# FLOWCAST: TRAJECTORY FORECASTING FOR SCALABLE ZERO-COST SPECULATIVE FLOW MATCHING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Flow Matching (FM) has recently emerged as a powerful approach for high-quality visual generation. However, their prohibitively slow inference due to a large number of denoising steps limits their potential use in real-time or interactive applications. Existing acceleration methods, like distillation, truncation, or consistency training, either degrade quality, incur costly retraining, or lack generalization. We propose FlowCast, a training-free speculative generation framework that accelerates inference by exploiting the fact that FM models are trained to preserve constant velocity. FlowCast speculates future velocity by extrapolating current velocity without incurring additional time cost, and accepts it if it is within a mean-squared error threshold. This constant-velocity forecasting allows redundant steps in stable regions to be aggressively skipped while retaining precision in complex ones. FlowCast is a plug-and-play framework that integrates seamlessly with any FM model and requires no auxiliary networks. We also present a theoretical analysis and bound the worst-case deviation between speculative and full FM trajectories. Empirical evaluations demonstrate that FlowCast achieves $> 2.5\times$ speedup in image generation, video generation, and editing tasks, outperforming existing baselines with no quality loss as compared to standard full generation.

## 1 INTRODUCTION

**Flow Matching (FM)** Lipman et al. (2022) has recently emerged as a powerful and principled approach for generative modeling Bond-Taylor et al. (2021). By learning a time-dependent *velocity field* that continuously transforms samples from a base distribution to a target distribution, FM models generate data by integrating an **Ordinary Differential Equation (ODE)**. This framework offers advantages over diffusion-based methods Croitoru et al. (2023), including faster convergence, stable training, and direct control over sample trajectories.

However, despite these strengths, **inference in FM models is costly and is a bottleneck** Kornilov et al. (2024). ODE integration is inherently sequential—each step depends on the output of the previous one—making the process slow and difficult to parallelize. For high-fidelity generation, a large number of fine-grained steps are needed, pushing the latency cost higher, which makes it difficult for real-time or interactive use. As data dimensionality and model complexity increase, so does the inference burden. This issue is even more pronounced in video generation Kong et al. (2024), where models operate on high-dimensional spatio-temporal data. The sequential ODE integration compounds across frames, making efficient inference essential for long videos Liu et al. (2025a).

To reduce inference latency, static reduction strategies such as *distillation* Luhman & Luhman (2021); Yan et al. (2024); Kornilov et al. (2024), *trajectory truncation* Dhariwal & Nichol (2021); Lu et al. (2022); Liu et al. (2025a), and *consistency training* Yang et al. (2024); Zhang & Zhou (2025); Dao et al. (2025) have been proposed. However, these approaches often degrade output quality—resulting in blurry textures, semantic drift, or broken spatial coherence. In the video domain, such approaches often cause temporal flickering and motion inconsistency, which severely degrade realism despite visually plausible individual frames. Furthermore, many of these methods require retraining the model with auxiliary objectives or handcrafted losses, which not only increases the engineering burden but also restricts transferability across model families. As a result, these approaches are less modular, harder to scale, and often infeasible to deploy in time-sensitive or interactive pipelines.

A promising line of work to accelerate inference in language models is **Speculative Decoding (SD)** Xia et al. (2024), where multiple drafts are proposed in parallel and selectively verified to reduce latency. However, this idea does not directly extend image and video generative models such as Flow Matching (FM). Some of the unique challenges are: (1) Unlike in language models, no readily available draft model exists. Then, how to design such a draft model whose trajectories closely couple with the full model? (2) Performance of FM is highly sensitive to velocity output. Even a small perturbation in velocity can accumulate along the integration path, leading to severe generative errors. Then how to evaluate the draft model velocity and control the trajectory drifts? We address these challenges using properties of FM trajectories.

**Our Key Insight.** We observe that FM trajectories are typically *locally smooth and slowly varying* Lipman et al. (2022), especially in well-trained models. Leveraging this property, we propose a speculative framework FlowCast where the model's own past velocity predictions act as a *zero-cost draft* for future steps. These drafts are selectively validated using a mean-squared error check: if accurate within the limits of a threshold, we skip recomputation entirely; if not, we fall back to the backbone. This mechanism introduces negligible overhead, requires no retraining or auxiliary models, and remains fully compatible with existing FM pipelines.

**Why it matters.** Unlike previous acceleration techniques that require retraining or trade off generation fidelity, our method is *plug-and-play*, adapts to the complexity of the trajectories, and leverages the inherent smoothness of the FM. In regions with stable dynamics, it skips steps aggressively to accelerate inference, while in complex regions it falls back to fine-grained integration for accuracy. This design enables significant speedups **without compromising output quality**. A key advantage is that the approach is entirely training-free and readily applicable to existing methods, allowing even current acceleration strategies to be further enhanced.

FlowCast is task-agnostic: it applies seamlessly to image generation, image editing, and video generation, where it ensures faithful, high-quality edits. This broad applicability highlights speculative generation as a unified framework for accelerating generative models. We compare the trajectories of the FlowCast and those obtained from the full ODE to develop a theoretical bound on their deviations. The resulting error bound guarantees the stability and fidelity of our method, ensuring that speculative generation remains close to the original FM solution while enabling faster inference.

In summary, our major contributions are:

- We introduce a speculative generation framework viz. FlowCast for flow-matching-based generative models that enables adaptive, partially parallel inference.
- FlowCast uses zero-cost drafts from the model's own velocity predictions, avoiding retraining and preserving trajectory fidelity.
- We derive a theoretical error bound that formally characterizes how much the speculative trajectory can deviate from full ODE integration.
- Our method achieves substantial inference acceleration ($> 2.5\times$) while maintaining sample quality, outperforming existing baselines—especially in multi-turn editing tasks.

## 2 Related works

Flow Matching Lipman et al. (2022); Dao et al. (2023); Labs et al. (2025); Deng et al. (2025) has recently emerged as a strong alternative to diffusion models Croitoru et al. (2023); Xing et al. (2024); Yang et al. (2023); Zhu et al. (2023), providing a deterministic mapping between noise and data. This determinism makes flow-based models appealing for applications such as image inversion Deng et al. (2024), editing Wang et al. (2024), and video generation Kong et al. (2024); Wan et al. (2025), as it reduces randomness and enables faster sampling with fewer neural function evaluations.

Multimodal extensions like FlowTok He et al. (2025) compress text and images into shared token spaces for faster inference, and FLUX.1 Labs et al. (2025) shows large-scale flows can rival diffusion models at lower cost. For video, Pyramidal Flow Matching Jin et al. (2024) reduces compute via hierarchical generation.

However, iterative sampling remains a major bottleneck in these models, restricting real-time and interactive deployment. To accelerate inference, prior work has focused on retraining-based solutions Lee et al. (2023); Bartosh et al. (2024). Distillation methods Song et al. (2023); Luhman & Luhman

(2021); Liu et al. (2022); Kornilov et al. (2024); Salimans & Ho (2022) like InstaFlow Liu et al. (2023), LeDiFlow Zwick et al. (2025) and Diff2Flow Schusterbauer et al. (2025) transfer knowledge from diffusion priors for one- or few-step sampling, while PeRFlow Yan et al. (2024) straightens trajectories with piecewise rectified flows. Methods used for sampling modification Dhariwal & Nichol (2021); Lu et al. (2022); Shaul et al. (2023) such as TeaCache Liu et al. (2025a) skip redundant steps by getting informed by the timestep-embedding, and consistency-based approaches Yang et al. (2024); Zhang & Zhou (2025); Haber et al. (2025); Dao et al. (2025) combine adversarial and consistency losses for high-quality few-step generation.

