VERIFIER THRESHOLD: AN EFFICIENT TEST-TIME SCALING APPROACH FOR IMAGE GENERATION

Vignesh Sundaresha
University of Illinois Urbana-Champaign
vs49@illinois.edu

Akash Haridas
AMD
akash.haridas@amd.com

Vikram Appia
AMD

Lav R. Varshney
Stony Brook University

ABSTRACT

Image generation has emerged as a mainstream application of large generative models. Just as test-time compute and reasoning have improved language model capabilities, similar benefits have been observed for image generation models. In particular, searching over noise samples for diffusion and flow models has been shown to scale well with test-time compute. While recent works explore allocating non-uniform inference-compute budgets across denoising steps, existing approaches rely on greedy heuristics and often allocate the compute budget ineffectively. In this work, we study this problem and propose a simple fix. We propose *Verifier-Threshold*, which automatically reallocates test-time compute and delivers substantial efficiency improvements. For the same performance on the GenEval benchmark, we achieve a $2\times$- $4\times$ reduction in computational time over the state-of-the-art method.

1 INTRODUCTION

Image generation has become a popular application of AI, with several models DeepMind (2025); OpenAI (2025); Apple Machine Learning Research (2025); Midjourney (2025); Stability AI (2024) now widely used. However, even state-of-the-art models often struggle to accurately follow detailed prompt instructions Ghosh et al. (2023); Huang et al. (2023). To address this limitation, the vision community has drawn inspiration from language models by incorporating test-time compute Snell et al. (2024) and reasoning Guo et al. (2025), leading to the development of analogous approaches for vision tasks Ma et al. (2025); Zhuo et al. (2025). Reasoning-based methods, however, inherently demand substantial computation and wall-clock time at inference, which becomes costly at scale Jin et al. (2025) or on resource-constrained devices Stelia (2025). Consequently, improving the efficiency of test-time compute algorithms is essential.

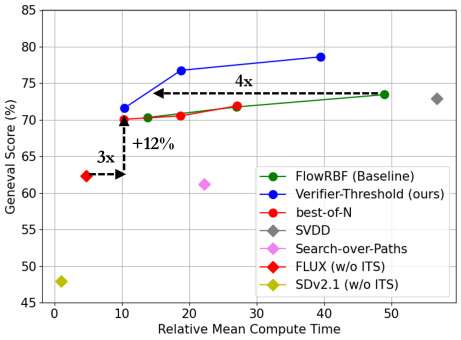


Figure 1: Test-time scaling; the horizontal axis represents relative wall-clock duration on an AMD MI300X GPU averaged across 553 GenEval Ghosh et al. (2023) prompts and the vertical axis represents the overall GenEval score. VQAScore Lin et al. (2024) is used as the verifier and FLUX-Schnell Labs et al. (2025) as the generator for all methods. *Verifier-Threshold* is consistently more efficient and delivers higher scores than the baseline FlowRBF and other methods.

Fig. 1 illustrates how test-time scaling increases computational requirements. The baseline implementation Kim et al. (2025) requires between $3\times$ and $10\times$ more compute time than regular image generation to achieve GenEval Ghosh et al. (2023) gains of 8% to 12%, respectively. In contrast, our proposed method, *Verifier-Threshold*, reaches the same 12% improvement using only $(1/4)^{\text{th}}$ of the baseline's computational cost, clearly demonstrating its efficiency. Furthermore, our algorithm continues to benefit from scaling, achieving gains of up to 15%.

The key contributions of this work are twofold: *identifying the "compute-dumping" issue* in an existing baseline algorithm and *introducing the Verifier-Threshold algorithm* as a solution. We validate our approach on state-of-the-art image generation models and datasets. We organize the rest of the paper as follows: Sec. 2 discusses the baseline algorithm and experimental setup. Sec. 3 analyzes the compute dumping phenomenon and presents solutions, including *Verifier-Threshold*. Lastly, Sec. 4 presents results, including example images generated by the different approaches.

