ARCFLOW: UNLEASHING 2-STEP TEXT-TO-IMAGE GENERATION VIA HIGH-PRECISION NON-LINEAR FLOW DISTILLATION

Anonymous authors
Paper under double-blind review

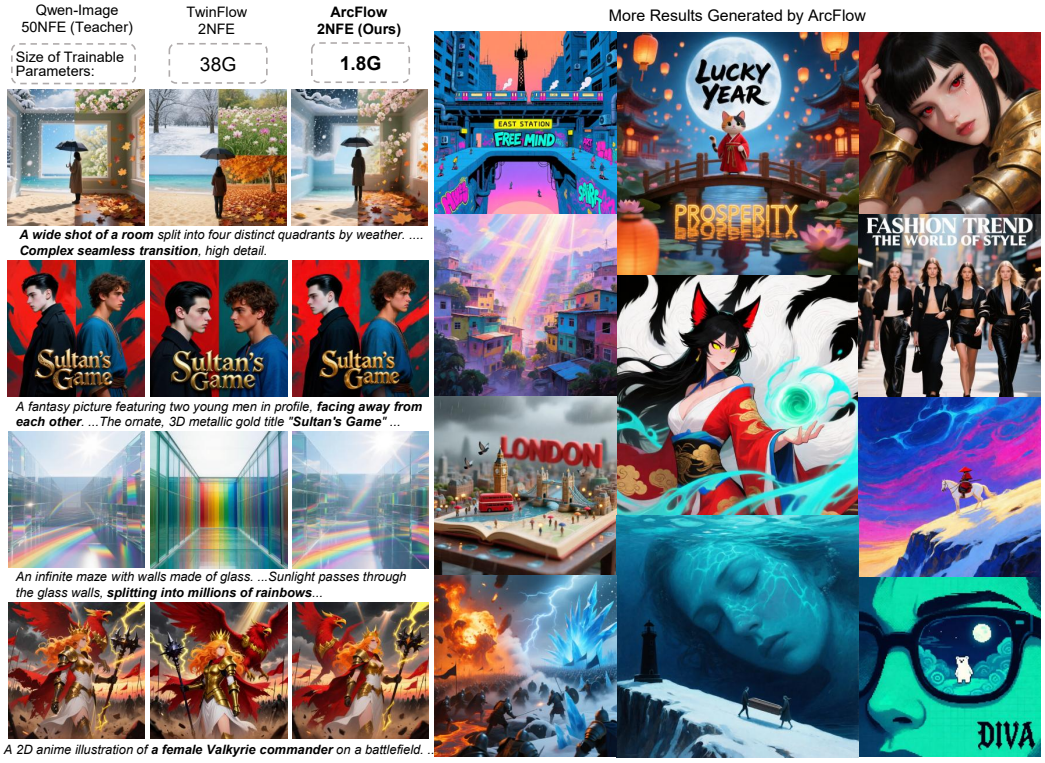

Figure 1: Comparisons between images generated by ArcFlow and other state-of-the-art distillation methods based on Qwen-Image-20B, demonstrating the power of ArcFlow for few-step high-fidelity generation while maintaining remarkable parameter efficiency.

ABSTRACT

Diffusion models have achieved remarkable generation quality, but they suffer from significant inference cost due to their reliance on multiple sequential denoising steps, motivating recent efforts to distill this inference process into a few-step regime. However, existing distillation methods typically approximate the teacher trajectory by using linear shortcuts, which makes it difficult to match its constantly changing tangent directions as velocities evolve across timesteps, thereby leading to quality degradation. To address this limitation, we propose ArcFlow, a few-step distillation framework that explicitly employs non-linear flow trajectories to approximate pre-trained teacher trajectories. Concretely, ArcFlow parameterizes the velocity field underlying the inference trajectory as a mixture of continuous momentum processes. This approximate velocity field enables ArcFlow to capture velocity evolution and extrapolate coherent velocities to form a continuous non-linear trajectory within each denoising step. Importantly, this parameterization admits an analytical integration of this non-linear trajectory, which mitigates linear

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

discretization errors and results in high-precision approximation of the teacher trajectory. To train this parameterization into a few-step generator, we implement ArcFlow via trajectory distillation on pre-trained teacher models using lightweight adapters. This strategy ensures fast, stable convergence while preserving generative diversity and quality. Built on large-scale models (Qwen-Image-20B and FLUX.1-dev), ArcFlow only fine-tunes on less than 5% of original parameters and achieves a $40\times$ speedup with 2 NFEs over the original multi-step teachers without significant quality degradation. Experiments on benchmarks show the effectiveness of ArcFlow both qualitatively and quantitatively.

1 INTRODUCTION

Diffusion and flow matching models have emerged as the dominant paradigms for high-fidelity visual generation (Ho et al., 2020; Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023). Despite their impressive capabilities, they rely on iterative differential equation solvers, typically requiring 40 to 100 denoising steps to traverse the trajectory from noise to data, making them impractical for real-time applications. Therefore, accelerating sampling without compromising quality remains a critical challenge.

To address this issue, recent research has explored various paradigms to distill a pre-trained teacher model into a few-step student generator. Different methods range from progressive distillation (Salimans & Ho, 2022; Meng et al., 2023) to consistency-based approaches (Song et al., 2023; Luo et al., 2023), and distribution matching (Sauer et al., 2023; Yin et al., 2024; Cheng et al., 2025) that employ adversarial or divergence losses to align distributions. However, their essence still lies in approximating the trajectory from the teacher generation process ($40 \sim 100$ steps), whose tangent directions vary over multiple timesteps, via a linear shortcut under very few steps ($2 \sim 4$ steps). This enforces the students to implicitly learn such tangent variation with linear trajectories, leading to geometric mismatch.

In light of this, we propose ArcFlow, a few-step distillation framework that introduces explicit non-linear flow trajectories via velocity parameterization to approximate the flow trajectories from a pre-trained teacher model. Since the flow trajectory is equivalent to how the velocity evolves across timesteps, we utilize the notion of momentum (Goldstein et al., 2002) to approximate this evolution, where the overall trajectory is determined only by the initial velocity and a momentum factor. Consequently, we parameterize the velocity field as a weighted mixture of continuous momentum processes. By harnessing the continuity of this momentum process over adjacent timesteps, the model can extrapolate coherent velocities through the parameterization, thus efficiently constructing the non-linear trajectories based on the predicted velocity shifts. Notably, our parameterization admits a closed-form analytical solution to our predicted velocity field which approximates the flow ODE (Song et al., 2021), enabling direct computation of the terminal state in a single forward pass. This ensures the predicted velocity evolution is applied analytically across timesteps within the interval, rather than being approximated by a linear discrete update, thereby ensuring high-precision flow distillation.

By tackling the geometric mismatch, ArcFlow ensures that the student trajectory better aligns with the teacher's inherent tangent variation. This alignment fundamentally simplifies the distillation task, enabling parameter-efficient training. Unlike prior methods requiring full-model training, ArcFlow achieves state-of-the-art results by fine-tuning only lightweight LoRA adapters and the output head.

As illustrated in Figure 1, with only 2 NFEs, ArcFlow achieves high-fidelity generation comparable to the teacher Qwen-Image-20B, surpassing the 2-step generation quality of TwinFlow (Cheng et al., 2025) while utilizing very few trainable parameters. Furthermore, the convergence analysis (Figure 5) highlights that the training of ArcFlow yields significantly faster convergence and superior stability, validating the effectiveness of the alignment with non-linear trajectory that efficiently eliminates the geometric optimization bottleneck.

Our main contributions are as follows: (1) We propose ArcFlow, the first distillation framework to explicitly construct a closed-form non-linear flow trajectory to approximate the teacher trajectory. We parameterize the velocity as a continuous momentum mixture, which admits analytic trajectory integration and achieves high-precision alignment with the teacher. (2) We introduce an analytic

trajectory solver for ArcFlow, which enables an efficient objective for distillation. It simplifies the training process, enabling parameter-efficient adaptation and fast convergence. (3) Evaluations on benchmark datasets demonstrate superior robustness of ArcFlow, achieving SOTA across diverse backbones. It achieves a $40\times$ inference speedup over the teacher and at most $4\times$ faster training convergence than prior methods, while fine-tuning only less than 5% of the original parameters.

2 Related Work

Text-to-image generation Diffusion (Ho et al., 2020) and flow matching models (Lipman et al., 2023) have emerged as the standard for high-resolution visual synthesis. Recent scaling efforts, such as Stable Diffusion 3 (Esser et al., 2024), FLUX (Labs, 2024; 2025), and Qwen-Image (Wu et al., 2025), leverage Transformers to achieve exceptional performance. However, their numerical solvers require 40 to 100 function evaluations (NFEs), hindering real-time deployment.

Few-step Image Generation Accelerating the inference of diffusion models has become a critical topic, aiming to achieve high-fidelity synthesis with few NFEs. To this end, knowledge distillation trains a student to approximate the complex sampling trajectory of a pre-trained teacher. One line of work, including Progressive Distillation (Salimans & Ho, 2022; Meng et al., 2023) and Rectified Flow (Liu et al., 2022), simplifies trajectories by iteratively straightening the flow to reduce NFEs. However, they struggle to eliminate discretization errors in the few-step regime. Consistency Models (Song et al., 2023; Luo et al., 2023) map points directly to the data via self-consistency constraints, while related approaches can incur costly Jacobian-vector products (Geng et al., 2025).

To further push limits to 1–4 steps, VSD (Wang et al., 2023) and DMD (Yin et al., 2024) introduce distribution-matching losses, and TwinFlow (Cheng et al., 2025) uses a self-adversarial objective. While these improve visual sharpness, distribution-level objectives can compromise diversity and increase training overhead. Recent attempts such as pi-Flow (Chen et al., 2025a) use Gaussian-mixture velocity policies (Chen et al., 2025b) to approximate teacher velocities. However, supervision acts on state-dependent local velocity predictions, leaving the full interval trajectory to emerge indirectly through successive updates. ArcFlow instead parameterizes continuous momentum dynamics, so teacher observations across timesteps directly optimize shared modes governing the entire interval. This yields an analytically tractable distillation system that unifies trajectory learning and closed-form transitions, facilitating more efficient optimization and faster convergence (Appendix H).