Our work accelerate the inference process using the speculative decoding framework making the FM more suitable for real-time and interactive applications. In contrast to the existing methods, our method do not require any retraining, distillation, trajectory straightening, or step skipping. Our method offers a fully plug-and-play solution for the existing flow models. It operates dynamically at inference time by speculating future steps and verifying them on the fly. This design delivers substantial speedup, while preserving the model's original generative quality.

# 3 SETUP

In this section, we provide the foundational formulation of Flow Matching (FM) and Speculative Decoding (SD) frameworks.

## 3.1 SPECULATIVE DECODING FRAMEWORK

SD is a parallelizable inference strategy designed to accelerate autoregressive generation, particularly in large language models (LLMs), without retraining or sacrificing output quality. The framework leverages the observation that evaluating a sequence of tokens in parallel is only marginally slower than evaluating a single token, enabling partial parallelism during decoding.

Let $\mathcal{M}_p$ denote the target (slow but accurate) model, and $\mathcal{M}_q$ denote a draft model (faster but less accurate). In SD, the objective is to accelerate token generations from $\mathcal{M}_p$ by verifying the speculative drafts generated by $\mathcal{M}_q$. The verification can be done at a higher speed than the generation, thus achieving speedup. Given an input with $i$ tokens $x_{1:i} = (x_1, x_2, \ldots, x_i)$, SD has two phases:

**1) Drafting Phase:** The draft model $\mathcal{M}_q$ generates $K$ (speculative) tokens autoregressively: $(x_{i+1}^{\text{draft}}, x_{i+2}^{\text{draft}}, \ldots, x_{i+K}^{\text{draft}}) \sim \mathcal{M}_q(\cdot \mid x_{1:i})$. This phase is fast due to the small size and shallow architecture of $\mathcal{M}_q$.

**2) Verification Phase:** The full model $\mathcal{M}_p$ evaluates the entire speculative sequence in a single forward pass, computing: $\mathcal{M}_p(x_{i+k} \mid x_{1:i+k-1}^{\text{draft}})$, for $k = 1, \ldots, K$. These predicted distributions are compared against the draft tokens to assess validity (e.g., via cross-entropy or top-k agreement). Let $j$ be the first index where $\mathcal{M}_p$'s prediction deviates significantly from the draft. Then the prefix $x_{i+1}^{\text{draft}}, \ldots, x_{i+j-1}^{\text{draft}}$ is accepted. Accepted tokens are then passed again to the draft model to generate new drafts based on updated context.

This speculative verification cycle continues iteratively until a full sequence is produced. The method improves throughput without modifying $\mathcal{M}_p$, and is particularly effective when the draft model's outputs closely align with the target model.

Under the assumption that $\mathcal{M}_q$ is sufficiently aligned with $\mathcal{M}_p$, the speculative decoding process preserves output quality while accelerating generation. Moreover, this approach avoids retraining and can be applied to any pair $(\mathcal{M}_q, \mathcal{M}_p)$ where $\mathcal{M}_q$ is reasonably well-calibrated.

Adapting speculative decoding to image generation is nontrivial, as drafting valid future states in continuous, high-dimensional latent spaces is far more complex than next-token prediction in language models. Drafts must maintain spatial and temporal consistency in predicted noise or velocity, which is difficult without a trained surrogate. Moreover, small trajectory errors can accumulate under ODE/SDE integration, causing semantic drift, structural distortions, or misalignment with conditioning. These compounding effects make speculative decoding fragile in visual domains, necessitating drafting and verification strategies that explicitly respect the geometry and stability of the underlying dynamics.

### 3.2 Flow Matching Overview

Let $\pi_0$ and $\pi_1$ be two smooth probability densities over $\mathbb{R}^d$, representing the source and target distributions, respectively. The goal of **Flow Matching (FM)** is to construct a deterministic and continuous mapping that transports samples from $\pi_0$ to $\pi_1$ through a time-indexed trajectory governed by an ordinary differential equation (ODE). Specifically, FM defines the transformation using a time-dependent velocity field $v : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$, yielding the following initial value problem:

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

Here, $x_t \in \mathbb{R}^d$ denotes the state of the sample at time $t$, and $v(x_t, t)$ is typically parameterized by a neural network trained to align the marginal density at $t = 1$, denoted $\pi_1$, with the target distribution. The ODE defines a flow map $\Phi_t(x_0)$ that moves points in $\mathbb{R}^d$ along continuous trajectories, with $\Phi_1(x_0) \sim \pi_1$. Learning a valid $v(x_t, t)$ that realizes this transport is the central objective in FM.

**Numerical Integration:** In practice, the solution $x_t$ of the ODE must be computed numerically, as closed-form solutions are generally unavailable for learned velocity fields. The most widely used method for solving the ODE in flow matching models is the **forward Euler method**, due to its simplicity and computational efficiency. The time interval $[0, 1]$ is discretized into a sequence of $K$ points $\{t_0 = 0, t_1, \ldots, t_K = 1\}$, which may be non-uniformly spaced. Starting from $x_0 \sim \pi_0$, the trajectory is updated iteratively as

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

This discretization approximates the continuous flow by a finite sequence of steps, with each update advancing the state in the direction prescribed by the velocity field. The Euler method is favoured in most implementations for its low computational overhead Deng et al. (2025); Labs et al. (2025).

**Trajectory Smoothness:** The validity and accuracy of Euler-based discretization depend on the regularity of the velocity field $v(x_t, t)$. In particular, when the ODE admits a unique solution, the discretization errors can be bounded. More importantly, empirical observations in trained FM models reveal that the velocity field tends to vary gradually across time steps and sample locations Liu et al. (2025a), particularly when trained using interpolants or synthetic velocity targets, indicating that the velocity does not fluctuate sharply between adjacent steps.

## 4 Speculative Decoding for Flow models

In this section, we present our core contributions: establishing a connection between speculative decoding and flow matching models, introducing our speculative generation framework, and providing theoretical error bounds that formally characterize its fidelity.

**Autoregressive nature of FM:** SD works well for the autoregressive models. FM process is inherently autoregressive as each denoising step depends entirely on the output of the previous step, creating a strict sequential dependency chain—much like generating tokens one by one in a language model. Though, SD is used in language tasks where the autoregression is in the discrete space (e.g., tokens from a discrete set), they can also be leveraged in FMs where autoregression is in the continuous space (e.g., velocity in a bounded interval).

**Zero-Cost Draft Construction:** To enable speculative decoding in FM models without relying on a separate draft network, we propose a lightweight and self-contained strategy for draft generation by directly reusing the velocity predictions from the target model itself.

Recall from Section 3.2 that FM models learn a time-dependent velocity field $v(x_t, t)$ that transports samples from the source distribution $\pi_0$ toward the target distribution $\pi_1$ via integration of the ODE:

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

During training, the model $v$ is typically supervised to match a reference velocity (often constant over time) along interpolated paths. This encourages the model $v$ to vary smoothly in both space and time. Leveraging this smoothness, we propose FlowCast, where the velocity is kept constant to extrapolate not just the immediate next state, but the entire remaining trajectory, resulting in zero-cost draft trajectories. Each step of the trajectories is verified by the full model in parallel, and

either accept them or apply corrections. FlowCast has three components: Drafting, Verification, and Correction. Its pseudo-code is given in Algorithm 1.

