k + h\, v^{used}(x_k, t_k),$$

其中， v^{used} 为 v_θ，或在接受时为推测速度 $\tilde{v}$。

对 $x(t)$ 在 t_k 处进行泰勒展开，存在某个 $\psi \in (t_k,\ t_{k+1})$，使得

$$x(t_{k+1}) = x(t_k) + h\, x'(t_k) + \frac{h^2}{2} x''(\psi).$$

因此,

$$\begin{aligned}
\|E_{k+1}\| &= \left\| x(t_k) + h v_\theta(x(t_k), t_k) + \frac{h^2}{2} x''(\psi) - \big(x_k + h v^{used}(x_k, t_k)\big) \right\| \\
&= \left\| E_k + h\big(v_\theta(x(t_k), t_k) - v^{used}(x_k, t_k)\big) + \frac{h^2}{2} x''(\psi) \right\| \\
&\le \|E_k\| + h\, \|v_\theta(x(t_k), t_k) - v_\theta(x_k, t_k)\| + h\, \big\|v_\theta(x_k, t_k) - v^{used}(x_k, t_k)\big\| + \frac{h^2}{2} N.
\end{aligned}$$

根据 v_θ in x 的利普希茨连续性,

$$\|v_\theta(x(t_k), t_k) - v_\theta(x_k, t_k)\| \le M \|x(t_k) - x_k\| = M \|E_k\|.$$

此外，根据假设，在推测速度被接受的步数中,

$$\|v_\theta(x_k, t_k) - \tilde{v}(x_k, t_k)\| \le \sqrt{\epsilon}.$$

考虑到此类步数所占的比例为 p ,

$$\big\|v_\theta(x_k, t_k) - v^{used}(x_k, t_k)\big\| \le p\sqrt{\epsilon}.$$

综合以上内容,

$$\|E_{k+1}\| \le \|E_k\| + hM\|E_k\| + hp\sqrt{\epsilon} + \frac{h^2}{2} N = (1 + hM)\|E_k\| + hp\sqrt{\epsilon} + \frac{h^2}{2} N.$$

Let

$$A := hp\sqrt{\epsilon} + \frac{h^2}{2} N.$$

递归变为

$$\|E_{k+1}\| \le (1+hM)\|E_k\| + A.$$

利用初始误差 $E_0 = 0$展开该递归:

$$\|E_k\| \le A\sum_{i=0}^{k-1}(1+hM)^i = A\frac{(1+hM)^k-1}{(1+hM)-1} = \frac{(1+hM)^k-1}{hM}A$$
$$= \frac{(1+hM)^k-1}{2M}\left(hN + 2p\sqrt{\epsilon}\right).$$

对于 $hM < 1$，众所周知

$$(1+hM)^k \le e^{hMk} = e^{Mt_k}.$$

因此,

$$\|E_k\| \le \frac{e^{Mt_k}-1}{2M}\left(hN + 2p\sqrt{\epsilon}\right).$$

由于 $t_k \le 1$，在整个积分区间上,

$$\|E_k\| \le \frac{e^{M}-1}{2M}\left(hN + 2p\sqrt{\epsilon}\right).$$

□

A.2 定理4.2的证明

根据引理4.1，我们知道，完整生成后（即在 $t_k = 1$ 处），总误差有如下上界:

$$\|E\| \;\le\; \frac{e^{M}-1}{2M}\left(hN + 2p\sqrt{\epsilon}\right).$$

该上界可分解为两部分:

$$\underbrace{\frac{e^{M}-1}{2M}\,hN}_{\text{discretization error}} \quad + \quad \underbrace{\frac{e^{M}-1}{M}\,p\sqrt{\epsilon}}_{\text{speculative deviation}}.$$

第一项对应完整生成过程固有的离散化误差，而第二项量化了推测生成所引入的额外偏差。为确保推测偏差不超过容许容差 q_d，只需满足

$$\frac{e^{M}-1}{M}\,p\sqrt{\epsilon} \;\le\; q_d.$$

重新整理后，可得充分条件

$$\epsilon \;\le\; \left(\frac{q_d}{2A}\right)^2, \qquad A = \frac{e^{M}-1}{M}.$$