2 PRELIMINARIES

2.1 RELATED WORKS

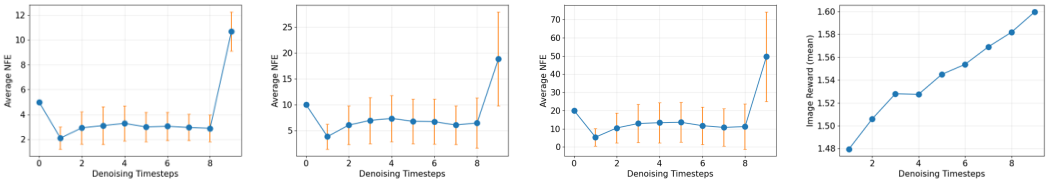
Diffusion models Dhariwal & Nichol (2021); Ho et al. (2020) and flow models Labs et al. (2025) along with autoregressive models Tian et al. (2024) have become the mainstream methods for image generation. Test-time scaling for image generation basically aims to use the compute available at test-time to improve performance, either through increasing the denoising steps Salimans & Ho (2022) or searching for better noises Ma et al. (2025); Kim et al. (2025). Noise-search in particular has been shown to scale much better than increasing denoising steps Ma et al. (2025); hence we focus on it here. FlowRBF introduces the key idea of *non-uniform inference-compute budget allocation across denoising steps*, which is a crucial advance in noise-search approaches. Using this idea, it outperforms prior baselines such as Best-of-N (which performs as well as zero-order and Search-over-Paths) from Ma et al. (2025). Hence, we use FlowRBF as our main baseline and show improvements on the same flow models as a proof of concept. FlowRBF's approach of non-uniform allocation and our proposed algorithm can also be extended to other diffusion models. Prior work is discussed in detail in Appendix B.

2.2 EXPERIMENTAL SETUP

We use the FLUX-Schnell model Labs et al. (2025) for all experiments, following Kim et al. (2025). We use ImageReward Xu et al. (2023) and VQAScore Lin et al. (2024) as verifier reward models to guide the noise-search process. Only open-source models are used throughout since these experiments cannot be conducted with closed-source models. To evaluate performance, we use the GenEval benchmark Ghosh et al. (2023), which consists of prompts that test for attributes such as color, number, and position and provides objective answers. We report the average percentage of correct images across attributes as our performance metric. For efficiency, we use the average wall-clock duration per prompt on a single AMD MI300X GPU (reported relative to SDv2.1 which takes the least amount of time; see Appendix C). We also report the number of function evaluations (NFEs), where lower is better. More details are provided in Appendix A.

3 VERIFIER THRESHOLD

3.1 COMPUTE DUMPING AND MANUAL BUDGETING



(a) (b) (c) (d)

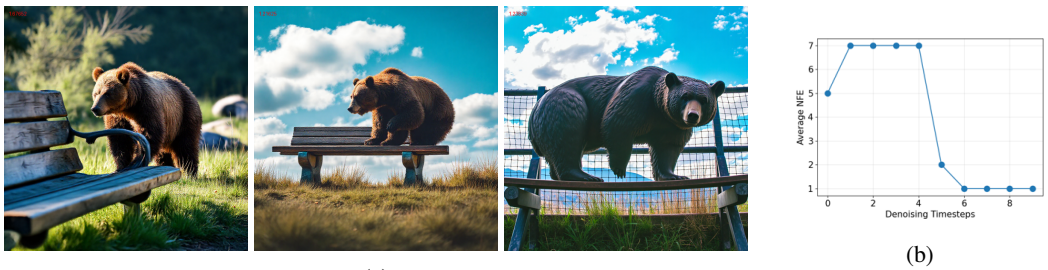
Figure 2: (a), (b) and (c) show how FlowRBF distributes NFEs (or compute budget) across denoising steps, averaged across all GenEval prompts with the total NFE budget set to 40, 80, and 160, respectively. Notice that in all cases, the inference-compute gets "dumped" at the last denoising step, and this issue persists even at total NFE budget as large as 500. (d) shows the verifier score as a function of denoising steps averaged over all GenEval prompts for total NFEs = 80. Notice that the verifier score does not increase at the last step in proportion to the amount of compute dedicated to it, thus wasting the test-time compute budget.

The efficacy of FlowRBF can be mainly attributed to its non-uniform inference-compute allocation across denoising steps, termed *rollover budget forcing (RBF)*. The basic concept behind RBF is that

the denoiser remains at a denoising step until either: (a) the verifier score for a new noise particle exceeds the previous step's best score, or (b) the per-step maximum compute budget is exhausted. The issue with these criteria is that (a) the method is greedy and advances even for marginal improvements, and (b) it does not account for "wrong" trajectories induced by per-step budget constraints.