**Algorithm 1** FlowCast

**Require:** Initial state $x_{t_0}$, timesteps $\{t_k\}_{k=0}^{K}$, threshold $\epsilon$, model $v$

1: Compute initial velocity: $v_{t_0} \leftarrow v(x_{t_0}, t_0)$
2: Predict next state: $x_{t_1} \leftarrow x_{t_0} + (t_1 - t_0)v_{t_0}$
3: Set current index $m \leftarrow 0$
4: **repeat**
5: **(Drafting)**: Generate speculative states:

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

6: **(Parallel Verification)**: For each $k = m+1, \ldots, K$, compute:

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

7: Find first index $j > m$ such that $e_j > \epsilon$
8: **if** no rejection (i.e., $e_k \leq \epsilon$ for all $k$) **then**
9: Accept full draft $\{\tilde{x}_{t_k}\}_{k=m+1}^{K}$, set $m \leftarrow K$
10: **else**
11: **(Correction)**: Accept draft $\{\tilde{x}_{t_k}\}_{k=m+1}^{j-1}$
12: $x_{t_j} \leftarrow \tilde{x}_{t_{j-1}} + v_{t_{j-1}}(t_j - t_{j-1}); m \leftarrow j-1$
13: Recompute velocity: $v_{t_m} \leftarrow v(x_{t_m}, t_m)$
14: **end if**
15: **until** $m = K$

**Drafting:** Given the current state $x_{t_i}$ at step $t_i$, define the speculative trajectory as:

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

for $k = 1, \ldots, n$ and $k > i$. Here $n$ denotes the number of remaining time steps to be traversed to complete the trajectory, and $\Delta t_k = t_k - t_i$. This effectively reuses the same velocity vector to linearly extrapolate the remaining path.

This method introduces no auxiliary models or additional computational cost during drafting. While simple, it leverages the inductive bias of FM models to generate coherent trajectories, particularly in domains where velocity evolution is smooth.

**Verification:** We employ a lightweight yet principled mechanism to verify the validity of each drafted state. For every extrapolated point $\tilde{x}_{t+\Delta t}$, we compare the reused velocity $v(x_t, t)$ with the model's own velocity prediction at the extrapolated point and step, i.e., $v(\tilde{x}_{t+\Delta t}, t + \Delta t)$. If the mean squared error (MSE) between the two remains below a pre-defined threshold $\epsilon$, the draft is accepted, i.e.,

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

**Correction:** If the criterion is violated at a given step, the draft is rejected, and all subsequent drafts are discarded. A new trajectory is then re-initiated starting from the rejection time step, using fresh velocity predictions from the model as per Equation 4.

This verification strategy is efficient and directly aligned with the FM formulation, where the velocity field encodes the generative dynamics. Unlike traditional methods that assess validity in data or latent space, our velocity-based criterion operates in function space, making it more sensitive to inconsistencies in local dynamics while remaining inexpensive to compute. This ensures the integration path remains faithful to the learned flow without unnecessary recomputation.

The working procedure of FlowCast is illustrated in Figure 1. The left part shows the normal generation process, where the FM model is invoked sequentially in each step of the trajectory. The right side shows the FlowCast process for a trajectory generated from the drafting step. All the speculatively generated drafts in the trajectory are processed in parallel through the FM model. In the regular FM generation, the output of the model in each step depends only on the output of the previous step. We use this fact to verify the correctness of the drafts. In particular, if the output of the FM model at step $k$ matches that of the speculated draft at step $k+1$ within the MSE threshold, the speculated draft at $k+1$ is accepted, else rejected. This verification is performed efficiently in a single forward pass. In Figure 1, drafts are accepted up to step 3, but the draft at step 4 is rejected. Once a draft is rejected at a step, all later drafts are discarded, and the trajectory is computed from the last accepted step.

### 4.1 Analysis

In this subsection, we bound the deviations in the draft trajectories and that of the full model. We then provide a bound on the MSE threshold to guarantee the worse case deviations.

**Lemma 4.1.** *Let $x(t)$ be the solution of $\frac{dx}{dt} = v(x,t), x(0) = x_0$, where $v$ is Lipschitz in $x$ with constant $M$. Assume: 1) $\|x''(t)\| \leq N$ for all $t \in [0,1]$, 2) $\|\frac{\partial v}{\partial x}(x,t)\| \leq M$ for all $(x,t)$, 3) a*

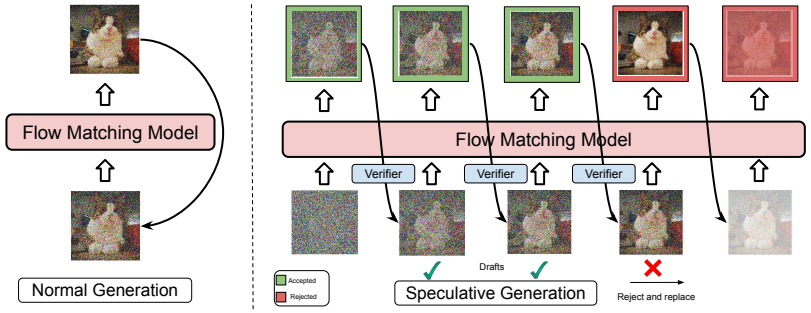


Figure 1: Comparison of normal vs. speculative image generation: The left side is the normal iterative process of generation in FM model. The right side is our approach which introduces intermediate speculative drafts that are rapidly proposed and verified in parallel by the backbone, enabling faster sampling while preserving quality.

*forward Euler scheme with step size $h$ is run for $k$ steps, where speculative velocities $\tilde{v}$ are accepted in a fraction $p \in [0,1]$ of the steps and satisfy $\|\tilde{v}(x_k,t_k) - v(x_k,t_k)\| \le \sqrt{\epsilon}$. Then the global discretization error $E_k = \|x(t_k) - x_k\|$ can be bounded as*

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

Here, the first term $\frac{e^{Mt_k}-1}{2M}hN$ represents the standard Euler discretization error, while the second term $\frac{e^{Mt_k}-1}{M}p\sqrt{\epsilon}$ quantifies the additional deviation introduced by speculative updates. The proof of this lemma is given in Appendix A.1. This lemma leads us to the theorem stated below.

**Theorem 4.2.** *Let $q_d > 0$ denote a user-specified tolerance on the maximum admissible deviation from full (non-speculative) generation trajectory. Then, for the speculative error $\|E_k^s\|$ (i.e., the deviation introduced by speculative generation) to remain bounded by $q_d$, it is sufficient to choose the speculative threshold $\epsilon$ such that*

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

Using this result, we can formally control the worst-case deviation between speculative and full trajectories by setting an appropriate $\epsilon$. Theorem 4.2 serves as a principled mechanism to tune $\epsilon$, ensuring that trajectory deviation through speculative steps does not exceed a quality budget. The proof of the Theorem 4.2 can be found in Appendix A.2.

## 5 Experiments

Below we provide details of our experimental setup and results. We begin with dataset descriptions.