3 Method

Pre-trained diffusion models follow PF-ODE integration trajectories with constantly varying tangents (Song et al., 2021), whereas existing distillation methods (Song et al., 2023; Yin et al., 2024; Cheng et al., 2025) approximate them using linear shortcuts, resulting in geometric mismatch.

In light of this, we propose ArcFlow, a text-to-image distillation framework that utilizes the notion of momentum process to construct non-linear flow trajectories across long timestep intervals. In this section, we first formalize the momentum-based parameterization (Section 3.1), derive the analytical trajectory integration solver (Section 3.2), and detail the trajectory distillation strategy to train ArcFlow (Section 3.3). The condition variable (text prompt) is omitted from subsequent descriptions for brevity. Since ArcFlow mainly relies on learning from a pre-trained teacher, we define the velocity field from the frozen teacher as ground truth for the subsequent analysis.

3.1 Momentum Parameterization of Probability Flow

The Probability Flow ODE framework (Song et al., 2021) reveals that the diffusion process follows a continuous trajectory, where the denoising velocities are strongly correlated between adjacent timesteps. However, standard numerical solvers (e.g., Euler method (Butcher, 2016)) appoximate the integration process by taking discrete steps independently without considering their associations across timesteps. Thus, we argue that the standard multi-step sampling, which re-evaluates the network repeatedly to traverse this smooth trajectory, suffers from severe redundancy.

To explicitly exploit this inherent continuous evolution of the velocity field across timesteps, we introduce the notion of momentum (Goldstein et al., 2002), to model and parameterize such properties.

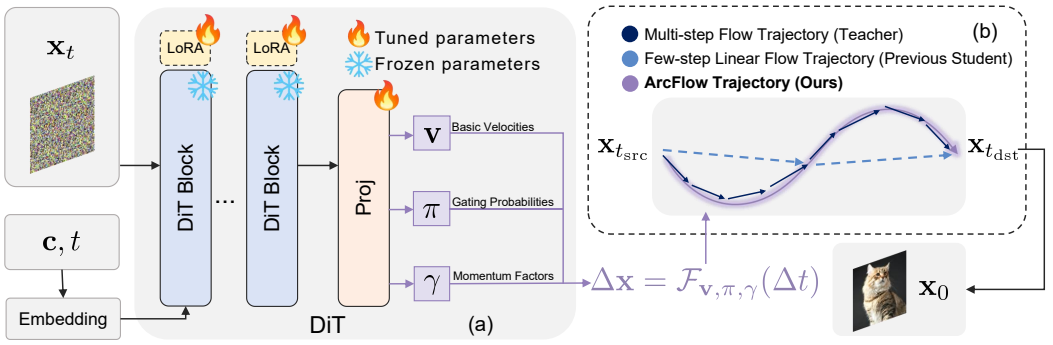

Figure 2: ArcFlow Framework. (a) The forward pipeline of ArcFlow. Given an input x_t, the condition c and timestep t, a DiT backbone with three projection heads predicts the parameters v, ω, γ across K dynamic modes, which respectively denote the mode-specific velocities, momentum factors, and the gating probabilities used to reconstruct the teacher velocity field. (b) A comparison of flow trajectories produced by the multi-step teacher model, the few-step linear student model and ArcFlow.

Specifically, let $\mathbf{v}(\mathbf{x}_t, t)$ denote the velocity field at timestep $t \in [0, 1]$. We assume that velocities at adjacent timesteps follow a momentum transmission ansatz parameterized by a factor γ, i.e., the velocity evolution is modeled as $\mathbf{v}(\mathbf{x}_t, t) = \mathbf{v}(\mathbf{x}_{t+\Delta t}, t + \Delta t) \cdot \gamma^{\Delta t}$ from $t + \Delta t$ to t.

Recursively apply the above formula from a starting timestep t_s to any ending timestep $t \in [0, t_s)$ along the PF-ODE trajectory, the velocity evolution yields:

$$\mathbf{v}(\mathbf{x}_t, t) = \mathbf{v}(\mathbf{x}_{t_s}, t_s) \cdot \gamma^{t_s - t}, \tag{1}$$

where $\gamma \in \mathbb{R}^+$. Based on the assumption of Eq. 1, given the initial velocity $\mathbf{v}(\mathbf{x}_{t_s}, t_s)$, $\mathbf{v}(\mathbf{x}_t, t)$ at any timestep $t \in [0, t_s)$ can be extrapolated directly. Therefore, our momentum parameterization allows the approximated velocity field to analytically predict the velocity at every timestep $t \in [0, t_s)$ after only a single NFE, reaching $\mathbf{x}_t$ directly.

While momentum helps approximate velocity evolution, a single momentum factor γ is insufficient to capture the hierarchical frequency dynamics in image generation. Empirical studies (Choi et al., 2022) show that different frequency components evolve at distinct rates during denoising, implying that the corresponding velocity field $\mathbf{v}(\mathbf{x}_t, t)$ inherently consists of multiple evolution modes with different decay rates. To model such dynamics, as shown in Figure 2(a), we formulate $\mathbf{v}(\mathbf{x}_t, t)$ as a probabilistic mixture of K distinct momentum modes, each driven by its own momentum factor. Specifically, we introduce a discrete latent variable $z \in \{1, \ldots, K\}$ to index the K different momentum modes, and then derive the overall velocity field $\mathbf{v}_\theta(\mathbf{x}_t, t)$, optimized by parameter θ:

$$\mathbf{v}_\theta(\mathbf{x}_t, t) = \mathbb{E}_{z \sim p_\theta(z|\mathbf{x}_t)} \left[\mathbf{v}(\mathbf{x}_t, t \mid z)\right] = \sum_{k=1}^{K} \underbrace{p_\theta(z = k|\mathbf{x}_t)}_{\pi_k(\mathbf{x}_t)} \cdot \underbrace{\mathbf{v}_k(\mathbf{x}_t) \cdot \gamma_k(\mathbf{x}_t)^{1-t}}_{\text{Mode-specific Dynamics}}, \tag{2}$$

where $\pi_k(\mathbf{x}_t) \in [0, 1]$ refers to the gating probability predicted by the parameter θ, subject to $\sum \pi_k = 1$. $\mathbf{v}_k(\mathbf{x}_t) \in \mathbb{R}^D$ and $\gamma_k(\mathbf{x}_t) \in \mathbb{R}^+$ are the predicted basic velocity and momentum factor for the k-th mode. Consequently, ArcFlow divides the trajectory into several mode-specific sub-trajectories, enabling each to be learned in a more targeted manner, thereby improving overall learning efficiency and quality.

To further demonstrate the expressiveness of our parameterization, we introduce a theorem showing that Eq. 2 with K dynamic modes theoretically admits a parameter setting that perfectly fits the sampled trajectory at $N \leq K$ distinct timesteps.

Theorem 3.1. *Consider the velocity field predicted by ArcFlow at any sampled latent* $\mathbf{y}$ *and timestep* t*, parameterized as* $\mathbf{v}_\theta(\mathbf{y}, t) = \sum_{k=1}^{K} \pi_k(\mathbf{y})\, \mathbf{v}_k(\mathbf{y})\, \gamma_k(\mathbf{y})^{1-t}$ *according to Eq. 2. Let* $\mathbf{u}^*(\mathbf{y}, t)$ *denote the ground-truth velocity field, observed at* N *distinct timesteps* $\mathcal{T} = \{t_1, \ldots, t_N\} \subset (0, 1]$*. If the number of modes satisfies* $K \geq N$*, then there exists a parameter configuration* $\theta = \{\pi_k, \mathbf{v}_k, \gamma_k\}_{k=1}^{K}$*:*

$$\mathbf{v}_\theta(\mathbf{y}, t_n) = \mathbf{u}^*(\mathbf{y}, t_n), \quad \forall t_n \in \mathcal{T}, \tag{3}$$

We prove the theorem in Section D.2. The theoretical result confirms that the momentum-based parameterization has sufficient capacity to interpolate the ground-truth velocity field at discrete timesteps. Since the teacher trajectories mostly exhibit smooth velocity evolution, using $K \ll N$ (e.g. $K = 8/16$) suffices for high-precision distillation, as validated in Table 12.

3.2 Analytic ODE Solvers

As described above, ArcFlow parameterizes the velocity field as a mixture of momentum modes (Eq. 2), which is mathematically equivalent to a linear combination of exponential time factors. This structure admits closed-form integration over timestep intervals, allowing direct latent updates.

Concretely, at an starting timestep t_s, our model predicts the parameters θ to define an approximate velocity field over the subsequent interval. For any sampling step from t_s to $t_e \in [0, t_s)$, we define the Analytic Transition Operator Φ as the latent displacement $\Delta \mathbf{x}_{t_s \to t_e}$ induced by this velocity field. By analytically integrating this velocity based on Eq. 2 across timesteps, $\Phi(\mathbf{x}_{t_s}, t_s, t_e; \theta)$ admits the following closed-form expression (the full derivation is provided in Section D.1):

$$\Phi(\mathbf{x}_{t_s}, t_s, t_e; \theta) \triangleq \Delta \mathbf{x}_{t_s \to t_e} = \sum_{k=1}^{K} \pi_k(\mathbf{x}_{t_s})\, \mathbf{v}_k(\mathbf{x}_{t_s})\, \mathcal{C}(\gamma_k(\mathbf{x}_{t_s}),\, t_s,\, t_e)\,, \tag{4}$$

where the Momentum Integral Coefficient $\mathcal{C}(\cdot)$ is defined as

$$\mathcal{C}(\gamma, t_s, t_e) = \begin{cases} \dfrac{\gamma^{1-t_e} - \gamma^{1-t_s}}{\ln \gamma}, & \gamma \neq 1, \\ t_s - t_e, & \gamma = 1, \end{cases} \tag{5}$$

Crucially, the coefficient $\mathcal{C}(\gamma, t_s, t_e)$ smoothly reduces to the linear form $t_s - t_e$ as $\gamma \to 1$ (see Section D.1 for full derivation). This ensures numerical stability of our solver at the singularity, showing that our parameterization seamlessly bridges non-linear dynamics ($\gamma \neq 1$) and the linear flow regime ($\gamma = 1$).