因此，推测生成引入的偏差仍在规定容差 q_d 以内，证明完毕。

□

A.3 多轮迭代编辑

图 7 评估了多步编辑的鲁棒性：依次应用一系列编辑操作，然后将其撤销，以重建原始图像。为了量化重建保真度，我们报告了恢复后的图像与真实输入之间的学习感知图像块相似度、结构相似性和峰值信噪比。

这一设置的动机在于，基于扩散的编辑模型通常会在每次编辑过程中向图像注入少量噪声。虽然单次编辑产生的这种噪声通常难以察觉，但在连续执行多次编辑时，噪声可能会显著累积。在这种情况下，

朴素的加速策略——例如均匀减少步数——往往会放大累积噪声，从而降低重建质量。

这一现象如图 11 所示。尽管完整的 50 步过程会引入一些噪声，但仍保留了原始输入的大部分结构和语义一致性。相比之下，直接缩减至 25 步会产生明显不同的输出，随着编辑次数增加，差异也会不断累积。

我们的方法避免了这一缺陷。通过有选择地仅消除真正冗余的步骤，该方法大幅缩短了推理时间，同时保持了与完整模型相当的重建质量。这表明我们的方法不会在连续编辑过程中累积误差，因此是更可靠、更稳定的实用编辑工作流解决方案。

A.4 消融实验

图 8 展示了多轮编辑数据集上的加速比与重建质量之间的权衡关系。曲线表明，增大 ϵ 的值可以提升计算加速比，但代价是感知质量下降，衡量指标为峰值信噪比（PSNR）。图中不同颜色表示不同的 ϵ 值，从而分离出该参数对观察到的权衡关系所产生的影响。这种表示方式有助于系统地选择超参数：对于给定的目标加速比，可以确定步长和 ϵ，使二者共同实现效率与保真度之间的最佳平衡。例如，如图 8 所示，采用 50 个采样步和 $\epsilon = 0.07$即可实现最佳的 $3\times$加速比，同时保持质量并大幅降低推理成本。因此，该图为在效率约束下，如何在推测解码中有原则地选择超参数提供了实证依据。

A.5 FlowCast 的自适应性

在图10中，我们重点展示了 FlowCast 的自适应特性。当输入提示对应相对简单的生成任务时，该方法会接受更大比例的草稿提案，从而显著提升加速比。相较之下，对于更复杂的提示，接受的草稿更少，因此加速比有所降低，但保真度更高——这反映出在具有挑战性的情况下需要进行更精细的调整。在本研究考察的所有任务中，包括图像生成、图像编辑和视频生成，都能持续观察到这一趋势。这种行为表明，FlowCast 会根据输入的复杂度自动调整效率与质量之间的权衡，使其成为适用于多种生成场景的灵活且广泛适用的框架。

A.6 基线方法比较

直接缩减。 一种简单的替代方案是直接减少步数，以截断扩散过程。然而，这种静态缩减假设可以重复使用上一步的更新来近似被移除步骤中的速度。这种过度简化会丢弃有价值的中间信息，导致质量明显下降。相比之下，FlowCast 不会盲目丢弃步骤，而是利用模型输出自适应地识别哪些步骤确实冗余，从而在不牺牲质量的情况下加快推理。

TeaCache。 TeaCache 通过对时间步调制的噪声输入拟合多项式，来确定缓存哪些步数，从而加速采样。虽然方法简单，但存在两个主要缺点：1）拟合多项式需要针对特定模型和任务进行校准。2）它并非真正自适应——一旦确定目标加速比，TeaCache 就会反复采用相同的缓存模式（例如，为实现 $2\times$ 加速比而交替缓存步数），而不考虑提示复杂度。在我们的实验中（50 个视频和 100 次多样化的图像生成），我们始终观察到这种重复的缓存行为，表明它难以应对提示特定难度。

FlowCast 的优势。与静态方法不同，FlowCast真正具备自适应能力。它会在推理过程中实时学习，根据生成任务的复杂程度调整加速比。对于较简单的提示，它会大幅减少模型调用以最大限度提高效率；对于较难的提示，则会保留更多步数以保障保真度。这种动态调整使 FlowCast 相较于直接缩减和 TeaCache，始终能够实现更优的效率—质量权衡。