- **Text-to-image generation**: We use **GenEval** dataset Ghosh et al. (2023), consisting of 553 prompts targeting compositional reasoning, with categories evaluating object co-occurrence, spatial positioning, color binding, and object count.
- **Image editing**: We adopt the **GEdit** benchmark Liu et al. (2025b), which provides 606 real-world editing instructions spanning object manipulation, color modification, etc.
- **Multi-turn editing**: We further construct an auxiliary dataset based on EditBench, where GPT-4.1 Achiam et al. (2023) generates three incremental edits followed by three reverse edits restoring the original image. Reconstruction fidelity is quantified via PSNR between the final output and the original.
- **Video generation**: We evaluate on the **VBench** dataset Huang et al. (2024), sampling 80 prompts by uniformly drawing 5 from each of the 16 evaluation dimensions, ensuring balanced coverage across motion, persistence, camera control, and scene composition.

| Method | BAGEL | | | FLUX | | |
|---|---|---|---|---|---|---|
| | Overall ↑ | CLIPIQA ↑ | Speedup ↑ | Overall ↑ | CLIPIQA ↑ | Speedup ↑ |
| *Full Model* | | | | | | |
| 50 steps | **0.78** | **0.84** | 1.00× | **0.65** | **0.83** | 1.0× |
| 25 steps | 0.77 | 0.82 | 2.0× | 0.64 | 0.80 | 2.00× |
| 10 steps | 0.73 | 0.75 | 5.0× | 0.57 | 0.59 | 5.00× |
| 5 steps | 0.57 | 0.40 | 10.0× | 0.44 | 0.43 | 10.0× |
| *Static Baselines* | | | | | | |
| 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× |
| *Ours: Speculative Decoding* | | | | | | |
| 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× |

Table 1: **Comparison of BAGEL and FLUX Models on Image Generation.** We report Overall GenEval, CLIPIQA image quality scores, and Speedup relative to the 50-step full generation.

**Baselines:** We compare our approach against the following baselines:

**Full Generation.** This is the standard sampling procedure, where the model executes the entire denoising trajectory without any acceleration. It provides the upper bound in terms of fidelity and serves as the reference point for all accelerated methods.

**TeaCache.** Liu et al. (2025a) It speeds up generation by caching intermediate representations and reusing them across timesteps, guided by timestep embeddings. By eliminating computations, it achieves faster inference, at the cost of fidelity degradation due to approximation in reused states.

**InstaFlow.**Liu et al. (2023) A flow-matching–based sampler trained for extremely fast generation (down to a single step) in Stable-Diffusionv-1.5 model. While efficient, it sacrifices fidelity compared to full sampling. We evaluate InstaFlow on the released Stable-Diffusion-v1.5 weights.

**PeRFlow (Piecewise Rectified Flow).** PeRFlow Yan et al. (2024) straightens the diffusion trajectory via piecewise-linear rectification over segmented timesteps, enabling few-step generation with strong quality–efficiency tradeoffs on Stable-Diffusion-xl model. We evaluate PerFlow on the released Stable-Diffusion-xl checkpoints.

**FlowCast (Ours).** Our approach accelerates inference by allowing a lightweight draft process to propose multiple future steps, which are then verified in parallel by the full model. We denote this as *FlowCast-*$n$, where $n$ represents the number of speculative steps.

Our method is complementary to existing acceleration techniques such as PeRFlow and TeaCache, further reducing redundant steps even in frameworks already optimized for efficiency. The corresponding results are reported in Tables 4 and 5.

For all baselines, we adopt the official hyperparameters provided in their codebases. In Table 1, TeaCache is applied as released, and in Figure 2, we further evaluate it across different timesteps to emphasize its plug-and-play flexibility. For the choice of $\epsilon$ in FlowCast, from Theorem 4.2, we have found that value of $\epsilon \in [0.01, 0.02]$ for image generation (all models), $\epsilon \in [0.07, 0.08]$ for image editing (all models), $\epsilon \in [0.001, 0.002]$ for video generation were giving high speedups with similar performance as the full model. An ablation study over $\epsilon$ across multiple steps can be found in the Appendix 7.

**Models.** To demonstrate the versatility of our approach, we evaluate across multiple state-of-the-art models: for image generation, BAGEL Deng et al. (2025), Flux-Kontext Labs et al. (2025), and PeRFlow Yan et al. (2024); for image editing, BAGEL, Flux-Kontext, and Step-1X-Edit Liu et al. (2025b); and for video generation, HunyuanVideo Kong et al. (2024) are used.

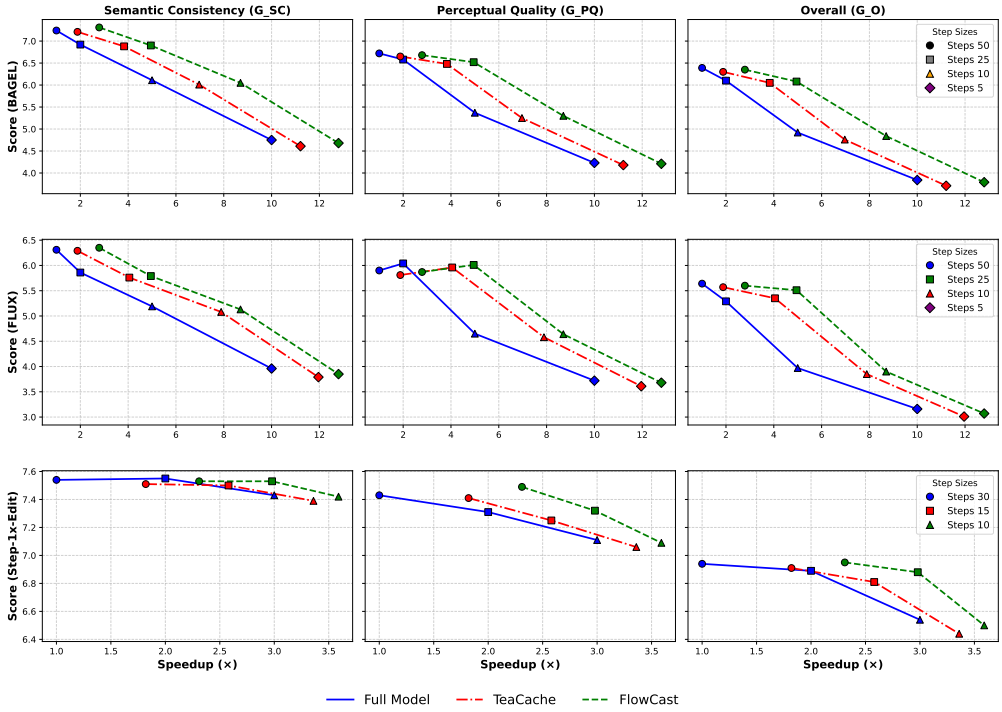


Figure 2: GEdit Scores across three models: BAGEL, FLUX, and Step-Edit. Each subfigure reports semantic consistency (G_SC), perceptual quality (G_PQ), and overall score (G_O) versus speedup.

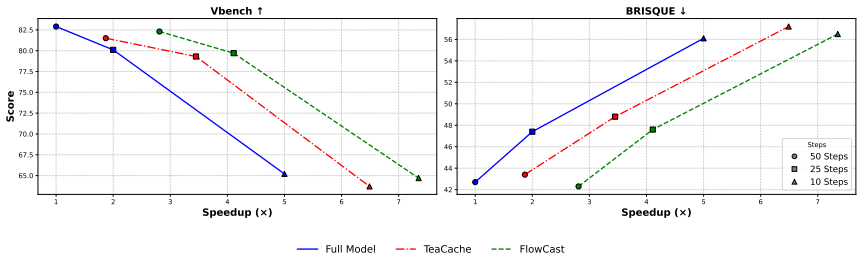


Figure 3: Results on Hunyuan Video model where we report the Vbench score and specifically for quality we use the BRISQUE metrics that assess the quality of individual frames.

