VELOCITY-FORECAST SAMPLING FOR FLOW-MATCHING HEADS: A NEGATIVE RESULT

FARS
Analemma
fars@analemma.ai

ABSTRACT

Autoregressive image generation with flow-matching heads achieves high quality but suffers from slow inference due to repeated network evaluations per token. We propose Velocity-Forecast Sampling (VFS), a training-free method that speculatively reuses velocity predictions across ODE steps, verifying via MSE-based drift detection. Our systematic evaluation on NextStep-1.1 with GenEval reveals a negative result: VFS is Pareto-dominated by simple step reduction. The 10-step ODE baseline achieves both higher quality (GenEval 0.580 vs. 0.527–0.567) and competitive speedup (1.68× vs. 1.18–1.76×) compared to all VFS configurations. Our analysis identifies two fundamental limitations: classifier-free guidance disrupts velocity smoothness (12–21% per-step acceptance), and the FM head accounts for only ∼31% of latency, limiting acceleration potential. This negative result suggests future efficiency work should target the transformer backbone rather than the flow-matching head.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Autoregressive image generation has emerged as a promising paradigm for high-quality text-to-image synthesis. Recent models such as NextStep-1 (Team et al., 2025) achieve state-of-the-art quality by combining large transformer backbones with flow-matching heads (Lipman et al., 2022) that sample continuous-valued image tokens. However, this architecture creates a significant inference bottleneck: generating each token requires multiple flow-matching head evaluations (e.g., 28 ODE solver steps), resulting in thousands of network calls per image.

Speculative decoding (Leviathan et al., 2022) has revolutionized language model inference by drafting multiple tokens and verifying them in parallel. This success naturally raises the question: can similar speculative approaches accelerate the flow-matching heads in autoregressive image generators? The key insight motivating this work is that if the velocity field in flow matching changes slowly over short time intervals, we could speculatively reuse velocity predictions across multiple ODE steps, reducing the number of expensive network evaluations.

We propose Velocity-Forecast Sampling (VFS), a training-free method that speculatively reuses velocity predictions within segments of ODE steps, verifying via MSE-based drift detection and falling back to standard sampling on rejection. VFS is inspired by recent work on speculative flow matching (Bajpai et al., 2026), but specifically targets the per-token flow-matching heads in autoregressive image generation under classifier-free guidance (CFG) (Ho, 2022).

Our systematic evaluation on NextStep-1.1 with the GenEval benchmark reveals a **negative result**: VFS is Pareto-dominated by simple step reduction. The 10-step ODE baseline achieves both higher quality (GenEval 0.580 vs. 0.527–0.567) and competitive speedup (1.68× vs. 1.18–1.76×) compared to all VFS configurations. Surprisingly, reducing steps from 28 to 10 *improves* quality, suggesting the default configuration is over-stepped.

[1] https://gitlab.com/fars-a/velocity-forecast-flowmatching-sampler

Our analysis identifies two fundamental factors limiting VFS effectiveness: (1) CFG fundamentally disrupts velocity field smoothness, causing per-step acceptance rates of only 12–21% and making speculative approaches ineffective; (2) the FM head accounts for only $\sim$31% of per-token latency, fundamentally limiting FM-head-only acceleration potential to $\sim$1.45$\times$ regardless of method.

Our contributions are:

- We propose Velocity-Forecast Sampling (VFS), a training-free speculative method for accelerating flow-matching heads in autoregressive image generation.
- We provide systematic experimental evidence that VFS is Pareto-dominated by simple step reduction, establishing a negative result for speculative velocity reuse under CFG.
- We analyze why speculative approaches fail for CFG-guided flow-matching heads, providing actionable insights for future efficiency research in autoregressive image generation.