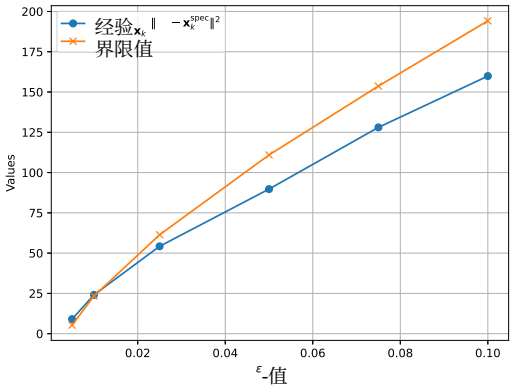


图 6：FlowCast 与真实界值的误差对比。

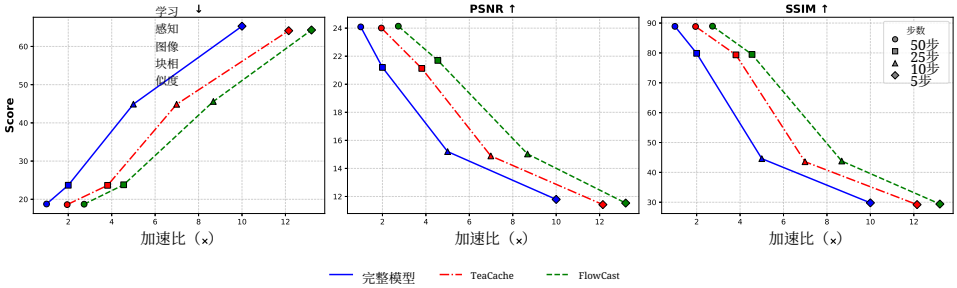


图 7：BAGEL 模型在多次迭代编辑任务上的结果，对比原始图像和最终图像，并比较学习感知图像块相似度、峰值信噪比和结构相似性指标。

A.7 界的紧致性

在图 6 中，我们将 4.1 节推导出的理论误差界与经验观测误差进行了比较。当 ϵ的值非常小时，理论误差界与经验误差非常接近，表明该误差界在低容差区间内是紧的。随着 ϵ增大，我们开始观察到误差界与经验值之间的差距略有扩大。这种现象是意料之中的：更大的ϵ 允许算法采取越来越激进的跳步，而这会带来更高的误差累积风险。

换言之，当接受阈值较大时，该界通过高估潜在误差来“求稳”，从而确保即使在最坏情况偏差下，保证仍然成立。因此，在较高的 ϵ 处略显宽松，正反映了该界在更易出错条件下的保护性。