Manifestations of these issues are shown in Figs. 2 and 3, where the model runs a ten-step denoising process. Fig. 2(a–c) show that although RBF is designed to enable non-uniform allocation of inference compute (NFEs), most steps (1–8) receive similar NFEs (except for step 0, which is hard-coded to use a predefined NFE). As a result, a large fraction of the test-time NFEs are "dumped" into the final step. When the NFE budget is lower ($= 40$; Fig. 2(a)), the algorithm quickly exhausts the per-step budget and advances to the next step. When the budget is higher ($= 80, 160$; Fig. 2(b,c)), the greediness of RBF causes it to advance quickly to the next step as well. In both cases, substantial budget remains and is forced into the last denoising step. Fig. 2(d) shows that the average verifier score at the last denoising step does not increase proportionally with the allocated compute, underscoring the dumping issue. Fig. 3(a) shows an example prompt that highlights the resulting failure mode.

Intuitively, the RBF budget allocations in Fig. 2 are sub-optimal because the structural changes needed to obtain a semantically correct image often occur in early denoising steps, suggesting that more budget should be allocated early rather than at the end. Manually hard-coding the compute distribution to allocate more budget to initial steps is one way to address this issue (Fig. 3(b)). Although this can improve some prompts, it does not generalize well. Therefore, an algorithm that automatically redistributes compute is needed.



(a) (b)

Figure 3: (a) shows three images generated at different denoising steps for the prompt "a photo of a bench left of a bear" using FlowRBF, with a total of 40 NFEs and ImageReward Xu et al. (2023) as the verifier. This example illustrates the compute-dumping issue. The first image (at the second denoising step) captures the correct structure, though the texture remains noisy. However, the second image, produced at the third step, achieves the highest verifier score and is therefore selected, despite placing the bear on top of the bench rather than to the left. Because no additional budget is allocated at earlier steps, the denoising process continues from this incorrect latent, ultimately producing a final image with the bear on top of the bench. (b) demonstrates how this issue could be mitigated by manually reallocating the compute budget through hard-coded NFEs at each step (i.e., manual budgeting).

3.2 Verifier-Threshold Algorithm

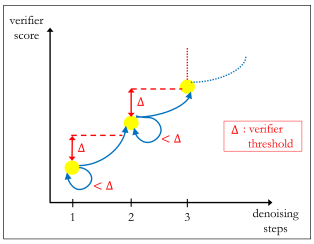
Rather than advancing to the next denoising step as soon as a higher-reward noise particle is obtained, we propose waiting until the verifier score surpasses the previous one by a significant margin (i.e., a threshold). In this way, the algorithm progresses only when the score at a given step improves sufficiently. This mechanism forms the basis of our *Verifier-Threshold* algorithm, as illustrated in Fig. 4(a) and detailed in Alg. 1.

4 Results

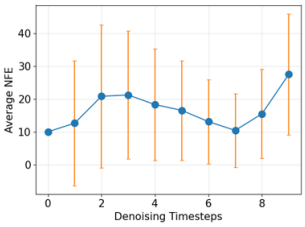
As described in Sec. 2, we evaluate different algorithms on the GenEval benchmark Ghosh et al. (2023). Fig. 1 highlights the advantages of our approach when using VQAScore Lin et al. (2024) as the verifier. The choice of threshold value Δ is described in Appendix A. To show that these benefits generalize across verifiers, we also evaluate ImageReward Xu et al. (2023), as shown in Fig. 5. In both cases, our method achieves $2\times$ to $4\times$ higher efficiency for the same benchmark score, or alternatively a 4% to 5% improvement in benchmark score at the same computational cost. All reported results correspond to practical test-time compute budgets (2–3 minutes). Additionally, Fig. 6

Algorithm 1 Verifier-Threshold Algorithm

Input: compute budget N, denoising timesteps T, generator model $\mathcal{G}_\theta$, verifier model $\mathcal{R}$, prompt p, Verifier-Threshold Δ
Output: image
$t = 1$
for $n = 1, ...N$ **do**
  # sample a noise particle and perform 1 NFE
  $\epsilon_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $z_t = \mathcal{G}_\theta(z_{t-1}, \epsilon_n, t, p)$
  # calculate reward score using verifier
  $r_t = \mathcal{R}(\text{Tweedie's}(z_t), p)$
  # Verifier-Threshold criteria
  if $r_t - r_{t-1} > \Delta$: **then**
    $t++$
  end if
  if $t > T$: **then**
    break
  end if
end for
return Tweedie's (z_t)



(a)



(b)

Figure 4: (a) verifier threshold mechanism and (b) its compute allocation.

presents example prompts where our algorithm produces objectively correct outputs compared to baseline methods. More comprehensive results and further examples are provided in Appendix A.