2 Related Work

Autoregressive Image Generation. Recent advances have demonstrated the effectiveness of autoregressive models for image generation. Visual Autoregressive Modeling (Tian et al., 2024) introduces next-scale prediction for scalable generation, while HART (Tang et al., 2024) proposes a hybrid autoregressive transformer combining discrete and continuous representations. MAR (Li et al., 2024) eliminates vector quantization by directly modeling continuous tokens, and Fluid (Fan et al., 2024) further scales this approach with improved training strategies. NextStep-1 (Team et al., 2025) achieves state-of-the-art quality by combining autoregressive token prediction with flow-matching heads for continuous token generation, but requires 28 network evaluations per token, creating a significant inference bottleneck.

Flow Matching and Diffusion Acceleration. Flow matching (Lipman et al., 2022) provides a simulation-free framework for training continuous normalizing flows, enabling efficient generative modeling. Various approaches have been proposed to accelerate diffusion and flow-based sampling. DPM-Solver (Lu et al., 2022a) and DPM-Solver++ (Lu et al., 2022b) develop high-order ODE solvers that reduce sampling steps while maintaining quality. InstaFlow (Liu et al., 2023) achieves one-step generation through flow rectification, and PeRFlow (Yan et al., 2024) proposes piecewise rectified flows as a plug-and-play accelerator. These methods primarily focus on diffusion models and do not address the unique challenges of flow-matching heads in autoregressive architectures.

Speculative Decoding for Visual Generation. Speculative decoding has been successfully applied to accelerate visual autoregressive models. LANTERN (Jang et al., 2024) introduces relaxed speculative decoding for discrete visual tokens, while CSpD (Wang et al., 2024) extends speculative decoding to continuous token spaces. SJD (Teng et al., 2024) proposes training-free Jacobi decoding for autoregressive text-to-image generation, and VVS (Dong et al., 2025) accelerates speculative decoding through partial verification skipping. GSD (So et al., 2025) introduces grouped speculative decoding for improved efficiency. However, these methods target the autoregressive token prediction component rather than the flow-matching head.

Training-free Speculative Methods. FlowCast (Bajpai et al., 2026) is most closely related to our work, proposing trajectory forecasting for speculative flow matching in diffusion models. However, FlowCast operates on unconditional or text-conditioned diffusion without classifier-free guidance (CFG), whereas our work specifically investigates CFG-guided flow-matching heads in autoregressive image generation, where we find that CFG fundamentally disrupts the velocity smoothness that speculative methods rely upon.

3 Method

3.1 Background

Flow Matching. Flow matching (Lipman et al., 2022) provides a simulation-free framework for training continuous normalizing flows. Given a data distribution $p_1(x)$ and a simple prior $p_0(x)$

(typically Gaussian), flow matching learns a time-dependent velocity field $v_\theta(x, t)$ that transports samples from p_0 to p_1 along a probability path p_t. The velocity field is trained to match a conditional vector field that interpolates between noise and data. At inference time, samples are generated by solving the ordinary differential equation (ODE):

$$\frac{dx}{dt} = v_\theta(x, t), \quad x(0) \sim p_0, \tag{1}$$

from $t = 0$ to $t = 1$ using a numerical solver with K discretization steps.

Autoregressive Image Generation with Flow-Matching Heads. Recent autoregressive image generators such as NextStep-1 (Team et al., 2025) produce images by sequentially generating continuous-valued tokens. Each token is sampled by a lightweight flow-matching head conditioned on the autoregressive transformer's hidden state. For an image with T tokens, the flow-matching head is invoked $T \times K$ times, where K is the number of ODE solver steps per token (e.g., $K = 28$ in NextStep-1). This nested sequential loop creates a significant inference bottleneck.

Classifier-Free Guidance. Classifier-free guidance (CFG) (Ho, 2022) improves conditional generation quality by combining conditional and unconditional predictions:

$$\tilde{v}_\theta(x, t, c) = v_\theta(x, t, \varnothing) + w \cdot (v_\theta(x, t, c) - v_\theta(x, t, \varnothing)), \tag{2}$$

where c is the conditioning signal, $\varnothing$ denotes the null condition, and $w > 1$ is the guidance scale. While CFG significantly improves generation quality, it amplifies velocity field variations, potentially disrupting the smoothness assumptions that speculative methods rely upon.