The parameters θ of the velocity field are evaluated at $\mathbf{x}_{t_s}$ and held fixed during integration. Under this velocity field, whose K momentum modes provide sufficient expressiveness to approximate the teacher (Theorem 3.1), the next latent for any step from t_s to t_e is given explicitly by $\mathbf{x}_{t_e} = \mathbf{x}_{t_s} - \Phi(\mathbf{x}_{t_s}, t_s, t_e; \theta)$, allowing analytical integration to target states. This enables our model to perform inference in a closed-form step (see Algorithm 2).

As shown in Figure 2(b), while previous few-step students rely on very few straight sub-lines to fit a multi-step teacher trajectory whose tangent directions rapidly changing, leading to a poor approximation, ArcFlow trajectory from our analytic solver can naturally inherit the non-linearity and better align with the teacher's overall trajectory.

3.3 Flow Distillation with Analytic Solvers

Since ArcFlow can better align with the teacher trajectory, we adopt a practical flow distillation strategy based on a pre-trained teacher model. As directly synthesizing the trajectory is infeasible within the flow matching framework which only predicts the velocity field, we reconstruct the teacher's trajectory by aligning its tangent direction (instantaneous velocity) at sampled timesteps. For ArcFlow, this tangent is analytically derived via Eq. 2, so the trajectory alignment reduces to a velocity-matching objective: minimizing the discrepancy between student and teacher instantaneous velocities at the sampled $(\mathbf{x}_t, t)$ pairs.

Mixed Latent Integration. For each interval $[t_{\text{dst}}, t_{\text{src}}]$ with $t_{\text{src}} > t_{\text{dst}}$, ArcFlow predicts K momentum modes $\Theta = \{\pi_k, \mathbf{v}_k, \gamma_k\}_{k=1}^{K}$ in a single backbone evaluation at the starting state $(\mathbf{x}_{t_{\text{src}}}, t_{\text{src}})$, where π_k, $\mathbf{v}_k$, and γ_k denote the gating probability, basic velocity, and momentum factor of mode k, respectively. We sample n intermediate timesteps $\{t_i\}_{i=1}^{n}$ in decreasing order, with $t_0 = t_{\text{src}}$. Starting from $\mathbf{x}_{t_0}$, we integrate the velocity field over each sub-interval $[t_{i+1}, t_i]$ between adjacent timesteps to update $\mathbf{x}_{t_i}$ to $\mathbf{x}_{t_{i+1}}$. This integration proceeds in two stages: the teacher first advances the latent over a fraction $1 - \lambda$ of the sub-interval with its current velocity held constant; the student then completes the remaining fraction λ, where $\lambda \in [0, 1]$. With Θ held fixed, the student portion uses the analytic transition in Eq. 4, without numerical substeps or additional backbone evaluations.

We increase λ from 0 to 1 during warm-up, stabilizing early training with teacher guidance before switching to the student's own rollout states, as at inference. These states are detached before velocity matching; the mixing rule and full procedure are detailed in Section B.2 and Algorithm 1.

Instantaneous Velocity Matching. At any sampled timestep t_i within the interval, ArcFlow directly evaluates the student velocity $\mathbf{v}(\mathbf{x}_{t_i}, t_i; \Theta)$ from the shared Θ via Eq. 2, without another student backbone evaluation. The frozen teacher provides the target velocity $\mathbf{u}(\mathbf{x}_{t_i}, t_i)$ at the corresponding rollout state $\mathbf{x}_{t_i}$. The optimization objective is

$$\mathcal{L}_{\text{distill}} = \mathbb{E}_{t_i, \mathbf{x}_{t_i}} \left[\|\mathbf{v}(\mathbf{x}_{t_i}, t_i; \Theta) - \mathbf{u}(\mathbf{x}_{t_i}, t_i)\|^2 \right], \tag{6}$$

where the expectation is over the sampled timesteps t_i and their detached rollout states $\mathbf{x}_{t_i}$.

Enforcing this loss encourages the student's overall continuous trajectory to adhere to the teacher's complex trajectory. ArcFlow further simplifies this distillation process due to the momentum parameterization. Specifically, a single prediction of the momentum parameters Θ analytically defines the student's continuous trajectory over the interval, allowing its velocities to be evaluated directly without successive discrete latent-state updates. Velocity-matching gradients therefore propagate through the analytic velocity expression to these shared parameters, jointly refining the entire interval trajectory. As the parameterization captures non-linear velocity evolution, matching instantaneous velocities at only a few intermediate timesteps ($n = 4$) is sufficient in our experiments for close teacher alignment. Empirically, we find that training only LoRA adapters in selected modules and the output projection heads is sufficient for fast convergence, supporting efficient distillation with limited trainable parameters (Section 4.3).

4 Experiments

4.1 Implementation Details

We apply our distillation framework to two text-to-image models: Qwen-Image-20B (Wu et al., 2025) and FLUX.1-dev (Labs, 2024). As a parameter-efficient strategy, we freeze the vast majority of the backbone and train only 256-rank LoRA adapters injected into the feed-forward layers along with the final output projection head to accommodate the momentum parameter predictions. We train ArcFlow on a large-scale prompt dataset (2.3 million samples) introduced by pi-Flow (Chen et al., 2025a). We provide more training details in Section C.

We evaluate 1024×1024 generation on GenEval (Ghosh et al., 2023), DPG-Bench (Hu et al., 2024), ImageReward (Xu et al., 2023), HPSv2 (Wu et al., 2023), and OneIG-Bench (Chang et al., 2025). Moreover, we use a text prompt set, Align5000, to evaluate FID/pFID against 50-step teacher outputs. And we randomly sample 10,000 captions from the COCO 2014 validation set (Lin et al., 2015) as COCO-10K and evaluate FID/pFID against these real images. We also conduct additional human preference evaluation in Section G.1 and comparisons with methods including rCM and MeanFlow in Section G.2. Detailed dataset composition of Align5000, metric definitions, and evaluation settings are provided in Section C.2.

4.2 Comparison Study

Quantitative Results. We compare with recent few-step generative models distilled from the same teacher. For FLUX.1-dev (Labs, 2024), we compare against: SenseFlow (Ge et al., 2025), which uses DMD; pi-Flow (GM-FLUX) (Chen et al., 2025a), which approximates the linear step with policy. For Qwen-Image-20B (Wu et al., 2025), we compare with: Qwen-Image-Lightning (ModelTC) based on VSD; TwinFlow (Cheng et al., 2025) based on self-adversarial loss; pi-Flow (GM-Qwen). For SenseFlow, we reuse its official checkpoint, following the 2-step procedure reported in its paper. Since pi-Flow's official checkpoints only support 4-NFE inference, we include additional 4-NFE comparisons to complement our evaluation and ensure fairness. We also compare both methods trained from scratch at 2 NFE under a matched 3,000-step budget, with details in Appendix E.3.

We observe that ArcFlow consistently outperforms or remains competitive with state-of-the-art few-step models across the five benchmarks in Table 1, demonstrating robust alignment with complex instructions. Specifically, while adversarial-based methods (e.g., Qwen-Image-Lightning) suffer from mode collapse (losing diversity to improve semantic alignment), ArcFlow achieves a substantial

Table 1: Quantitative comparisons on Geneval, DPG-Bench, ImageReward, HPSv2 and OneIG-Bench. The NFE of Qwen-Image-20B is recorded as 50×2 since it uses CFG (Ho & Salimans, 2022).

Model	NFE↓	Geneval↑	DPG-Bench↑	ImageReward↑	HPSv2↑	OneIG-Bench				
						Alignment↑	Text↑	Diversity↑	Style↑	Reasoning↑
FLUX.1-dev	50	0.66	84.16	0.915	31.76	0.790	0.556	0.238	0.307	0.257
SenseFlow (FLUX)	2	0.60	79.86	0.901	30.42	0.743	0.230	0.139	0.341	0.212
Pi-Flow (GM-FLUX)	2	0.58	82.36	0.736	28.46	0.764	0.141	**0.216**	0.332	0.212
ArcFlow-FLUX (Ours)	2	**0.65**	**84.29**	**0.907**	**31.22**	**0.798**	**0.368**	0.210	**0.350**	**0.224**
Pi-Flow (GM-FLUX)	4	**0.66**	84.14	0.909	31.09	0.801	**0.454**	0.228	0.356	0.253
ArcFlow-FLUX (Ours)	4	**0.66**	**84.15**	**0.911**	**31.43**	**0.812**	**0.454**	**0.231**	**0.362**	**0.255**
Qwen-Image-20B	50×2	0.87	88.32	1.203	32.09	0.880	0.888	0.194	0.427	0.306
Qwen-Image-Lightning	2	**0.85**	88.42	0.967	29.83	0.875	**0.879**	0.098	0.415	**0.292**
pi-Flow (GM-Qwen)	2	0.83	86.45	0.922	27.56	0.837	0.634	0.176	0.382	0.259
TwinFlow (Qwen)	2	0.82	87.01	0.980	30.08	0.862	0.825	0.130	0.364	0.267
ArcFlow-Qwen (Ours)	2	**0.85**	**88.46**	**1.140**	**30.57**	**0.877**	0.853	**0.182**	**0.421**	0.289
pi-Flow (GM-Qwen)	4	0.84	88.42	**1.145**	31.37	0.879	0.884	0.181	0.434	0.294
ArcFlow-Qwen (Ours)	4	**0.85**	**88.45**	1.143	**32.46**	**0.886**	**0.901**	**0.185**	**0.440**	**0.302**

Table 2: Quantitative comparisons on Align5000 and COCO-10K. FID and pFID use 50-step teacher generations as the reference on Align5000 and real images on COCO-10K.