(a)

(b)

(c)

Figure 6: The outputs of the model for the prompt "a photo of a wine glass right of a hot dog". (a), (b), (c) represent the images generated by FLUX-Schnell Labs et al. (2025) with no test-time scaling, FlowRBF, and Verifier-Threshold, respectively. Figures (a) & (b) have the wine glass on the left while (c) accurately provides what the prompt is requesting.

5 Conclusion

In this work, we identify a key flaw (compute dumping) in the state-of-the-art test-time scaling algorithm (Kim et al. (2025)) and propose the Verifier-Threshold algorithm as an effective solution. We demonstrate its performance and efficiency advantages across multiple verifiers on the GenEval benchmark. For future work, we plan to investigate adaptive verifier thresholds that vary across denoising steps and explore strategies to automate their selection. There is also a lack of theoretical or interpretability grounding for test-time compute in the image generation community currently, and we will look into that as well as a future direction for our work.

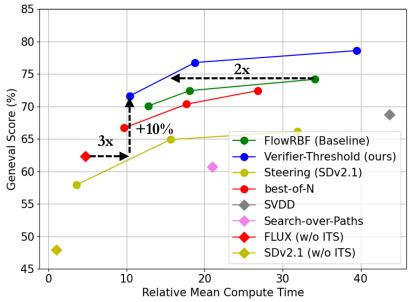


Figure 5: Test-time scaling when ImageReward Xu et al. (2023) is used as verifier.

REFERENCES

Apple Machine Learning Research. Univg: A generalist diffusion model for unified image generation and editing. https://machinelearning.apple.com/research/univg-diffusion-model, 2025. Apple ML Research page describing the UniVG image generation/editing model.

Google DeepMind. Nano banana: Gemini 2.5 flash image. urlhttps://blog.google/products/gemini/updated-image-editing-model/, 2025. Image generation / editing model, announcement August 26, 2025.

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.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747, 2023.

Yunho Jin, Gu-Yeon Wei, and David Brooks. The energy cost of reasoning: Analyzing energy usage in llms with test-time compute. *arXiv preprint arXiv:2505.14733*, 2025.

Jaihoon Kim, Taehoon Yoon, Jisung Hwang, and Minhyuk Sung. Inference-time scaling for flow models via stochastic generation and rollover budget forcing. *arXiv preprint arXiv:2503.19385*, 2025.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL https://arxiv.org/abs/2506.15742.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024a.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024b.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pp. 366–384. Springer, 2024.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

Midjourney. Version 7. https://docs.midjourney.com/hc/en-us/articles/32199405667853-Version, April 2025. V7 released Apr 3, 2025; default since Jun 17, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 2025-09-29.

OpenAI. Introducing our latest image generation model in the api. https://openai.com/index/image-generation-api/, April 2025. API access to gpt-image-1 (same image system as 4o in ChatGPT).

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Stability AI. Introducing stable diffusion 3.5. https://stability.ai/news/introducing-stable-diffusion-3-5, October 2024. Launch post for SD 3.5 (Large / Large Turbo / Medium).

Stelia. How reasoning ai models are transforming edge infrastructure, March 2025. URL https://newsroom.stelia.ai/how-reasoning-ai-models-are-transforming-edge-infrastructure/. Accessed: YYYY-MM-DD.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Le Zhuo, Liangbing Zhao, Sayak Paul, Yue Liao, Renrui Zhang, Yi Xin, Peng Gao, Mohamed Elhoseiny, and Hongsheng Li. From reflection to perfection: Scaling inference-time optimization for text-to-image diffusion models via reflection tuning. *arXiv preprint arXiv:2504.16080*, 2025.