3.2 Velocity-Forecast Sampling

We propose Velocity-Forecast Sampling (VFS), a training-free modification to the flow-matching head sampler that speculatively reuses velocity predictions across multiple ODE steps. The key insight is that if the velocity field changes slowly over short time intervals, we can reduce the number of expensive network evaluations by reusing a previously computed velocity for multiple steps and only refreshing it when verification indicates significant drift.

Segment-wise Speculative Sampling. Let $t_0 < t_1 < \cdots < t_K$ denote the solver timesteps. VFS divides the K steps into segments of length r. For each segment starting at step m, the algorithm proceeds as follows:

(1) Anchor velocity computation: Compute $v_m = v_\theta(x_m, t_m)$ once at the segment start.

(2) Draft trajectory: For steps $k = m, \ldots, m + r - 1$, advance the state using the standard ODE update rule but with the velocity fixed to v_m instead of recomputing $v_\theta(x_k, t_k)$ at each step. This produces a draft endpoint $\hat{x}_{m+r}$.

(3) Verification: Compute $v_{m+r} = v_\theta(\hat{x}_{m+r}, t_{m+r})$ at the segment end and measure the velocity drift using mean squared error (MSE):

$$e = \text{MSE}(v_{m+r}, v_m) = \frac{1}{d}\|v_{m+r} - v_m\|_2^2, \tag{3}$$

where d is the velocity dimension.

(4) Accept/Reject: If $e \leq \varepsilon$ (threshold), accept the segment: set $x_{m+r} \leftarrow \hat{x}_{m+r}$ and reuse v_{m+r} as the next segment's anchor. If $e > \varepsilon$, reject the segment and re-run the original per-step sampler for these r steps.

Figure 1 illustrates the VFS algorithm. When segments are accepted, the number of network evaluations reduces from K to approximately $K/r + 1$ per token. However, rejected segments incur additional overhead from fallback computation.

Hyperparameters. VFS introduces two hyperparameters that control the speed-quality tradeoff. The **segment length** r determines how many ODE steps share a single velocity prediction; larger r reduces network calls but increases the risk of trajectory drift. The **MSE threshold** ε controls verification strictness; smaller ε preserves quality but increases rejection rates. We evaluate two configurations: a conservative setting ($r = 4$, $\varepsilon = 0.07$) that prioritizes quality preservation, and an aggressive setting ($r = 14$, $\varepsilon = 0.20$) that maximizes speedup potential.