### 5.1 Results

**Image Generation:** Table 1 reports results on the GenEval benchmark with BAGEL and FLUX, and Table 4 shows additional evaluation on the optimized PerFlow model. GenEval scores capture semantic dimensions (detailed in Tables 2, 3), while fidelity is assessed via CLIPIQA. Static reduction methods (e.g., fixed step truncation) cause significant quality loss, whereas our adaptive approach achieves the lowest quality drop and preserves semantic alignment.

**Image Editing:** Figure 2 presents GEdit results with BAGEL, FLUX, and Step-1X-Edit, reporting G_SC (semantic consistency), G_PQ (perceptual quality), and G_O (overall editability). Static step reduction degrades both editability and quality, while our adaptive strategy consistently matches full-step generation. Multi-iteration results are shown in Figure 6 and discussed in Appendix A.3.

**Video Generation:** In Figure 3, we extend our analysis to video generation using the Hunyuan-Video model. Performance is reported with the vBench metric, which evaluates temporal coherence, background stability, flicker artifacts, etc. We also report the frame-wise quality score using the BRISQUE metric. Static reduction methods introduce visible temporal inconsistencies and motion artifacts, whereas FlowCast scheme maintains high-quality, temporally stable videos.

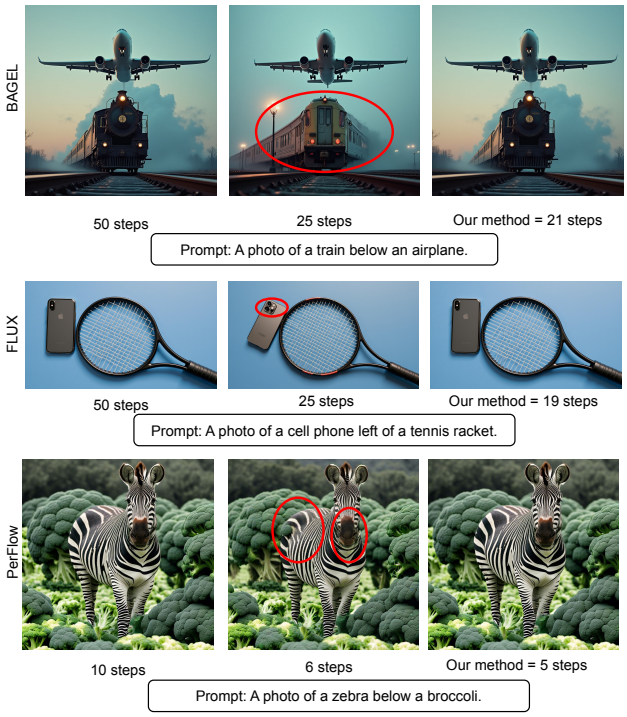


Figure 4: Speculative step reduction maintains image quality across models, whereas direct step reduction leads to noticeable degradation.

**Qualitative analysis:** We qualitatively compare static and adaptive step reduction across tasks. In image generation (Figure 4), static reduction often introduces semantic inconsistencies and artifacts, whereas our adaptive method skips only redundant steps, producing outputs closely aligned with full-step generations. For editing (Figures 8, 11), static reduction yields incomplete or unintended edits and compounds residual noise, while our method preserves semantic accuracy and visual fidelity—even in multi-turn settings. In video generation (Figure 9), static reduction causes temporal ruptures and frame drops, whereas FlowCast maintains smoothness and fidelity.

**Summary:** Across diverse tasks—image generation, editing, multi-iteration editing, and video generation—existing acceleration techniques consistently suffer from loss of fidelity and alignment, highlighting their limited generality. In contrast, FlowCast emerges as a universal adaptive framework for accelerating flow-based models that adapts to the generation complexity (see Appendix A.5). Unlike heuristic truncation, which causes performance drops, FlowCast leverages model-informed verification to skip only redundant updates. This ensures that outputs remain *indistinguishable from full-step generation* while achieving substantial speedups. These results establish FlowCast as the preferred solution for efficient, reliable, and broadly transferable step reduction across modalities.

## 6 Conclusion

We introduced an adaptive inference strategy that accelerates generation by eliminating truly redundant computation. Our approach extends speculative decoding with zero-cost drafts obtained directly from prior velocity predictions, which are then verified in parallel through a lightweight mean-squared error criterion. Beyond its simplicity, we establish theoretical guarantees on the deviation from the original trajectory, ensuring both reliability and rigor. Extensive experiments across diverse models and tasks demonstrate that our method achieves consistent speedups without sacrificing performance, highlighting its potential as a general framework for faster and reliable inference. The main limitation of FlowCast is its dependence on parallel draft evaluations, which require adequate compute; cutting drafts reduces overhead but also reduces speedup.

## 7 Ethics and Reproducibility statement

We have read and adhere to the Code of Ethics as detailed out for this conference. To the best of our knowledge, this is our original work and all related work has been appropriately cited. All authors have contributed towards this work and are responsible for it's content. There is no implicit use of Large Language Models (LLMs), except for the cases where these models are already part of our technical method, and in cases where the use has already been disclosed.

We describe a detailed algorithm to reproduce our work. We use publicly available code, models, and datasets. Additionally, we intend to provide source code using which the results of this work can be reproduced. The results from related works are taken from their official manuscripts and/or official source repositories (as separately specified for each such work).

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Grigory Bartosh, Dmitry P Vetrov, and Christian Andersson Naesseth. Neural flow diffusion models: Learnable forward process for improved diffusion modelling. *Advances in Neural Information Processing Systems*, 37:73952–73985, 2024.

Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7327–7347, 2021.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9): 10850–10869, 2023.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

Quan Dao, Hao Phung, Trung Tuan Dao, Dimitris N Metaxas, and Anh Tran. Self-corrected flow distillation for consistent one-step and few-step image generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 2654–2662, 2025.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Yingying Deng, Xiangyu He, Changwang Mei, Peisong Wang, and Fan Tang. Fireflow: Fast inversion of rectified flow for image semantic editing. *arXiv preprint arXiv:2412.07517*, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

Eldad Haber, Shadab Ahamed, Md Shahriar Rahim Siddiqui, Niloufar Zakariaei, and Moshe Eliasof. Iterative flow matching–path correction and gradual refinement for enhanced generative modeling. *arXiv preprint arXiv:2502.16445*, 2025.

Ju He, Qihang Yu, Qihao Liu, and Liang-Chieh Chen. Flowtok: Flowing seamlessly across text and image tokens. *arXiv preprint arXiv:2503.10772*, 2025.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Nikita Kornilov, Petr Mokrov, Alexander Gasnikov, and Aleksandr Korotin. Optimal flow matching: Learning straight trajectories in just one step. *Advances in Neural Information Processing Systems*, 37:104180–104204, 2024.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Sangyun Lee, Beomsu Kim, and Jong Chul Ye. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pp. 18957–18973. PMLR, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 7353–7363, 2025a.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, et al. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025b.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2023.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Johannes Schusterbauer, Ming Gui, Frank Fundel, and Björn Ommer. Diff2flow: Training flow matching models via diffusion model alignment. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 28347–28357, 2025.

Neta Shaul, Juan Perez, Ricky TQ Chen, Ali Thabet, Albert Pumarola, and Yaron Lipman. Bespoke solvers for generative flow models. *arXiv preprint arXiv:2310.19075*, 2023.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing. *arXiv preprint arXiv:2411.04746*, 2024.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

Zhen Xing, Qijun Feng, Haoran Chen, Qi Dai, Han Hu, Hang Xu, Zuxuan Wu, and Yu-Gang Jiang. A survey on video diffusion models. *ACM Computing Surveys*, 57(2):1–42, 2024.