A Experimental Setup

A.1 Dataset & Models

For evaluating our models we employ the widely-used GenEval dataset Ghosh et al. (2023). It contains 553 prompts which checks six different attributes: single object, two object, counting, colors, position, and attribute binding. For the generator, we use the FLUX-Schnell model Labs et al. (2025) extensively due to its SOTA performance and baseline implementation from FlowRBF. We run it consistently for ten denoising steps for best results. For the verifiers, we use ImageReward Xu et al. (2023) and VQAScore Lin et al. (2024) which are both variants of the CLIP model Radford et al. (2021).

A.2 Hyperparameters & Verifier-Threshold Choice

We use the same setup and hyperparameters as FlowRBF for all the methods: Best-of-N, search-over-paths, SVDD, FlowRBF and ours. When using 40 NFEs, we used 5 initial noise samples, and doubled/quadrupled it for 80/160 respectively. We used guidance-scale 3.5, diffusion norm 3.0, exponential time-scheduler with $t_{\text{max}} = 1000$ and image-size 1024×1024.

The threshold value Δ is chosen independently for each verifier. The difference between the average verifier score for the initial timestep and final timestep is divided by the number of timesteps to obtain it. We use a verifier threshold value (Δ) of 0.005 and 0.00125 for ImageReward and VQAScore respectively when the total NFEs is 40. For subsequent total NFEs of 80 and 160, we correspondingly multiplied the threshold values with $2\times, 4\times$ the above value. Fig. 7 shows the verifier score variation for ImageReward which was obtained when running a simple evaluation of the FlowRBF algorithm (NFE$= 80$) for GenEval prompts. From the explanation above, we can calculate the value as $\Delta = \frac{1.62-1.52}{10} = 0.01 = 2 \times 0.005$.

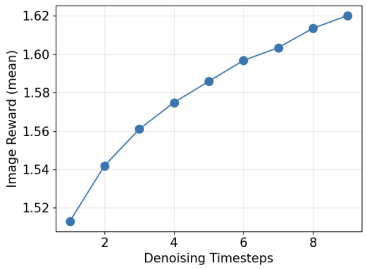


Figure 7: Verifier score variation for ImageReward used to derive Δ.

B Related Works

Test-time compute as a new scaling axis

In recent years, scaling of AI models has mainly involved increasing model size and the amount of compute and data utilized during training. However, recent work shows that spending more compute at test-time can substantially improve output quality and reasoning capability, thus creating a new dimension for scaling compute to improve model performance. OpenAI's o1 series popularized this paradigm by spending more test-time budget on deliberate "thinking" before answering OpenAI (2024). DeepSeek-R1/R1-Zero similarly leverages extended reasoning trajectories at inference Guo et al. (2025). Snell et al. formalize when scaling test-time compute can be more effective than scaling model size Snell et al. (2024). Complementing these analyses, *s1* demonstrates a simple and effective test-time scaling recipe with strong empirical gains Muennighoff et al. (2025).

Test-time scaling for image generation

While diffusion models and flow models have achieved impressive results in image and video generation, naive generation with these models often fails to satisfy complex user instructions involving object quantities, relative position, and size, among other conceptual attributes. Recently, test-time scaling methods have been developed for diffusion and flow models that significantly outperform naive generation.

Reward-based sampling and noise-search

Reward-based sampling methods repeatedly sample from the model's learned distribution, guided by a reward such as text–image alignment or aesthetic quality, to select higher-quality outputs. The most basic example is Best-of-N (BoN), which generates N independent images with different random seeds and picks the one with the highest reward. Ma et al. (2025) formalize this paradigm by introducing both **BoN** and Search-over-Paths (**SoP**), the latter refining trajectories by sampling particles along the denoising path Ma et al. (2025). Building on this idea, "noise trajectory search" methods incorporate reward-guided selection directly into the denoising process: **SVDD** Li et al.

(2024b) selects, at every denoising step, the particle with the highest expected reward. Extending reward-based search to flow models, Kim et al. (2025) Kim et al. (2025) propose three complementary techniques: test-time SDE conversion, which introduces stochasticity into otherwise deterministic flows; interpolant conversion, which alters the trajectory interpolant to expand the search space; and Rollover Budget Forcing (**RBF**), which dynamically reallocates compute across timesteps by advancing any particle that exceeds the previous best reward and rolling over unused function evaluations to future steps, thereby ensuring efficient utilization of the total budget.