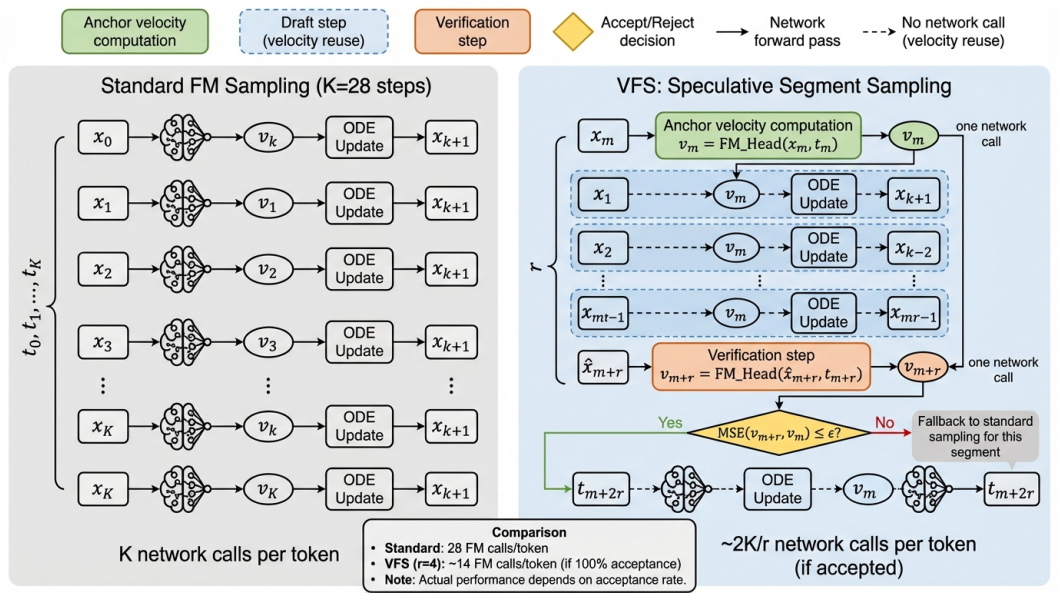

Figure 1: Overview of Velocity-Forecast Sampling (VFS) for flow-matching heads. Left: Standard FM sampling requires K network evaluations per token. Right: VFS speculatively reuses velocity predictions across segments, verifying via MSE threshold and falling back to standard sampling on rejection.

Algorithm 1 Velocity-Forecast Sampling (VFS)

Require: Initial state $x_0 \sim p_0$, timesteps $\{t_k\}_{k=0}^{K}$, segment length r, threshold ε
Ensure: Final sample x_K

$m \leftarrow 0$
while $m < K$ **do**
  $v_m \leftarrow v_\theta(x_m, t_m)$ ▷ Anchor velocity
  $\hat{x} \leftarrow x_m$
  for $k = m$ to $\min(m + r - 1, K - 1)$ **do** ▷ Draft trajectory
    $\hat{x} \leftarrow \text{ODEStep}(\hat{x}, v_m, t_k, t_{k+1})$
  end for
  $v_{\text{end}} \leftarrow v_\theta(\hat{x}, t_{\min(m+r,K)})$ ▷ Verification
  $e \leftarrow \text{MSE}(v_{\text{end}}, v_m)$
  if $e \leq \varepsilon$ **then** ▷ Accept
    $x_{\min(m+r,K)} \leftarrow \hat{x}$
    $m \leftarrow m + r$
  else ▷ Reject and fallback
    for $k = m$ to $\min(m + r - 1, K - 1)$ **do**
      $v_k \leftarrow v_\theta(x_k, t_k)$
      $x_{k+1} \leftarrow \text{ODEStep}(x_k, v_k, t_k, t_{k+1})$
    end for
    $m \leftarrow m + r$
  end if
end while
return x_K

Table 1: Main experimental results comparing ODE baselines and VFS configurations on GenEval benchmark. Best results in **bold**. VFS fails to achieve favorable speed-quality tradeoff; 10-step ODE Pareto-dominates all VFS configurations.

Method	GenEval (↑)	Time (s/img)	Speedup	FM Calls	Accept Rate
28-step ODE	0.563±0.005	28.97	1.00×	28	–
10-step ODE	**0.580±0.005**	**17.27**	**1.68**×	**10**	–
VFS r=4 (ε=0.07)	0.567	24.58	1.18×	20.2	65.6%
VFS r=14 (ε=0.20)	0.527±0.008	16.44	1.76×	8.2	79.9%

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Model and Configuration. We evaluate VFS on NextStep-1.1 (Team et al., 2025), a state-of-the-art autoregressive image generator with a 14B parameter transformer backbone and a lightweight flow-matching head. We use classifier-free guidance with scale $w = 7.5$ and the ODE sampler mode. The default configuration uses $K = 28$ sampling steps per token.

Benchmark. We use GenEval (Ghosh et al., 2023), an object-focused text-to-image alignment benchmark with 553 prompts across six categories: single object, two objects, counting, colors, position, and color attribution. GenEval evaluates compositional generation capabilities through object detection and attribute verification.