Hanshu Yan, Xingchao Liu, Jiachun Pan, Jun Hao Liew, Qiang Liu, and Jiashi Feng. Perflow: Piecewise rectified flow as universal plug-and-play accelerator. *Advances in Neural Information Processing Systems*, 37:78630–78652, 2024.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39, 2023.

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

Yuchen Zhang and Jian Zhou. Inverse flow and consistency models. In *Forty-second International Conference on Machine Learning*, 2025.

Yuanzhi Zhu, Zhaohai Li, Tianwei Wang, Mengchao He, and Cong Yao. Conditional text image generation with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14235–14245, 2023.

Pascal Zwick, Nils Friederich, Maximilian Beichter, Lennart Hilbert, Ralf Mikut, and Oliver Bringmann. Lediflow: Learned distribution-guided flow matching to accelerate image generation. *arXiv preprint arXiv:2505.20723*, 2025.

# A Appendix

## A.1 Proof of Lemma 4.1

*Proof.* Define the error at step $k+1$ as

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

Using the Euler update,

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

where $v^{used}$ is either $v_\theta$ or the speculative velocity $\tilde{v}$ when accepted.

By Taylor expansion of $x(t)$ around $t_k$, there exists some $\psi \in (t_k, t_{k+1})$ such that

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

Therefore,

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

By the Lipschitz continuity of $v_\theta$ in $x$,

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

Also, by assumption, at steps where the speculative velocity is accepted,

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

Accounting for the fraction $p$ of such steps,

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

Combining the above,

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

Let

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

The recursion becomes

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

Unrolling this recursion with initial error $E_0 = 0$:

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

For $hM < 1$, it is well-known that

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

Hence,

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

Since $t_k \le 1$, for the entire integration interval,

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

□

### A.2 Proof of Theorem 4.2

From Lemma 4.1, we know that after full generation (i.e., at $t_k = 1$), the total error is bounded by

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

This upper bound can be decomposed into two contributions:

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

The first term corresponds to the inherent discretization error of the full generation process, whereas the second term quantifies the additional deviation induced by speculative generation. To ensure that the speculative deviation does not exceed the admissible tolerance $q_d$, it suffices to require

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

Rearranging, we obtain the sufficient condition

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

Hence, the deviation introduced by speculative generation remains within the prescribed tolerance $q_d$, completing the proof. □

### A.3 MULTI-ITERATION EDITING

Figure 6 evaluates the robustness of multi-step editing, where a sequence of edits is applied and subsequently reverted to reconstruct the original image. To quantify reconstruction fidelity, we report LPIPS, SSIM, and PSNR between the restored and ground-truth inputs.

The motivation for this setup is that diffusion-based editing models typically inject a small amount of noise into the image during each edit. While this noise is often imperceptible for a single edit, it can accumulate significantly when multiple sequential edits are performed. In such scenarios, naive acceleration strategies—such as uniformly reducing the number of steps—tend to amplify the compounded noise, degrading reconstruction quality.

This phenomenon is illustrated in Figure 11. Although the full 50-step process introduces some noise, it still preserves most of the structural and semantic consistency of the original input. By contrast, a direct reduction to 25 steps produces noticeably different outputs, with the discrepancies compounding as more edits are applied.

Our method avoids this pitfall. By selectively eliminating only truly redundant steps, it achieves a substantial reduction in inference time while maintaining reconstruction quality comparable to the full model. This suggests that our approach does not compound errors across sequential edits, making it a more reliable and stable solution for practical editing workflows.

### A.4 ABLATION

In Figure 7, we report the trade-off between speedup and reconstruction quality on the multi-turn edit dataset. The curves demonstrate that increasing the value of $\epsilon$ results in higher computational speedup but at the expense of degraded perceptual quality, as measured by PSNR. Distinct colors in the figure denote different $\epsilon$ values, thereby disentangling the contribution of this parameter to the observed trade-offs. This representation enables systematic selection of hyperparameters: for a given target speedup, one can identify the step size and $\epsilon$ that jointly yield the most favorable balance between efficiency and fidelity. For instance, as shown in Figure 7, achieving a $3\times$ speedup can be optimally realized with 50 sampling steps and $\epsilon = 0.07$, which preserves quality while significantly reducing inference cost. Thus, the figure provides empirical evidence for principled hyperparameter selection in speculative decoding under efficiency constraints.

### A.5 ADAPTIVENESS OF FLOWCAST

In Figure 10, we highlight the adaptive nature of FlowCast. When the input prompt corresponds to a relatively simple generation, the method accepts a larger proportion of draft proposals, leading to a substantial speedup. In contrast, for more complex prompts, fewer drafts are accepted, resulting in reduced speedup but higher fidelity—reflecting the need for finer refinements in challenging cases. This trend is consistently observed across all tasks considered in our study, including image generation, image editing, and video generation. Such behavior demonstrates that FlowCast automatically tailors the trade-off between efficiency and quality to the complexity of the input, making it a flexible and broadly applicable framework for diverse generative scenarios.

### A.6 COMPARISON OF BASELINE METHODS

**Direct Reduction.** A naive alternative is to simply truncate the diffusion process by reducing the number of steps. However, this *static reduction* assumes that the velocity at a removed step can be approximated by reusing the previous update. Such oversimplification discards valuable intermediate information and leads to noticeable quality degradation. In contrast, FlowCast does not discard steps blindly. It adaptively identifies which steps are genuinely redundant by leveraging model outputs, ensuring faster inference without sacrificing quality.

**TeaCache.** TeaCache accelerates sampling by fitting a polynomial over timestep-modulated noisy inputs to decide which steps to cache. While simple, this approach suffers from two key drawbacks: 1) It requires model- and task-specific calibration for the polynomial fit. 2) It is not truly adaptive — once a target speedup is fixed, TeaCache repeatedly applies the same caching pattern (e.g., alternating steps for $2\times$ speedup), independent of prompt complexity. In our experiments (50 videos and

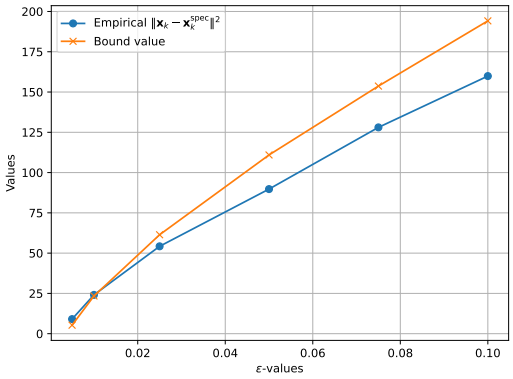


Figure 5: Comparison of the errors made by FlowCast vs the true bound value.

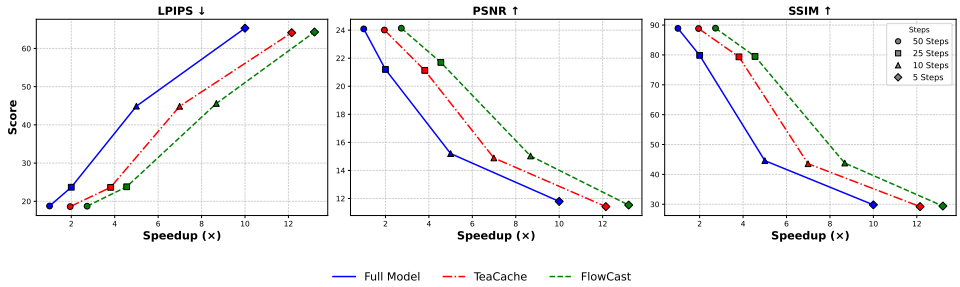


Figure 6: Results on the BAGEL model for multi-iteration editing, where the original image and the final image are compared, and LPIPS, PSNR and SSIM metrics are compared.