Model	NFE↓	Align5000		COCO-10K		Model	NFE↓	Align5000		COCO-10K	
		FID↓	pFID↓	FID↓	pFID↓			FID↓	pFID↓	FID↓	pFID↓
Qwen-Image-20B	50×2	–	–	31.78	55.03	FLUX.1-dev	50	–	–	27.45	43.83
Qwen-Image-Lightning	2	16.86	11.32	38.04	71.86	SenseFlow (FLUX)	2	27.55	**9.25**	31.47	48.29
pi-Flow (GM-Qwen)	2	20.07	12.42	40.51	67.39	Pi-Flow (GM-FLUX)	2	32.62	37.84	35.67	48.46
TwinFlow (Qwen)	2	16.77	4.34	39.35	58.59	ArcFlow-FLUX (Ours)	2	**16.83**	11.20	**27.76**	**45.17**
ArcFlow-Qwen (Ours)	2	**12.40**	**3.78**	**34.14**	**57.01**						
pi-Flow (GM-Qwen)	4	13.48	6.07	35.29	56.50	Pi-Flow (GM-FLUX)	4	16.54	13.92	28.37	**42.26**
ArcFlow-Qwen (Ours)	4	**12.24**	**4.01**	**32.80**	**55.46**	ArcFlow-FLUX (Ours)	4	**16.39**	**12.51**	**27.55**	42.81

+85.7% improvement in Diversity on OneIG-Bench, proving that our parameterization effectively preserves the teacher's pre-trained priors and the generative diveristy. Furthermore, as shown in Table 2, ArcFlow achieves the lowest FID on both Align5000 and COCO-10K across both backbones among 2-NFE methods, demonstrating strong fidelity to both teacher and real-image distributions. Notably, although Qwen-Image-Lightning achieves competitive prompt-following scores, its higher teacher-reference FID highlights a trade-off where perceptual optimization compromises trajectory fidelity, whereas ArcFlow maintains high fidelity to the original distribution. We provide more detailed dicussion on this distinction in Section G.5. For a fair comparison with pi-Flow, we also evaluate at 4 NFE, where ArcFlow maintains strong performance.

Qualitative Results. We conduct a qualitative comparison between ArcFlow and prior state-of-the-art methods by generating images from the same batch of initialized noise and comparing them with the teacher outputs, as shown in Figure 3. Linear distillation methods, including TwinFlow and Qwen-Image-Lightning, exhibit clear mode collapse and quality degradation, often producing nearly identical samples. Moreover, TwinFlow suffers from degraded visual aesthetics (the 3rd column), while Qwen-Image-Lightning shows blurred textures (background in the 3rd column) and structural artifacts (bent or duplicated swords in the 2nd column). These failures reveal a fundamental limitation of linear-step distillation methods in comprehensively approximating the teacher trajectory. In contrast, at the same batch of initialized noise, ArcFlow consistently preserves both high visual quality and generation diversity, producing results that are more closely aligned with the teacher. This proves its superiority in both high-quality generation and high-precision approximation of the teacher, ensuring generation diversity.

Although Qwen-Image-Lightning achieves competitive quantitative performance in the benchmarks, zoomed-in comparisons in Figure 4 show that ArcFlow yields noticeably finer and more coherent details. We attribute this discrepancy to the training objective of Qwen-Image-Lightning, which may sacrifice fine-grained visual fidelity for better semantic alignment objective, underscoring the inherent challenge of linear-step trajectory approximation (see more discussion in Section G.5).

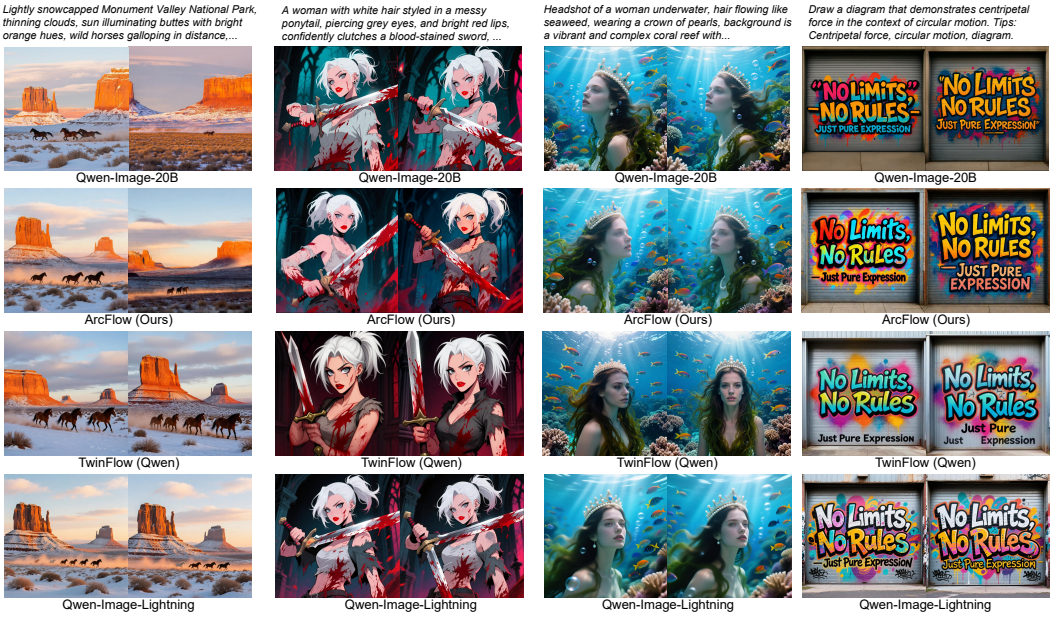

Figure 3: Qualitative comparisons with methods distilled on Qwen-Image-20B (2NFE). Every column contains two images which are generated from the same batch of initial noise. ArcFlow generates diverse samples that better align with teacher than competitors.

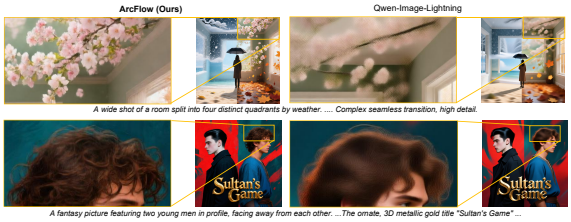

Figure 4: Qualitative comparisons with Qwen-Image-Lightning. Our ArcFlow exhibits visibly clearer details.

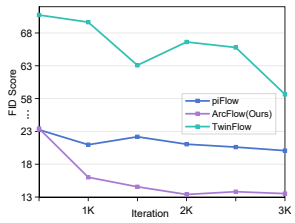

Figure 5: Teacher-reference FID during controlled 2-NFE distillation on Qwen-Image-20B.

4.3 Discussions on the Effectiveness and Efficiency of Momentum Mixtures

To further investigate why ArcFlow's momentum-mixture parameterization is effective and efficient, we conduct a series of targeted experiments on 2-NFE distillation of Qwen-Image-20B. We first assess its alignment with teacher velocities and trajectories, then disentangle momentum and mixture contributions from the parameterization, next validate whether modes actually learn distinct and useful roles, and finally evaluate its practical convergence and training efficiency. Detailed experimental settings and full analyses are in Appendix E.

Comparing students' trajectory alignment with teacher. We first validate whether ArcFlow better captures the teacher's velocity properties than Qwen-Image-Lightning, a linear distillation baseline. On 1,000 HPSv2 prompts with shared initial noise, we first assess teacher trajectory properties by measuring changes in its velocity direction and magnitude at adjacent timesteps and over longer intervals, and compare them with those of both students. Table 5 shows that the teacher's velocities are locally smooth but change substantially over longer intervals, motivating nonlinear trajectory modeling. ArcFlow reproduces these changes more faithfully than Qwem-Image-Lightning.

We then examine whether capturing these aggregate dynamics translates into closer alignment along individual trajectories. Using latent-state MSE and the cosine similarity between corresponding student and teacher updates, we find that ArcFlow reduces MSE by 58.5% and improves directional similarity from 0.62 to 0.85 relative to Qwen-Image-Lightning (Table 6). Together, these results support our motivation: modeling velocity evolution helps ArcFlow capture the teacher's trajectory properties and align with its intermediate states. Refer to Appendix E.1 for measurement protocols, metric definitions, and full trajectory analyses.

Disentangling the contributions of two components in momentum mixtures. Having established ArcFlow's closer teacher alignment, we next separate the contributions of momentum dynamics and mixture capacity, and investigate how they complement each other. We compare linear modes ($\gamma_k = 1$) with learnable-momentum modes at different mixture number $K \in \{1, 4, 16\}$ under matched training settings. As shown in Table 7, larger mixtures can improve the performance of linear mode, partly compensating for its limited expressiveness. Nevertheless, combining mixture capacity with momentum achieves the lowest teacher-reference FID (14.56 versus 17.06 for linear modes at $K = 16$). This highlights their complementary strengths: mixtures supply multiple velocity components, while momentum allows their relative contributions to evolve across timesteps. Refer to Appendix E.2.1 for the controlled ablation protocol and complete results.