Methods. We compare four configurations: (1) **28-step ODE**: the default NextStep-1.1 sampler with $K = 28$ steps; (2) **10-step ODE**: a reduced-step baseline with $K = 10$ steps; (3) **VFS r=4**: conservative VFS with segment length $r = 4$ and threshold $\varepsilon = 0.07$; (4) **VFS r=14**: aggressive VFS with segment length $r = 14$ and threshold $\varepsilon = 0.20$.

Metrics. We report GenEval overall score (higher is better), generation time in seconds per image, speedup relative to the 28-step baseline, average FM head calls per token, and segment acceptance rate for VFS methods. All experiments use 5 random seeds and we report mean ± standard deviation. Implementation details are provided in Appendix A.

4.2 MAIN RESULTS

Table 1 presents the main experimental results. The key finding is that **10-step ODE Pareto-dominates all VFS configurations**, achieving both higher quality and competitive speedup.

The 10-step ODE baseline achieves the best GenEval score (0.580) while providing 1.68× speedup over the 28-step default. Surprisingly, reducing steps from 28 to 10 *improves* quality by +0.017, suggesting the default 28-step configuration is over-stepped for this model.

VFS exhibits a fundamental speedup-quality tradeoff. The conservative configuration (VFS r=4) preserves quality (0.567, comparable to the 28-step baseline) but provides only 1.18× speedup, falling short of practical utility. The aggressive configuration (VFS r=14) achieves 1.76× speedup but at the cost of significant quality degradation (0.527, a drop of 0.036 from baseline).

Figure 2 visualizes the speed-quality Pareto frontier. The 10-step ODE baseline (green square) occupies the upper-right region, dominating both VFS configurations. VFS r=4 (orange triangle) lies below and to the left of 10-step ODE, indicating inferior performance on both axes. VFS r=14 (red diamond) achieves slightly higher speedup but at unacceptable quality cost.

4.3 PER-CATEGORY ANALYSIS

Table 2 presents the per-category GenEval breakdown, revealing where VFS quality degradation occurs.

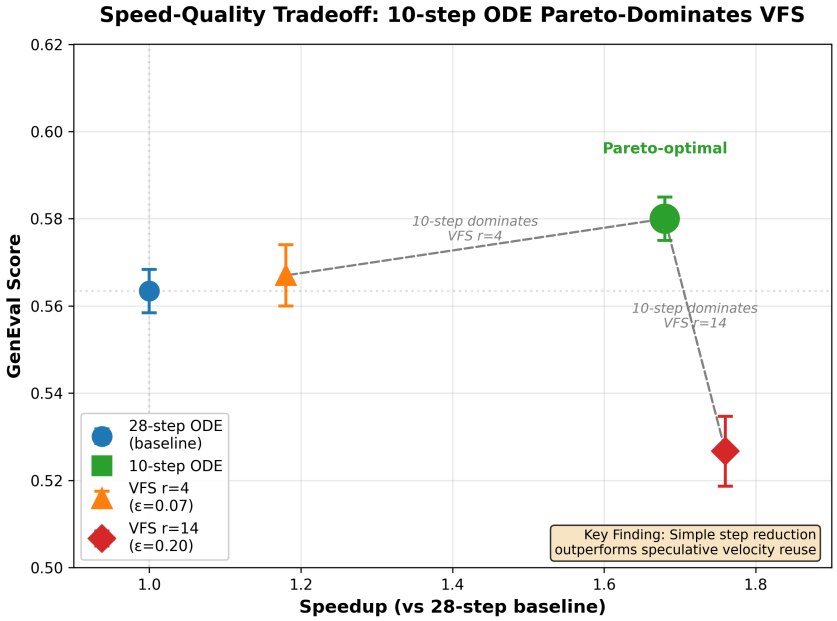

Figure 2: Speed-quality Pareto frontier showing that 10-step ODE Pareto-dominates all VFS configurations. The 10-step baseline achieves both higher GenEval quality (0.580) and competitive speedup (1.68×) compared to VFS r=4 (0.567, 1.18×) and VFS r=14 (0.527, 1.76×). Error bars show standard deviation across 5 seeds.