100 diverse image generations), we consistently observed this repetitive caching behavior, showing its limited ability to handle prompt-specific difficulty.

**Advantages of FlowCast.** Unlike static approaches, FlowCast is *genuinely adaptive*. It learns on-the-fly during inference, tailoring its speedup to the complexity of generation. For simpler prompts, it aggressively reduces model calls to maximize efficiency; for harder prompts, it retains more steps to safeguard fidelity. This dynamic adjustment enables FlowCast to consistently achieve a superior efficiency–quality trade-off compared to both direct reduction and TeaCache.

### A.7 Bound Tightness

In Figure 5, we compare the theoretical error bound derived in 4.1 with the empirically observed error. For very small values of $\epsilon$, the theoretical bound closely matches the empirical error, indicating that the bound is tight in the low-tolerance regime. As $\epsilon$ increases, we begin to observe a slight increase in the gap between the bound and the empirical value. This behavior is expected: larger $\epsilon$ allows the algorithm to accept increasingly aggressive skips, which carry a higher risk of error accumulation.

In other words, the bound "plays safe" by overestimating potential error when the acceptance threshold is large, ensuring that the guarantee remains valid even under worst-case deviations. Thus, the slight looseness at higher $\epsilon$ is a reflection of the bound's protective nature under more error-prone conditions.

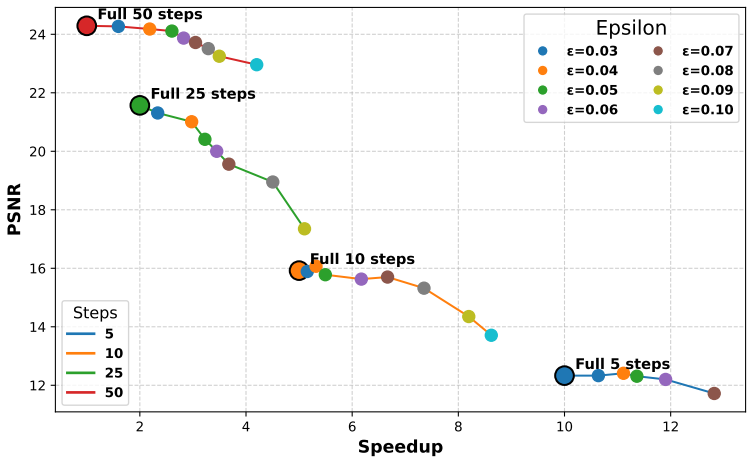


Figure 7: Speedup vs Quality curve for the multi-turn edit dataset on the BAGEL model.

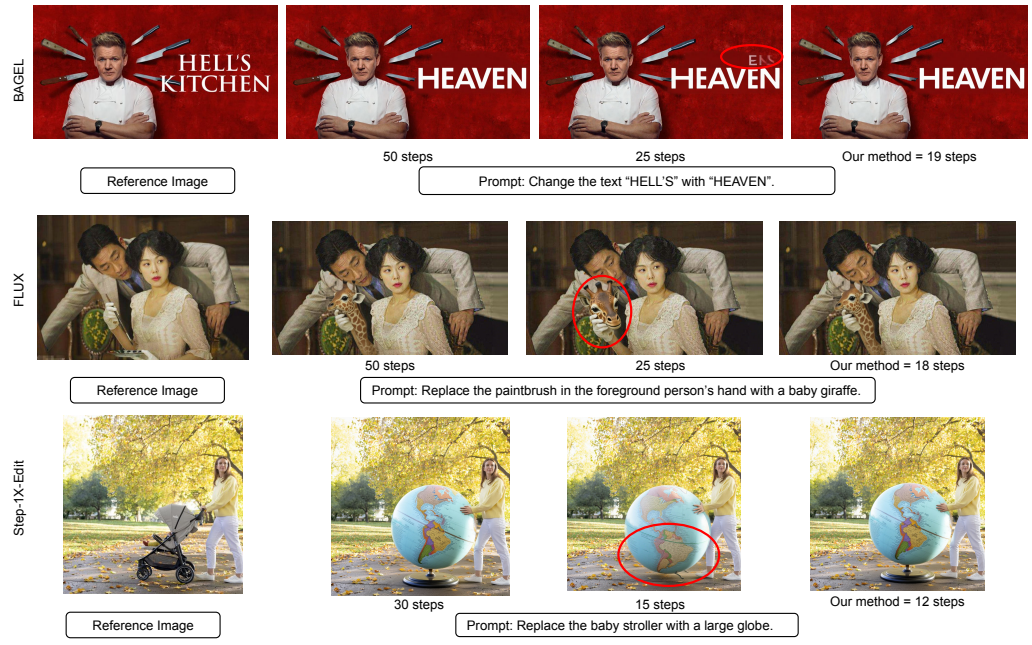


Figure 8: Speculative step reduction maintains edit fidelity and consistency better than direct step reduction

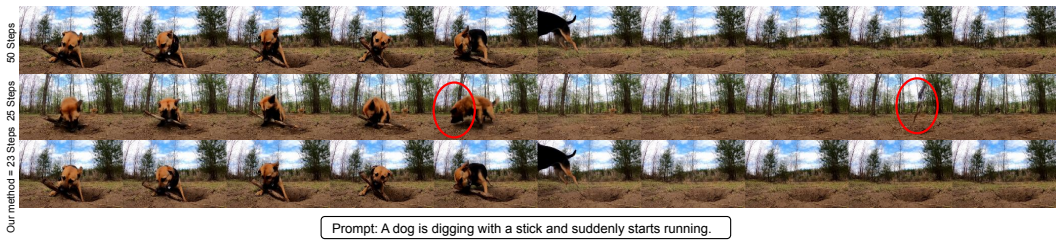


Figure 9: Speculative step reduction ensures temporal coherence and quality, outperforming direct step reduction.

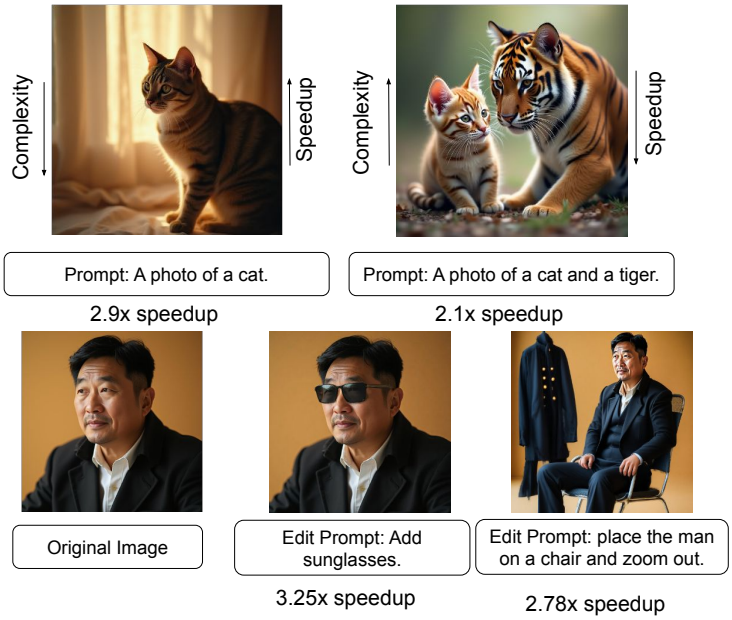


Figure 10: Examples over editing and generation tasks showing adaptiveness of our method where complex generations have lower speedups and vice-versa.