Visualization of learned distinct velocity modes. Since additional modes improve momentum-based trajectory approximation, we next examine whether the model actually learns and uses distinct components, rather than merely benefiting from a larger parameterization. As shown in Figure 6, we visualize representative modes from a $K = 16$ model, and find that some emphasize fine textures and contours, whereas others preserve coarse structures and dominant colors. To test whether these differences matter for generation, we remove or amplify individual modes; the resulting changes in boundaries, smoothness, and residual noise reveal distinct effects (Figure 7). Importantly, since we supervise only the combined velocity without assigning separate targets or predefined roles to individual modes during training, this observed specialization suggests that different modes tend to learn complementary patterns on their own through gradient updates. This indicates that the model can effectively learn and exploit the inductive bias introduced by the momentum mixture parameterization, validating its effectiveness. Refer to Appendix E.2.2 for visualization settings, weight-intervention experiments, and detailed observations.

Learning efficiency and convergence study. Since preceding analyses validate the effectiveness of ArcFlow's parameterization, to test whether this makes teacher trajectories easier to learn in practice, we train ArcFlow, pi-Flow, and TwinFlow on Qwen-Image-20B using the same data, batch size, 2-NFE setting and 3,000 training steps. We evaluate teacher-reference FID on Align5000 every 500 steps. As shown in Figure 5 and Table 8, ArcFlow converges faster, reaching FID 13.52 after 3,000 steps. The early emergence of coherent image structure (Figure 8) further suggests ArcFlow's effective reuse of pretrained generation priors. With the same teacher-supervision budget per update as pi-Flow (Table 10), ArcFlow translates its faster convergence into practical training savings: it reaches FID 20 in 1,000 updates (an estimated 11.54 hours on four A100 GPUs). These results support our motivation that momentum mixtures make nonlinear teacher dynamics easier to learn, enabling high-precision distillation with fewer adaptation. Refer to Appendix E.3 for controlled protocols, convergence visualizations, and training-cost measurements.

4.4 Ablation Study

We also conduct ablation studies on fixed versus learnable γ, shared versus mode-specific velocities and momentum factors, mode count K, and mixed trajectory integration, as detailed in Appendix F. These experiments show that learnable γ and mode-specific parameters improve fidelity, and that $K = 16$ is the most suitable mode count for balancing fidelity and computational cost.

5 Conclusion

In this paper, we proposed ArcFlow, a few-step distillation framework that explicitly employs non-linear trajectories to approximate the complex dynamics of pre-trained diffusion teachers. By approximating and parameterizing the velocity field as a mixture of continuous momentum processes, ArcFlow admits a closed-form analytic solver and enables accurate trajectory integration. We further introduced a flow distillation strategy to align the student's analytical trajectory with the teacher. Benefiting from the intrinsic non-linearity of the parameterization, ArcFlow ensures high-precision alignment with the teacher. Moreover, it avoids unstable adversarial objectives and invasive full-parameter training, leading to faster convergence and more efficient distillation. Extensive experiments demonstrated that ArcFlow consistently achieves superior generation quality with fewer trainable parameters compared to linear baselines. We believe ArcFlow highlights the importance of respecting the underlying flow dynamics for efficient generative inference.

Ethics statement

ArcFlow improves the efficiency of text-to-image generation by distilling pre-trained Qwen-Image-20B and FLUX.1-dev models. The reported experiments use existing prompt datasets and public benchmarks, some of which contain images of people (Section J).We also conduct a pairwise human preference study with 20 annotators (Section G.1). Distilled students may inherit biases and harmful-content capabilities from their teachers, while lower generation costs may amplify existing risks by making misleading imagery and privacy- or copyright-infringing content easier to produce at scale. The fidelity and preference metrics reported in this paper do not establish safety or fairness. We encourage responsible use consistent with applicable laws and model and dataset licenses, including content moderation, provenance measures, and application-specific safety assessment before deployment. These safeguards are not evaluated in this work; further discussion of potential benefits and misuse appears in Section K.

Reproducibility statement

The momentum-mixture formulation, analytic solver, and distillation objective are given in Sections 3.1 to 3.3, with training and inference procedures in Algorithms 1 and 2. The solver derivation and expressivity proof are provided in Sections D.1 and D.2. Sections C and 4.1 describe the pre-trained backbones, prompt data, LoRA configuration, optimization settings, and hardware. Section C.2 defines the benchmarks, metrics, and reference distributions, distinguishing teacher-reference evaluation on Align5000 from real-image-reference evaluation on COCO-10K. Controlled convergence and ablation protocols are described in Sections E and F. We will provide an anonymized code archive as supplementary material and plan to publicly release the code and model weights, subject to applicable licenses.

AI use statement

We used generative AI tools to assist with manuscript drafting and language refinement, retrieve potentially relevant literature, improve the readability of research code, and support the analysis and interpretation of experimental results. The authors independently checked the cited sources, reviewed code changes, and verified analyses and interpretations against the underlying experimental results. All final scientific decisions were made by the authors, who take full responsibility for the manuscript, methods, results, and claims. Separately, the use of Qwen-Image-20B and FLUX.1-dev as distillation teachers and generators of reference images is part of the research methodology described in Sections C.2 and 3.3.

References

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants, 2023. URL https://arxiv.org/abs/2209.15571.

John C Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

Jingjing Chang, Yixiao Fang, Peng Xing, Shuhan Wu, Wei Cheng, Rui Wang, Xianfang Zeng, Gang Yu, and Hai-Bao Chen. Oneig-bench: Omni-dimensional nuanced evaluation for image generation, 2025. URL https://arxiv.org/abs/2506.07977.

Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. pi-flow: Policy-based few-step generation via imitation distillation, 2025a. URL https://arxiv.org/abs/2510.14974.

Hansheng Chen, Kai Zhang, Hao Tan, Zexiang Xu, Fujun Luan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. Gaussian mixture flow matching models. In *ICML*, 2025b.

E. W. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, 1966.

Zhenglin Cheng, Peng Sun, Jianguo Li, and Tao Lin. Twinflow: Realizing one-step generation on large models with self-adversarial flows, 2025. URL https://arxiv.org/abs/2512.05150.

Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models, 2022. URL https://arxiv.org/abs/2204.00227.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv.org/abs/2403.03206.

Xingtong Ge, Xin Zhang, Tongda Xu, Yi Zhang, Xinjie Zhang, Yan Wang, and Jun Zhang. Senseflow: Scaling distribution matching for flow-based text-to-image distillation, 2025. URL https://arxiv.org/abs/2506.00523.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447.

Dhruba Ghosh, Hanna Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment, 2023. URL https://arxiv.org/abs/2310.11513.

Herbert Goldstein, Charles Poole, and John Safko. *Classical Mechanics*. Addison-Wesley, 3rd edition, 2002.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment, 2024.

Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023. URL https://arxiv.org/abs/2310.04378.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models, 2023. URL https://arxiv.org/abs/2210.03142.

ModelTC. Qwen-image-lightning. https://github.com/ModelTC/Qwen-Image-Lightning. GitHub repository, accessed 2026-01-23.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022. URL https://arxiv.org/abs/2202.00512.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation, 2023. URL https://arxiv.org/abs/2311.17042.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. URL https://arxiv.org/abs/2011.13456.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023. URL https://arxiv.org/abs/2303.01469.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation, 2023. URL https://arxiv.org/abs/2305.16213.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

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. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 15903–15935, 2023.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation, 2024. URL https://arxiv.org/abs/2311.18828.

Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Jintao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. *arXiv preprint arXiv:2510.08431*, 2025.

A PRELIMINARIES

In this section, we first introduce the Flow Matching framework and the Probability Flow ODE (PF-ODE). We then discuss the numerical simulation of this ODE, highlighting the difference between multi-step integration solvers and few-step solvers via distillation.

Flow Matching and Probability Flow ODE. Let $p(\mathbf{x}_0)$ denote the data distribution and $p(\mathbf{x}_1) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ the noise distribution. Flow Matching (Lipman et al., 2023) defines a probability trajectory that iteratively transforms $p(\mathbf{x}_1)$ to $p(\mathbf{x}_0)$ over timesteps $t \in [0, 1]$. This transformation is driven by the probability flow ODE, which is defined as:

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{u}^*(\mathbf{x}_t, t), \tag{7}$$

Integrating this velocity field $\mathbf{u}^*(\cdot)$ over timestep $t \in [0, 1]$ forms the flow trajectories from noise $\mathbf{x}_1$ to data $\mathbf{x}_0$, which we term as trajectory integration.

To construct this velocity field, we use the Conditional Flow Matching (CFM) (Lipman et al., 2023) formulation. We define a linear trajectory between a data sample $\mathbf{x}_0$ and noise $\mathbf{x}_1$ to derive the latent $\mathbf{x}_t$ at any timestep t:

$$\mathbf{u}(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) = \frac{d}{dt}\mathbf{x}_t = \mathbf{x}_1 - \mathbf{x}_0, \tag{8}$$

Therefore, the practical objective is to train a flow-matching model to approximate the marginal velocity field $\mathbf{u}_t(\cdot)$ via a predicted velocity field $\mathbf{v}_\theta(\mathbf{x}, t)$ parameterized by θ, through the minimization of the expectation over these conditional flow trajectories:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,p(\mathbf{x}_0),p(\mathbf{x}_1)}\left[\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{u}(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1)\|^2\right], \tag{9}$$

Once optimized, numerical solvers are employed to integrate the learned PF-ODE v_θ, formulated as $\frac{d\mathbf{x}_t}{dt} = \mathbf{v}_\theta(\mathbf{x}_t, t)$, enabling deterministic data generation.