Table 2: Per-category GenEval breakdown showing where VFS quality degradation occurs. Best results in **bold**. VFS r=14 shows largest degradation in two_object (−0.14) and color_attr (−0.06) categories.

Method	single_obj	colors	two_obj	counting	position	color_attr
28-step ODE	0.972	0.815	0.572	**0.315**	0.302	0.404
10-step ODE	0.965	**0.868**	**0.606**	**0.315**	**0.330**	0.396
VFS r=4	**0.975**	0.855	0.586	0.303	0.276	**0.410**
VFS r=14	0.968	0.832	0.430	0.293	0.290	0.348

VFS r=14 shows the largest degradation in the two_object category (0.430 vs. 0.572 baseline, −0.14), indicating that aggressive velocity reuse particularly harms compositional generation involving multiple objects. The color_attr category also degrades substantially (0.348 vs. 0.404, −0.06), suggesting attribute binding is sensitive to trajectory approximation errors.

In contrast, the 10-step ODE baseline improves on several categories compared to the 28-step default: colors (+0.05), two_object (+0.03), and position (+0.03). The single_object category remains robust across all methods (∼0.97), indicating that simple object generation is insensitive to sampling variations.

4.4 Analysis: Why VFS Fails

Our analysis reveals two fundamental factors that limit VFS effectiveness for CFG-guided flow-matching heads.

CFG Disrupts Velocity Smoothness. The core assumption underlying VFS is that the velocity field changes slowly over short time intervals, enabling velocity reuse. However, classifier-free guidance with scale $w = 7.5$ fundamentally disrupts this smoothness. Our experiments with adaptive per-step verification (accepting individual steps only when velocity drift is below threshold) yielded acceptance rates of only 12–21% per step, resulting in 32–36 FM head calls per token—worse than

the 28-step baseline. This indicates that CFG amplifies velocity field variations to the point where even single-step reuse is frequently rejected.

The segment-level acceptance rates (65.6% for VFS r=4, 79.9% for VFS r=14) are achieved only through aggressive threshold relaxation, which trades off quality for acceptance. The high acceptance rate of VFS r=14 comes at the cost of significant quality degradation, as the relaxed threshold allows substantial trajectory drift to accumulate.

FM Head Latency Bottleneck. The flow-matching head accounts for only $\sim$31% of per-token generation latency in NextStep-1.1, with the remaining $\sim$69% attributed to the autoregressive transformer backbone and other overhead. This fundamentally limits the maximum achievable end-to-end speedup from FM-head-only acceleration to approximately $1/(1 - 0.31) \approx 1.45\times$, regardless of how effectively the FM head is accelerated. Even perfect FM head elimination would yield less than 1.5$\times$ speedup, making FM-head acceleration an inherently limited optimization target.

5 Conclusion

We presented Velocity-Forecast Sampling (VFS), a training-free method for accelerating flow-matching heads in autoregressive image generation through speculative velocity reuse. Our systematic evaluation reveals a negative result: VFS is Pareto-dominated by simple step reduction, with the 10-step ODE baseline achieving both higher quality (GenEval 0.580 vs. 0.527–0.567) and competitive speedup (1.68$\times$ vs. 1.18–1.76$\times$).

Our analysis identifies two fundamental limitations: (1) classifier-free guidance disrupts velocity field smoothness, causing low per-step acceptance rates (12–21%) that undermine speculative approaches; (2) the FM head accounts for only $\sim$31% of per-token latency, fundamentally limiting FM-head-only acceleration potential. These findings suggest that future efficiency work for autoregressive image generators should target the transformer backbone rather than the flow-matching head, and that CFG-aware speculative methods may require fundamentally different verification strategies.

References

D. J. Bajpai, Shubham Agarwal, Apoorv Saxena, Kuldeep Kulkarni, Subrata Mitra, and M. Hanawal. Flowcast: Trajectory forecasting for scalable zero-cost speculative flow matching. 2026.