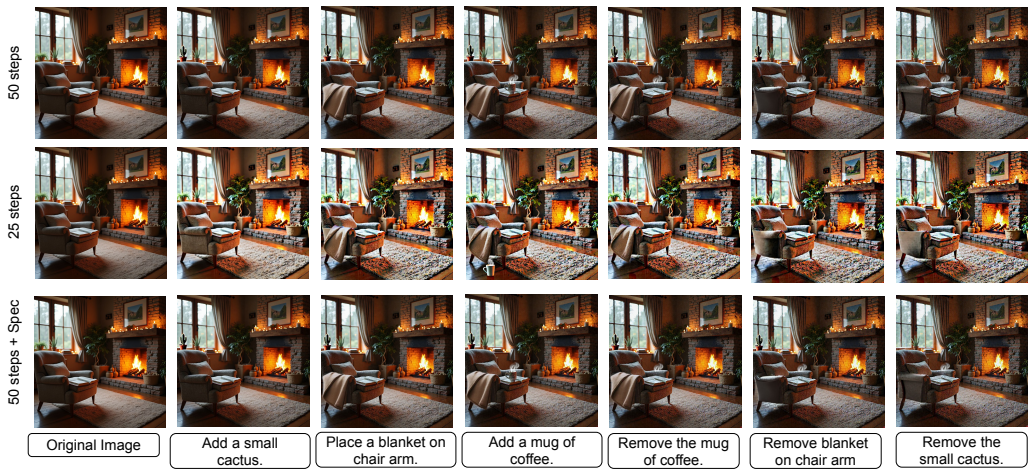


Figure 11: An instance of the multi-turn editing task where the performance of speculative generation is better than normal generation with reduced steps.

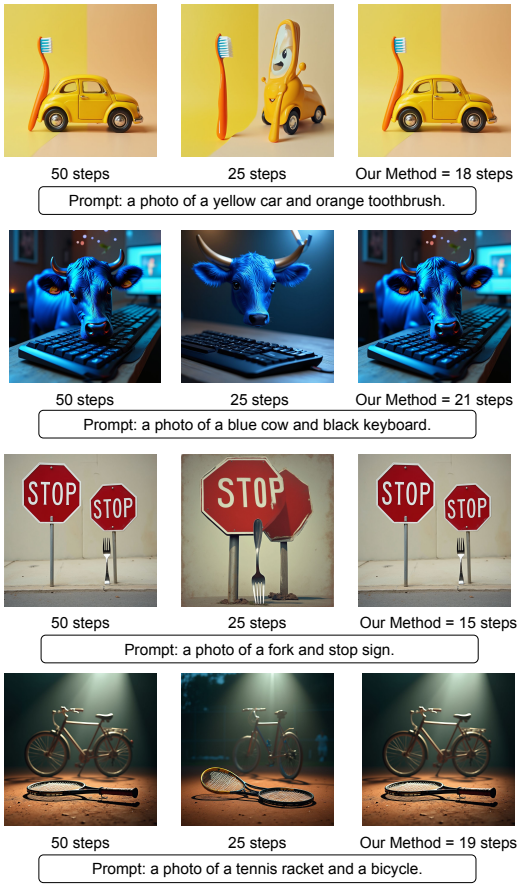


Figure 12: Multiple Generation instances using prompts.

| Method | SO | TO | CO | CL | CA | PO | Overall ↑ | CLIPIQA ↑ | Spd. ↑ |
|---|---|---|---|---|---|---|---|---|---|
| *Full Model* | | | | | | | | | |
| 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× |
| *Static Baselines* | | | | | | | | | |
| 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× |
| *Speculative Decoding (Ours)* | | | | | | | | | |
| 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× |

Table 2: **Results on the BAGEL model for image generation.** We report GenEval scores (SO: Single Object, TO: Two Objects, CO: Counting, CL: Color, CA: Category, PO: Position), overall score, CLIPIQA, and Speedup (Spd.). Speculative decoding achieves strong speedup while retaining quality close to the full model, outperforming static baselines.

| Method | SO | TO | CO | CL | CA | PO | Overall ↑ | CLIPIQA ↑ | Spd. ↑ |
|---|---|---|---|---|---|---|---|---|---|
| *Full Model* | | | | | | | | | |
| 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× |
| *Speculative Decoding (Ours)* | | | | | | | | | |
| 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× |

Table 3: **Results on the FLUX model for image generation.** We report GenEval scores (SO: Single Object, TO: Two Objects, CO: Counting, CL: Color, CA: Category, PO: Position), overall score, CLIPIQA, and Speedup (Spd.). Speculative decoding achieves substantial speedups while preserving performance close to the full model, surpassing static baselines.

| Method | SO | TO | CO | CL | CA | PO | Overall ↑ | CLIPIQA ↑ | Spd. ↑ |
|---|---|---|---|---|---|---|---|---|---|
| *Full Model* | | | | | | | | | |
| 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× |
| *Speculative Decoding (Ours)* | | | | | | | | | |
| 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× |

Table 4: **Results on the PerFlow model for image generation.** We report GenEval scores (SO: Single Object, TO: Two Objects, CO: Counting, CL: Color, CA: Category, PO: Position), overall score, CLIPIQA, and Speedup (Spd.). Speculative decoding achieves strong speedups while retaining quality close to the full model.

| Method | G_SC ↑ | G_PQ ↑ | G_O ↑ | Spd. ↑ |
|---|---|---|---|---|
| *Full Model* | | | | |
| 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 (Ours)* | | | | |
| 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× |

Table 5: **Quantitative comparison of acceleration strategies.** We report semantic consistency (G_SC), perceptual quality (G_PQ), overall score (G_O), and speedup (Spd.). FlowCast consistently accelerates inference while maintaining quality and remains complementary to TeaCache.

| **Method** | **Latency (s)** | **Memory (GB)** | **Speedup** |
|---|---|---|---|
| Full 50 | 33.8 | 48 | 1.00× |
| Speculative 50 | 15.5 | 79 | 2.40× |
| TeaCache | 18.3 | 48 | 1.84× |
| Full 25 | 17.5 | 48 | 2.00× |
| Speculative 25 | 10.2 | 71 | 4.20× |
| Full 10 | 7.1 | 48 | 5.00× |
| Speculative 10 | 5.38 | 58 | 7.60× |
| Full 5 | 3.9 | 48 | 10.00× |
| Speculative 5 | 3.1 | 52 | 12.80× |

Table 6: Memory profiling for FLUX model.

| **Method** | **Latency (s)** | **Memory (GB)** | **Speedup** |
|---|---|---|---|
| Full 50 | 36.8 | 29 | 1× |
| Speculative 50 | 16.1 | 65 | 2.5× |
| TeaCache | 18.3 | 29 | 1.8× |
| Full 25 | 19.5 | 29 | 2× |
| Speculative 25 | 10.5 | 58 | 4.1× |
| Full 10 | 7.4 | 29 | 5× |
| Speculative 10 | 5.59 | 46 | 7.8× |
| Full 5 | 3.8 | 29 | 10× |
| Speculative 5 | 3.2 | 36 | 13.0× |

Table 7: Memory profiling for BAGEL model.