ODE Sampling and Distillation. Inference requires integrating the PF-ODE in timesteps from $t = 1$ to $t = 0$, yielding the sample $\hat{\mathbf{x}}_0 = \mathbf{x}_1 + \int_1^0 v_\theta(\mathbf{x}_\tau, \tau)d\tau$. Since this integral is analytically intractable, numerical solvers (e.g., Euler or Heun (Butcher, 2016)) are employed to approximate the integration by accumulating discrete velocities in multiple timesteps. To minimize the discretization error, this approximation typically involves $40 \sim 100$ function evaluations (NFEs). With step size Δt, an Euler solver iteratively updates x_t with the formula as follows:

$$\mathbf{x}_{t-\Delta t} = \mathbf{x}_t - v_\theta(\mathbf{x}_t, t) \cdot \Delta t, \tag{10}$$

However, this iterative process imposes a significant computational bottleneck. To mitigate this, knowledge distillation is widely adopted, where the student v_ϕ is trained to emulate the behavior of the teacher v_θ across multiple timesteps, allowing it to traverse the trajectory with fewer NFEs. The standard distillation objective is typically formulated as a regression problem, where the student minimizes the discrepancy between its predicted velocity field and a target signal $\mathbf{y}_t$ derived from the teacher:

$$\mathcal{L}_{\text{KD}}(\phi) = \mathbb{E}_{t,\mathbf{x}_t}\left[\|v_\phi(\mathbf{x}_t, t) - \mathbf{y}_t(\mathbf{x}_t, v_\theta)\|^2\right], \tag{11}$$

where $\mathbf{y}_t$ refers to the supervision target from the teacher, which varies depending on the specific method.

B ADDITIONAL TECHNICAL DETAILS

B.1 DISTILLATION AND INFERENCE ALGORITHMS

Algorithms 1 and 2 summarize training and the analytic inference transition. The following subsections specify mixed integration and momentum-factor parameterization.

B.2 MIXED TRAJECTORY INTEGRATION

This section details the mixed latent integration used to construct the velocity-supervision states in Section 3.3.

Algorithm 1 Flow Distillation for ArcFlow

Require: NFE, Teacher G_ψ, Student G_ϕ, Ratio λ

Sample start timestep $t_{\text{src}} \in \{\frac{1}{\text{NFE}}, \dots, 1\}$
Initialize $\mathbf{x}_{t_{\text{src}}}$
Sample timesteps $\{t_1, \dots, t_K\} \subseteq [t_{\text{src}} - \frac{1}{\text{NFE}}, t_{\text{src}}]$
$\Theta \leftarrow G_\phi(\mathbf{x}_{\text{src}}, t_{\text{src}})$;
for $t \in \{t_1, \dots, t_K\}$ **do**
  $\mathbf{x}_t \leftarrow \text{MixedIntegration}(\mathbf{x}_{\text{src}}, t_{\text{src}}, t; \Theta, \lambda)$
  $\hat{\mathbf{x}}_t \leftarrow \text{stopgrad}(\mathbf{x}_t)$
  $\mathbf{v}_{\text{stu}} \leftarrow \mathbf{v}(\hat{\mathbf{x}}_t, t; \Theta)$
  $\mathbf{u} \leftarrow G_\psi(\hat{\mathbf{x}}_t, t)$
  $\mathcal{L} \leftarrow \mathcal{L} + \|\mathbf{v}_{\text{stu}} - \mathbf{u}\|^2$
end for
Update ϕ using $\nabla_\phi \mathcal{L}$

Algorithm 2 Inference for ArcFlow ($t_{\text{src}} \to t_{\text{end}}$)

Require: $\mathbf{x}_{t_{\text{src}}}$, t_{src}, t_{end}, Student G_ϕ // Step 1: Single forward pass

$\{\mathbf{v}_k, \log \gamma_k, \log \omega_k\}_{k=1}^K \leftarrow G_\phi(\mathbf{x}_{t_{\text{src}}}, t_{\text{src}})$
$\pi_k \leftarrow \text{softmax}(\log \omega_k)$
$\gamma_k \leftarrow \exp(\log \gamma_k)$ // Step 2: Compute closed-form displacement
$\mathcal{C}_k \leftarrow \frac{\gamma_k^{1-t_{\text{end}}} - \gamma_k^{1-t_{\text{src}}}}{\ln \gamma_k}$ {*Base case:* $t_{src} - t_{end}$ *if* $\gamma_k \to 1$}
$\Phi \leftarrow \sum_{k=1}^K \pi_k \cdot \mathbf{v}_k \cdot \mathcal{C}_k$ // Step 3: Update latent state
$\mathbf{x}_{t_{\text{end}}} \leftarrow \mathbf{x}_{t_{\text{src}}} - \Phi$
return $\mathbf{x}_{t_{\text{end}}}$

Sequential Teacher–Student Handoff. For an interval from t_{src} to t_{dst}, we sample n intermediate timesteps and order them as $t_{\text{src}} = t_0 > t_1 > \cdots > t_n > t_{n+1} = t_{\text{dst}}$. Starting from the latent $\mathbf{x}_{t_0}$, we sequentially construct $\mathbf{x}_{t_{i+1}}$ from $\mathbf{x}_{t_i}$ for $i = 0, \dots, n$, where $\mathbf{x}_{t_i}$ denotes the latent state at timestep t_i. Within each sub-interval, the switching timestep is $t_{\text{mix}} = t_i - (1-\lambda)(t_i - t_{i+1})$, with the mixing coefficient $\lambda \in [0, 1]$ gradually increased during training. The teacher advances the latent from t_i to t_{mix}, after which the student completes the transition to t_{i+1}. Following the decreasing-time update convention in Section 3.2, this sequential handoff yields:

$$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} - \int_{t_{\text{mix}}}^{t_i} \mathbf{u}(\mathbf{x}_{t_i}, t_i)\, dt - \int_{t_{i+1}}^{t_{\text{mix}}} \mathbf{v}(\mathbf{x}_t, t; \Theta)\, dt, \tag{12}$$

Here, $\mathbf{u}(\mathbf{x}_{t_i}, t_i)$ is the instantaneous velocity predicted by the frozen teacher at $(\mathbf{x}_{t_i}, t_i)$. The student velocity $\mathbf{v}(\mathbf{x}_t, t; \Theta)$ is derived via Eq. 2 from the momentum parameters $\Theta = \{\pi_k, \mathbf{v}_k, \gamma_k\}_{k=1}^K$ predicted at $(\mathbf{x}_{t_0}, t_0)$ and held fixed throughout the interval. The K modes comprise gating probabilities π_k, basic velocities $\mathbf{v}_k$, and momentum factors γ_k. The intermediate states are detached before velocity matching, as summarized in Algorithm 1.

Teacher Integration. For the teacher phase within each sub-interval $[t_i, t_{\text{mix}}]$, we use the instantaneous velocity prediction $\mathbf{u}(\mathbf{x}_{t_i}, t_i)$ as a constant velocity. Since each sub-interval is sufficiently small, this approximation significantly reduces computational cost without affecting training stability.

Student Integration via Analytic Transition. Using the same Θ predicted at t_0, we evaluate the student portion without re-evaluating ArcFlow at the switching state. The analytic transition operator Φ in Eq. 4 gives the integrated student velocity from t_{mix} to t_{i+1}:

$$\begin{aligned} \int_{t_{i+1}}^{t_{\text{mix}}} \mathbf{v}(\mathbf{x}_t, t; \Theta)\, dt &= \int_{t_{i+1}}^{t_0} \mathbf{v}(\mathbf{x}_t, t; \Theta)\, dt - \int_{t_{\text{mix}}}^{t_0} \mathbf{v}(\mathbf{x}_t, t; \Theta)\, dt \\ &= \Phi(\mathbf{x}_{t_0}, t_0, t_{i+1}; \Theta) - \Phi(\mathbf{x}_{t_0}, t_0, t_{\text{mix}}; \Theta), \end{aligned} \tag{13}$$

This formulation allows efficient and exact integration of the student dynamics over arbitrary time intervals without numerical solvers.

Practical Implementation of the Mixing Schedule. The mixing coefficient λ governs the transition from teacher-guided ($\lambda = 0$) to fully student-driven integration ($\lambda = 1$). In our implementation, we employ a linear schedule for λ over a warm-up period of T iterations, where T is defined as the *Mixed Trajectory Guidance Steps* (refer to Table 3 for specific values). Formally, the schedule is defined as:

$$\lambda = \min\left(1, \frac{\text{iter}}{T}\right),$$

where iter denotes the current training iteration.

This design is specifically intended to stabilize the early stages of training by leveraging teacher guidance for integration. In practice, T is set to a relatively small value (e.g., 1,000 steps), after which λ remains fixed at 1 for the vast majority of the training process. Consequently, the model is then trained to predict parameters and integrate trajectories in an student-driven manner, which is consistent with the inference phase.

B.3 Momentum Factor Related Setting.

Log-parameterization of the Momentum Factor γ. According to the notion of momentum, the momentum factor γ is required to be strictly positive. However, directly regressing γ introduces an implicit positivity constraint, which may lead to optimization difficulties and numerical instability during training, especially when the predicted values approach zero. To address this, we parameterize the momentum factor in the logarithmic space. Specifically, in training, the momentum factor projection head is designed to predict $\log \gamma$ instead of γ, and the actual momentum factor is recovered by exponentiation. This reparameterization naturally enforces the positivity constraint while providing smoother gradients and more stable optimization behavior in practice.

Projection Layer Initialization. To ensure that the model captures a diverse range of trajectory dynamics across timesteps, we initialize the momentum factors $\{\gamma_k\}_{k=1}^{K}$ as a geometric progression spanning the interval $[0.5, 4.0]$. This range allows the mixture to cover both decelerating regimes ($\gamma < 1$) and accelerating regimes ($\gamma > 1$), enabling flexible modeling of complex flow trajectories. In implementation, we first construct the geometric sequence in the γ space and then convert it to the logarithmic domain. Since the momentum factor projection layer is a linear layer, we initialize its weight matrix to zeros and assign the corresponding $\log \gamma_k$ values to the bias vector. As a result, at the beginning of training, the predicted momentum factors exactly match the predefined geometric progression, providing a stable and interpretable initialization.

Crucially, we explicitly constrain one specific mode to be fixed at $\gamma = 1$. This design introduces a linear inductive bias, serving as a stable anchor that allows the model to naturally fall back to standard linear flow matching when the velocity evolution is negligible.