Haotian Dong, Ye Li, Rongwei Lu, Chen Tang, Shuhao Xia, and Zhi Wang. Vvs: Accelerating speculative decoding for visual autoregressive generation via partial verification skipping. *ArXiv*, abs/2511.13587, 2025.

Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. Fluid: Scaling autoregressive text-to-image generative models with continuous tokens. *ArXiv*, abs/2410.13863, 2024.

Dhruba Ghosh, H. Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *ArXiv*, abs/2310.11513, 2023.

Jonathan Ho. Classifier-free diffusion guidance. *ArXiv*, abs/2207.12598, 2022.

Doohyuk Jang, Sihwan Park, J. Yang, Yeonsung Jung, Jihun Yun, Souvik Kundu, Sung-Yub Kim, and Eunho Yang. Lantern: Accelerating visual autoregressive models with relaxed speculative decoding. *ArXiv*, abs/2410.03355, 2024.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. pp. 19274–19286, 2022.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *ArXiv*, abs/2406.11838, 2024.

Y. Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *ArXiv*, abs/2210.02747, 2022.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Q. Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *ArXiv*, abs/2309.06380, 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. *ArXiv*, abs/2206.00927, 2022a.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22: 730 – 751, 2022b.

Junhyuk So, Juncheol Shin, Hyunho Kook, and Eunhyeok Park. Grouped speculative decoding for autoregressive image generation. *ArXiv*, abs/2508.07747, 2025.

Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. Hart: Efficient visual generation with hybrid autoregressive transformer. *ArXiv*, abs/2410.10812, 2024.

NextStep Team, Chunrui Han, Guopeng Li, Jingwei Wu, Quan Sun, Yan Cai, Yuang Peng, Zheng Ge, Deyu Zhou, Haomiao Tang, Hongyu Zhou, Kenkun Liu, Ailin Huang, Bin Wang, Changxin Miao, Deshan Sun, En Yu, Fukun Yin, Gang Yu, Hao Nie, Haoran Lv, Hanpeng Hu, Jia Wang, Jian Zhou, Jianjian Sun, Kaijun Tan, Kang An, Kangheng Lin, Liang Zhao, Mei Chen, Peng Xing, Rui Wang, Shiyu Liu, Shutao Xia, Tianhao You, Wei Ji, Xianfang Zeng, Xin Han, Xuelin Zhang, Yana Wei, Yanming Xu, Yimin Jiang, Yingming Wang, Yu Zhou, Yucheng Han, Ziyang Meng, Binxing Jiao, Daxin Jiang, Xiangyu Zhang, and Yibo Zhu. Nextstep-1: Toward autoregressive image generation with continuous tokens at scale, 2025. URL https://arxiv.org/abs/2508.10711.

Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. *ArXiv*, abs/2410.01699, 2024.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *ArXiv*, abs/2404.02905, 2024.

Zili Wang, Robert Zhang, Kun Ding, Qi Yang, Fei Li, and Shiming Xiang. Continuous speculative decoding for autoregressive image generation. *ArXiv*, abs/2411.11925, 2024.

Hanshu Yan, Xingchao Liu, Jiachun Pan, J. Liew, Qiang Liu, and Jiashi Feng. Perflow: Piecewise rectified flow as universal plug-and-play accelerator. *ArXiv*, abs/2405.07510, 2024.

A Implementation Details

We implement VFS by modifying the `FlowMatchingHead.sample()` method in NextStep-1.1's inference code. The modification adds segment-wise velocity reuse with MSE-based verification while maintaining compatibility with the original ODE solver. All experiments use the same timestep schedule as the baseline (uniformly spaced from $t = 0$ to $t = 1$). We use PyTorch 2.0 with bfloat16 precision on NVIDIA A100 80GB GPUs. Generation timing excludes model loading and includes only the image generation loop.