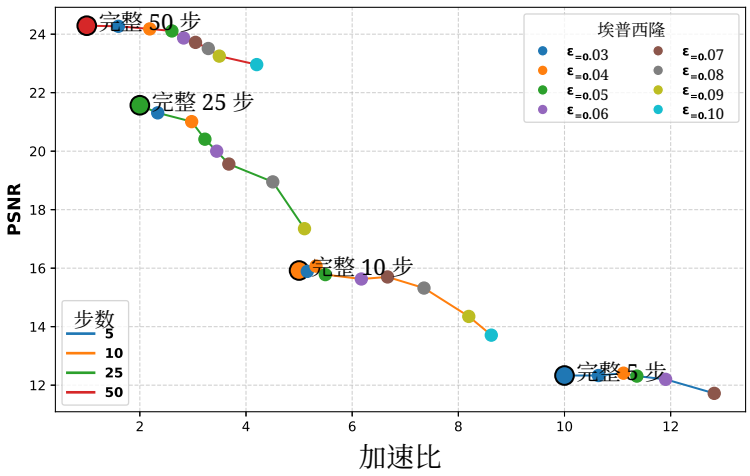


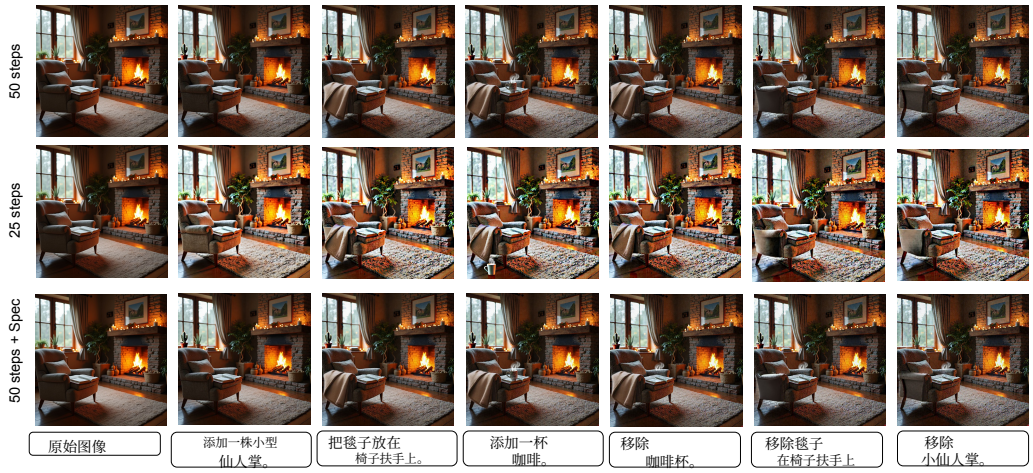
图 8：BAGEL 模型在多轮编辑数据集上的加速比与质量曲线。



图9：推测步数缩减确保时间连贯性和质量，优于直接步数缩减。



图10：编辑和生成任务的示例，展示了我们的方法的自适应性：复杂生成任务的加速比较低，反之亦然。



Fig图11：一个多轮编辑任务示例，其中推测式生成的性能 - tion 优于步数减少后的常规生成。



图12：使用提示词进行多次生成的示例。

方法	SO	TO	CO	CL	CA	PO	总体↑	CLIPIQA↑	速度 ↑
完整模型									
BAGEL-50	0.99	0.90	0.81	0.85	0.59	0.54	**0.78**	**0.84**	1.0×
BAGEL-25	0.99	0.90	0.78	0.84	0.62	0.51	0.77	0.82	2.0×
BAGEL-10	0.99	0.88	0.68	0.84	0.56	0.48	0.73	0.71	5.0×
BAGEL-5	0.93	0.62	0.55	0.72	0.25	0.36	0.57	0.40	10×
静态基线									
InstaFlow	0.85	0.19	0.20	0.65	0.04	0.02	0.33	0.70	50×
PerFlow	0.99	0.79	0.43	0.85	0.25	0.15	0.58	0.79	5.0×
TeaCache	0.99	0.89	0.78	0.83	0.58	0.52	0.76	0.80	1.8×
推测解码（本文）									
Spec-50	0.99	0.90	0.81	0.85	0.61	0.54	**0.78**	0.83	2.5×
Spec-25	0.99	0.91	0.77	0.84	0.62	0.51	0.77	0.82	4.1×
Spec-10	0.99	0.89	0.67	0.82	0.55	0.48	0.73	0.70	7.8×
Spec-5	0.93	0.61	0.55	0.73	0.25	0.36	0.57	0.38	13.0×

表 2：**BAGEL 模型图像生成结果**。我们报告 GenEval 分数（SO：单个物体，TO：两个物体，CO：计数，CL：颜色，CA：类别，PO：位置）、总分、CLIPIQA 和加速比（速度）。推测式解码在保持质量接近完整模型的同时实现了显著加速，优于静态基线。

方法	SO	TO	CO	CL	CA	PO	总体↑	CLIPIQA ↑	加速比 ↑
完整模型									
FLUX-50	0.97	0.79	0.73	0.78	0.44	0.20	**0.65**	**0.83**	1.0×
FLUX-25	0.97	0.78	0.71	0.75	0.43	0.18	0.64	0.80	2.0×
FLUX-10	0.97	0.65	0.58	0.67	0.41	0.15	0.57	0.59	5.0×
FLUX-5	0.88	0.43	0.47	0.54	0.23	0.08	0.44	0.43	10×
TeaCache	0.97	0.78	0.70	0.75	0.43	0.20	0.64	0.81	1.9×
推测解码（我们的方法）									
Spec-50	0.97	0.78	0.72	0.77	0.45	0.20	**0.65**	**0.83**	2.4×
Spec-25	0.97	0.78	0.70	0.76	0.42	0.18	0.64	0.80	4.2×
Spec-10	0.96	0.66	0.58	0.67	0.40	0.15	0.57	0.60	7.6×
Spec-5	0.87	0.42	0.46	0.53	0.24	0.08	0.43	0.47	12.8×