Learning Rate for the Momentum Factor Projection Layer. Since the momentum factor projection layer predicts $\log \gamma$ rather than γ, as shown in Eq. 2, updates to this layer lead to exponential changes in the effective velocity field. Consequently, using the same learning rate as other network components may cause unstable updates.

To improve numerical stability during training, we apply a reduced learning rate to this projection layer, specifically setting it to $0.1\times$ that of all other trainable layers. This targeted adjustment effectively stabilizes optimization while preserving sufficient learning capacity for adapting the momentum factors.

C Training and Evaluation Protocols

C.1 Final-Checkpoint Training

The configurations below apply to the final ArcFlow checkpoints. The training settings for the convergence and ablation experiments are given separately in Sections E and F.

Table 3: Detailed training configurations for distillation on Qwen-Image-20B and FLUX.1-dev.

Configuration	ArcFlow-Qwen	ArcFlow-FLUX
Method-specific Settings		
Num of momentum modes K	16	16
γ initialization range	[0.5, 4.0]	[0.5, 4.0]
Num of intermediate timesteps	4	4
Trained NFEs	2	2
Mixed Trajectory Guidance Steps	1000	2000
Training Details		
Batch Size	384	384
Total Training Steps	7500	8000
Optimizer Settings		
Optimizer	AdamW	AdamW
Learning rate	$1e^{-4}$	$1e^{-4}$
Learning rate for γ	$1e^{-5}$	$1e^{-5}$
Weight Decay	0	0
(β_1, β_2)	(0.9, 0.95)	(0.9, 0.95)

We freeze the teacher backbone and only train LoRA adapters together with extended output heads for predicting velocities, momentum factors, and gating probabilities.

For Qwen-Image-20B, we insert rank-256 LoRA adapters into a small subset of modules, including the image MLP, timestep embedding layers, and the text MLP blocks of the transformer. Specifically, LoRA is applied to the projection layers of the image MLP, both linear layers of the timestep embedder, and the text MLPs across all transformer blocks, while all remaining parameters are kept frozen.

For FLUX.1-dev, we apply rank-256 LoRA adapters to the projection and feed-forward modules that dominate the model's conditional and feature transformation capacity. Specifically, LoRA is inserted into the MLP projection layers, the output projection head, the feed-forward networks in both the main and context branches, as well as the timestep embedding layers. All other parameters of the teacher backbone are kept frozen.

We conduct our experiment on 96 H100 GPUs. All models are trained with BF16 mixed precision. We detail our other training configurations in Table 3.

C.2 Evaluation Datasets and Metric Definitions

Quantitative image-generation comparisons use 1024×1024 outputs. GenEval (Ghosh et al., 2023) measures compositional object generation, DPG-Bench (Hu et al., 2024) evaluates dense and long prompts, ImageReward (Xu et al., 2023) and HPSv2 (Wu et al., 2023) provide learned human-preference scores, and OneIG-Bench (Chang et al., 2025) separately reports alignment, text, diversity, style, and reasoning.

Align5000 contains 3,200 prompts from the HPSv2 prompt set and 1,800 prompts randomly sampled from COCO 2014 validation (Lin et al., 2015). This combines diverse artistic styles with natural-image descriptions. FID and patch FID (pFID) use images generated by the 50-step teacher as the reference. Lower values indicate closer alignment with the teacher's image distribution. These metrics do not directly measure human preference. ImageReward and HPSv2 provide learned preference scores, while Section G.1 reports a separate study with human annotators. Patch FID uses patch size 64 and stride 128. The Qwen teacher uses classifier-free guidance, so its 50 integration steps require 50×2 backbone evaluations.

COCO-10K uses 10,000 randomly sampled captions from COCO 2014 validation, with real images as the reference for FID and pFID.

The trajectory experiments in Section E.1 use 1,000 HPSv2 prompts to compare latent states, update directions, and velocity changes at the sampled timesteps. Align5000 instead evaluates the generated

images. Unless explicitly stated otherwise, FID results in the appendix use the Align5000 teacher reference.

D Theoretical Analysis

D.1 Implementation and Derivation of the Analytic Transition Operator Φ

This supplement provides the step-by-step derivation of Eq. 4 and the limiting behavior $\gamma \to 1$.

Expansion of the velocity mixture. Assume the ArcFlow velocity at base state $\mathbf{x}_{t_s}$ is expressed as

$$\mathbf{v}_\theta(\mathbf{x}_{t_s}, t) = \sum_{k=1}^{K} \pi_k(\mathbf{x}_{t_s})\ \mathbf{v}_k(\mathbf{x}_{t_s})\ \gamma_k(\mathbf{x}_{t_s})^{\,1-t},$$

where all mode-dependent quantities $\pi_k, \mathbf{v}_k, \gamma_k$ are evaluated at $\mathbf{x}_{t_s}$ and considered constant w.r.t. the integration variable t.

Then

$$\begin{aligned}\Phi(\mathbf{x}_{t_s}, t_s, t_e; \theta) &= \int_{t_e}^{t_s} \mathbf{v}_\theta(\mathbf{x}_{t_s}, t)\, dt \\ &= \sum_{k=1}^{K} \pi_k(\mathbf{x}_{t_s})\, \mathbf{v}_k(\mathbf{x}_{t_s}) \int_{t_e}^{t_s} \gamma_k(\mathbf{x}_{t_s})^{\,1-t}\, dt, \end{aligned} \tag{14}$$

Define the scalar integral

$$I(\gamma; t_s, t_e) \triangleq \int_{t_e}^{t_s} \gamma^{\,1-t}\, dt,$$

For $\gamma \neq 1$ we compute

$$\begin{aligned} I(\gamma; t_s, t_e) &= \int_{t_e}^{t_s} e^{(1-t)\ln\gamma}\, dt = \int_{t_e}^{t_s} e^{\ln\gamma}\, e^{-t\ln\gamma}\, dt \\ &= \gamma \int_{t_e}^{t_s} e^{-t\ln\gamma}\, dt = \gamma \cdot \frac{e^{-t\ln\gamma}}{-\ln\gamma}\Big|_{t_e}^{t_s} \\ &= \frac{\gamma^{1-t_e} - \gamma^{1-t_s}}{\ln\gamma}, \end{aligned} \tag{15}$$

This yields Eq. 5 for $\gamma \neq 1$.

Singular case $\gamma = 1$ and continuity. When $\gamma = 1$ the integrand equals 1, hence $I(1; t_s, t_e) = t_s - t_e$. We also show the limit $\lim_{\gamma\to 1} I(\gamma; t_s, t_e) = t_s - t_e$ to prove continuity. Set $\gamma = e^h$ with $h \to 0$. Then

$$I(e^h; t_s, t_e) = \frac{e^{h(1-t_e)} - e^{h(1-t_s)}}{h},$$

As $h \to 0$, apply Taylor expansion (or equivalently L'Hôpital's rule via h):

$$\lim_{h\to 0} \frac{e^{h(1-t_e)} - e^{h(1-t_s)}}{h} = (1 - t_e) - (1 - t_s) = t_s - t_e,$$

Thus the coefficient is continuous at $\gamma = 1$, and the analytic expression recovers the linear dynamic mode.

Full analytic operator. Combining Eq.14 and Eq.15 yields

$$\Phi(\mathbf{x}_{t_s}, t_s, t_e; \theta) = \sum_{k=1}^{K} \pi_k(\mathbf{x}_{t_s})\, \mathbf{v}_k(\mathbf{x}_{t_s})\, \mathcal{C}(\gamma_k(\mathbf{x}_{t_s}), t_s, t_e),$$

with $\mathcal{C}(\cdot)$ as in Eq. 5.

Numerical remarks. For numerical stability when γ is very close to 1, we branch to the second case $(t_s - t_e)$ when $|\ln \gamma| < \epsilon$ ($\epsilon = 10^{-6}$).

D.2 Proof of Theorem 3.1

In this section, we provide the proof for Theorem 3.1. We demonstrate that the momentum parameterization in ArcFlow can accurately approximate any ground truth trajectory at N discrete timesteps by using only $K = N$ momentum modes.

D.2.1 Problem Reformulation

Let $\{t_1, \ldots, t_N\}$ denote the set of sampled distinct timesteps, and let $\mathbf{u}_n^* = \mathbf{u}^*(\mathbf{y}, t_n) \in \mathbb{R}^D$ represent the corresponding ground truth velocities for any latent state $\mathbf{y}$ sampled from the data manifold. Our objective is to demonstrate that there exists a parameter set $\theta = \{\pi_k, \mathbf{v}_k, \gamma_k\}_{k=1}^K$ that exactly satisfies the conditions:

$$\sum_{k=1}^{K} \pi_k \mathbf{v}_k \gamma_k^{1-t_n} = \mathbf{u}_n^*, \quad \forall n \in \{1, \ldots, N\}, \tag{16}$$

where $K = N$. Directly solving Eq. 16 is complicated by the bilinear coupling between π_k and $\mathbf{v}_k$. Therefore, we introduce the composite parameter $\mathbf{w}_k \triangleq \pi_k \mathbf{v}_k$. The independence of the D dimensions allows us to decouple the problem into D identical scalar equations. Thus, we focus on a single scalar dimension, letting w_k and u_n^* denote the scalar components of $\mathbf{w}_k$ and $\mathbf{u}_n^*$, respectively.