Method	**Generator**	**Verifier**	**NFEs**	**time** ($\times$)	**GenEval (%)**
Regular-T2I	SDv2.1	-	1	1	47.93
Regular-T2I	FLUX-Schnell	-	10	4.74	62.33
Steering	SDv2.1	ImageReward	4	3.57	57.93
Steering	SDv2.1	ImageReward	20	15.62	64.91
Steering	SDv2.1	ImageReward	40	31.91	66.18
BoN	FLUX-Schnell	ImageReward	40	9.70	70.62
BoN	FLUX-Schnell	ImageReward	80	17.16	70.36
BoN	FLUX-Schnell	ImageReward	160	26.39	72.45
BoN	FLUX-Schnell	VQAScore	40	10.29	70.07
BoN	FLUX-Schnell	VQAScore	80	18.72	70.54
BoN	FLUX-Schnell	VQAScore	160	27.07	71.91
SoP	FLUX-Schnell	ImageReward	250	21.00	60.68
SoP	FLUX-Schnell	VQAScore	250	22.25	61.19
SVDD	FLUX-Schnell	ImageReward	250	43.69	68.76
SVDD	FLUX-Schnell	VQAScore	250	56.74	72.89
FlowRBF	FLUX-Schnell	ImageReward	40	12.80	70.10
FlowRBF	FLUX-Schnell	ImageReward	80	18.11	72.45
FlowRBF	FLUX-Schnell	ImageReward	160	34.11	74.20
FlowRBF	FLUX-Schnell	VQAScore	40	13.83	70.32
FlowRBF	FLUX-Schnell	VQAScore	80	27.03	71.77
FlowRBF	FLUX-Schnell	VQAScore	160	48.98	73.45
VT (Ours)	FLUX-Schnell	ImageReward	40	10.31	71.61
VT (Ours)	FLUX-Schnell	ImageReward	80	18.68	76.77
VT (Ours)	FLUX-Schnell	ImageReward	160	39.36	**78.62**
VT (Ours)	FLUX-Schnell	VQAScore	40	12.80	73.83
VT (Ours)	FLUX-Schnell	VQAScore	80	25.90	76.12
VT (Ours)	FLUX-Schnell	VQAScore	160	51.97	77.20

Table 1: Comparison of different approaches across models, verifiers, NFEs, along with their efficiency (wall-clock time) & performance (GenEval) scores. Methods are Steering Singhal et al. (2025), Best-of-N Ma et al. (2025) (BoN), Search-over-Paths Ma et al. (2025) (SoP), Soft Value-based Decoding in Diffusion Li et al. (2024a) (SVDD), FlowRBF Kim et al. (2025), and Verifier-Threshold (VT). Verifiers are ImageReward Xu et al. (2023) and VQAScore Lin et al. (2024).

C Comprehensive Results

In this section, we expand the plots in Fig. 1 and Fig. 5 into Tab. 1. We also show images for various example prompts, including cases where our approach succeeds while other baselines fail, as well as cases where all approaches fail. In all example triplets, Fig. (a) corresponds to generation without

test-time scaling, Fig. (b) corresponds to FlowRBF, and Fig. (c) corresponds to our *Verifier-Threshold* algorithm.

(a) (b) (c)

Figure 8: "a photo of a purple wine glass and a black apple". For all examples in this section, Fig. (a) is the image generated without test-time scaling, Fig. (b) is generated using FlowRBF and Fig. (c) is generated by our Verifier-Threshold algorithm.

(a) (b) (c)

Figure 9: "a photo of four clocks"

(a) (b) (c)

Figure 10: "a photo of a bench left of a bear"

(a) (b) (c)

Figure 11: “a photo of three sports balls”

(a) (b) (c)

Figure 12: “a photo of a train above a potted plant”

(a) (b) (c)

Figure 13: “a photo of a brown oven and a purple train”

(a) (b) (c)

Figure 14: “a photo of a purple carrot”

(a) (b) (c)

Figure 15: “a photo of a wine glass right of a hot dog”

(a) (b) (c)

Figure 16: “a photo of a purple computer keyboard and a red chair”

(a) (b) (c)

Figure 17: “a photo of a white banana and a black elephant”

(a) (b) (c)

Figure 18: “a photo of an orange handbag and a green carrot”

(a) (b) (c)

Figure 19: “a photo of an orange donut and a yellow stop sign”

(a) (b) (c)

Figure 20: “a photo of a sports ball left of an umbrella”