表 3：**FLUX 图像生成模型上的结果**。我们报告 GenEval 分数（SO：单个物体，TO：两个物体，CO：计数，CL：颜色，CA：类别，PO：位置）、总分、CLIPIQA 和加速比（速度）。推测解码在保持性能接近完整模型的同时实现了显著加速，并超越了静态基线。

方法	SO	TO	CO	CL	CA	PO	总体↑	CLIPIQA↑	速度 ↑
完整模型									
PerFlow-10	0.99	0.79	0.37	0.88	0.21	0.15	**0.57**	**0.86**	1.0×
PerFlow-6	0.99	0.78	0.36	0.86	0.20	0.15	0.55	0.83	2.0×
PerFlow-3	0.98	0.75	0.29	0.85	0.18	0.12	0.52	0.78	3.3×
推测解码（我们的方法）									
Spec-10	1.00	0.78	0.37	0.88	0.22	0.15	**0.57**	0.85	1.9×
Spec-6	0.99	0.77	0.37	0.87	0.21	0.15	0.55	0.82	2.7×
Spec-3	0.98	0.74	0.28	0.85	0.18	0.11	0.52	0.77	3.5×

表 4：**PerFlow 模型在图像生成上的结果**。 我们报告 GenEval 分数（SO：单个物体，TO：两个物体，CO：计数，CL：颜色，CA：类别，PO：位置）、总分、CLIPIQA 和加速比（速度）。推测解码在保持质量接近完整模型的同时，实现了显著加速。

方法	G_SC ↑	G_PQ ↑	G_O ↑	速度 ↑
完整模型				
Step-Edit-30	7.54	7.43	**6.94**	1.0×
Step-Edit-15	**7.55**	7.31	6.89	2.0×
Step-Edit-10	7.43	7.11	6.54	3.0×
+ TeaCache				
TeaCache-30	7.51	7.41	6.91	1.8×
TeaCache-15	7.50	7.25	6.81	2.6×
TeaCache-10	7.39	7.06	6.44	3.4×
+ FlowCast（我们的方法）				
FlowCast-30	7.53	**7.49**	**6.95**	2.3×
FlowCast-15	7.53	7.32	6.88	3.0×
FlowCast-10	7.42	7.09	6.50	3.6×
+ TeaCache + FlowCast				
TeaCache+FlowCast-30	7.52	7.41	6.90	2.4×
TeaCache+FlowCast-15	7.48	7.28	6.85	3.0×
TeaCache+FlowCast-10	7.35	6.98	6.35	3.7×

表5：**加速策略的定量比较**。我们报告语义一致性（G SC）、感知质量（G PQ）、总分（G O）和加速比（速度）。FlowCast持续加速推理，同时保持质量，并与TeaCache保持互补。

方法	延迟（秒）	内存（GB）	加速比
完整 50	33.8	48	1.00×
推测式 50	15.5	79	2.40×
TeaCache	18.3	48	1.84×
完整 25	17.5	48	2.00×
推测式 25	10.2	71	4.20×
完整 10	7.1	48	5.00×
推测式 10	5.38	58	7.60×
完整 5	3.9	48	10.00×
推测式 5	3.1	52	12.80×

表6：FLUX 模型的内存分析。

方法	延迟（秒）	内存（GB）	加速比
完整 50	36.8	29	1×
推测式 50	16.1	65	2.5×
TeaCache	18.3	29	1.8×
完整 25	19.5	29	2×
推测式 25	10.5	58	4.1×
完整 10	7.4	29	5×
推测式 10	5.59	46	7.8×
完整 5	3.8	29	10×
推测式 5	3.2	36	13.0×

表7：BAGEL 模型的内存分析。