Since our goal is to establish the existence of at least one feasible parameter set θ, we may fix a subset of the parameters without loss of generality. Specifically, we fix the momentum factors $\Gamma = \{\gamma_1, \ldots, \gamma_N\}$ to be arbitrary distinct positive real values. With Γ fixed, the exponential terms become known constants, and the problem of finding $\{\pi_k, \mathbf{v}_k\}$ reduces to solving for the composite weights w_k. This reduction holds because any valid solution for w_k guarantees the existence of π_k and $\mathbf{v}_k$. Consequently, the problem can be formulated as the following linear system:

$$\mathbf{M}\mathbf{c} = \mathbf{b}, \tag{17}$$

where $\mathbf{c} = [w_1, \ldots, w_K]^\top \in \mathbb{R}^K$ and $\mathbf{b} = [u_1^*, \ldots, u_N^*]^\top \in \mathbb{R}^N$. The matrix $\mathbf{M} \in \mathbb{R}^{N \times K}$ is the basis matrix determined by the fixed γ_k:

$$M_{nk} = \gamma_k^{1-t_n}, \tag{18}$$

Establishing the existence of θ for an arbitrary ground truth $\mathbf{b}$ is equivalent to guaranteeing that the linear system $\mathbf{M}\mathbf{c} = \mathbf{b}$ is solvable for any $\mathbf{b}$. This condition holds if and only if the basis matrix $\mathbf{M}$ is non-singular (invertible). Thus, the proof reduces to demonstrating the invertibility of $\mathbf{M}$.

D.2.2 Proving Invertibility via Chebyshev Systems

The solvability of the linear system relies on the non-singularity of the basis matrix $\mathbf{M}$. To establish this, we frame the problem within the theory of Chebyshev Systems.

Definition D.1 (Chebyshev System)**.** Let $\{f_1, \ldots, f_N\}$ be a set of continuous functions defined on an interval $\mathcal{I}$. This set constitutes a Chebyshev System if every non-trivial linear combination $F(t) = \sum_{k=1}^N c_k f_k(t)$ (where coefficients $c_k \in \mathbb{R}$ are not all simultaneously zero) possesses at most $N - 1$ distinct zeros in $\mathcal{I}$.

The significance of this definition lies in the Haar Condition, which directly links the zero-counting property of functions to the determinant of their basis matrix:

Lemma D.2 (Haar Condition Cheney (1966))**.** *If the set $\{f_1, \ldots, f_N\}$ forms a Chebyshev System on $\mathcal{I}$, then for any set of distinct sampling points $\{t_1, \ldots, t_N\} \subset \mathcal{I}$, the resulting matrix $\mathbf{\Phi}$ with entries $\Phi_{nk} = f_k(t_n)$ is non-singular.*

To apply Lemma D.2 to our specific problem, we must demonstrate that our proposed momentum dynamics functions satisfy the definition of a Chebyshev System.

Proposition D.3. *The set of functions $\{\gamma_k^{1-t}\}_{k=1}^N$, parameterized by distinct momentum factors $\gamma_k \in \mathbb{R}^+$, forms a Chebyshev System on $\mathbb{R}$.*

Proof. Let $F(t)$ be an arbitrary non-trivial linear combination of the basis functions with coefficients $c_k \in \mathbb{R}$. We can rewrite the expression as a generalized polynomial of exponentials:

$$F_N(t) = \sum_{k=1}^{N} c_k \gamma_k^{1-t} = \sum_{k=1}^{N} (c_k \gamma_k) e^{-(\ln \gamma_k)t} = \sum_{k=1}^{N} \alpha_k e^{\lambda_k t},$$

Here, we define the new coefficients $\alpha_k \triangleq c_k \gamma_k$ (which remain non-zero if c_k are non-zero) and the distinct exponents $\lambda_k \triangleq -\ln \gamma_k$. We prove that $F_N(t)$ has at most $N-1$ zeros by induction on N.

Base case ($N = 1$): $F_1(t) = \alpha_1 e^{\lambda_1 t}$. Since exponentials are strictly positive and $\alpha_1 \neq 0$, $F_1(t)$ has no zeros.

Inductive step: Assume that any linear combination of $N-1$ exponentials $F_{N-1}(t)$ has at most $N-2$ distinct zeros. Suppose, for contradiction, that $F_N(t)$ has N distinct zeros. Define the auxiliary function $G_N(t) = e^{-\lambda_1 t} F_N(t)$, which shares the same zeros as $F_N(t)$. Its derivative is:

$$G'_N(t) = \frac{d}{dt}\left(\alpha_1 + \sum_{k=2}^{N} \alpha_k e^{(\lambda_k - \lambda_1)t}\right) = \sum_{k=2}^{N} \alpha_k (\lambda_k - \lambda_1) e^{(\lambda_k - \lambda_1)t}.$$

Note that $G'_N(t)$ is a linear combination of $N-1$ exponentials with distinct exponents $\lambda_k - \lambda_1$. By the induction hypothesis, $G'_N(t)$ can have at most $N-2$ zeros. However, by Rolle's Theorem, if $G_N(t)$ has N distinct zeros, its derivative $G'_N(t)$ must have at least $N-1$ distinct zeros. This contradiction implies that $F_N(t)$ cannot have N distinct zeros. □

Conclusion. Since Proposition D.3 confirms that our basis functions form a Chebyshev System, Lemma D.2 ensures that the matrix $\mathbf{M}$ is invertible for any set of distinct timesteps. This guarantees the existence of a solution vector $\mathbf{c}$. Translating this mathematical result back to our original objective, the solvability of $\mathbf{c}$ implies that for any ground truth velocities $\mathbf{u}_n^*$, we can explicitly construct a parameter set θ (e.g., by setting $\sum \pi_k = 1$ and $\mathbf{v}_k$ from $\mathbf{c}$) that satisfies Eq. 16 exactly. This completes the proof of Theorem 3.1, theoretically validating that the proposed momentum parameterization possesses sufficient expressivity to perfectly align with arbitrary trajectory dynamics on the data manifold.

D.3 Computational Overhead Analysis

To evaluate the efficiency of our method, we provide a quantitative assessment of the additional computational cost incurred by the ArcFlow prediction head. Despite introducing new momentum parameters, the complexity of the added head remains negligible relative to the massive frozen backbones used in our experiments.

In our implementation, we use a default basis size of $K = 16$. The architectural parameters for the backbones (Qwen-Image and FLUX.1-dev) are summarized in Table 4.

Table 4: Architectural specifications of the backbone models.

Metric	**Qwen-Image**	**FLUX.1-dev**
Transformer Layers	60	19 + 38 = 57
Hidden Width (H)	$24 \times 128 = 3072$	$24 \times 128 = 3072$
Output Channels (D)	16	64

Based on the standard $2mnk$ FLOPs estimation for linear layers, the extra computational cost per token introduced by the prediction head is:

$$\text{FLOPs}_{\text{head}} \approx 2 \cdot H \cdot K(D+2). \tag{19}$$

where H is the hidden dimension, K is the number of basis functions, and D is the latent channel width. To calculate the relative increase compared to the backbone (approximated as $24 \cdot L \cdot H^2$ for L layers), we have:

- For Qwen:
$$\frac{2 \cdot 3072 \cdot 16 \cdot (16+2)}{24 \cdot 60 \cdot 3072^2} \approx 1.3 \times 10^{-4} = 0.013\%.$$
- For FLUX:
$$\frac{2 \cdot 3072 \cdot 16 \cdot (64+2)}{24 \cdot 57 \cdot 3072^2} \approx 5.0 \times 10^{-4} = 0.050\%.$$

The results demonstrate that the head of $K = 16$ increases the backbone computation by only approximately 0.01%–0.05%. This negligible overhead confirms that ArcFlow achieves efficient continuous trajectory modeling without compromising the inference latency or memory efficiency of the original large-scale models.

E Experimental Analysis of Momentum Mixtures

This section provides the experimental settings and results for Section 4.3. We measure teacher velocity changes across timesteps and student–teacher trajectory alignment, compare momentum settings and mixture sizes, visualize individual velocity modes, and compare convergence with the same data and number of training steps. The settings for each experiment are specified below.

E.1 Teacher Dynamics and Trajectory Alignment

A linear student uses a constant velocity within each inference interval, whereas the teacher velocity changes in direction and magnitude across timesteps. We measure these changes for the teacher and students, and then compare student and teacher latent states and update directions at the same timesteps.

E.1.1 Velocity Changes at Adjacent and Non-adjacent Timesteps

We use 1,000 HPSv2 prompts with the same initial noise for the 50-step Qwen-Image-20B teacher, ArcFlow, and Qwen-Image-Lightning. Both students generate with 2 NFEs. We record the teacher velocity at each of its 50 timesteps and compute the student velocities at the same timesteps. Within each inference interval, ArcFlow computes its velocity from the predicted momentum-mixture parameters, while Qwen-Image-Lightning uses a constant velocity.

Let $\mathbf{v}_{p,i}$ denote the velocity vector for prompt p at timestep t_i. The set $\mathcal{P}_\Delta$ contains the prompt indices $p \in \{1, \ldots, 1{,}000\}$ and timestep indices $i \in \{1, \ldots, 50 - \Delta\}$ used to compare $\mathbf{v}_{p,i}$ and $\mathbf{v}_{p,i+\Delta}$. We compute

$$D_\Delta = \frac{1}{|\mathcal{P}_\Delta|} \sum_{(p,i)\in\mathcal{P}_\Delta} \left(1 - \frac{\langle \mathbf{v}_{p,i}, \mathbf{v}_{p,i+\Delta} \rangle}{\|\mathbf{v}_{p,i}\|_2 \|\mathbf{v}_{p,i+\Delta}\|_2}\right), \tag{20}$$

$$R_\Delta = \frac{1}{|\mathcal{P}_\Delta|} \sum_{(p,i)\in\mathcal{P}_\Delta} \left|\log \frac{\|\mathbf{v}_{p,i+\Delta}\|_2}{\|\mathbf{v}_{p,i}\|_2}\right|. \tag{21}$$

We set $\Delta = 1$ to compare adjacent timesteps and $\Delta = 25$ to compare timesteps separated by 25 teacher steps. D_Δ measures the change in velocity direction, and R_Δ measures the absolute log ratio of velocity norms. We compare each student's values with the teacher's values to examine how closely the student follows the teacher's velocity dynamics.

We also compare the distributions of velocity changes at adjacent timesteps. For each model, the 1,000 trajectories provide $1{,}000 \times 49 = 49{,}000$ adjacent velocity pairs. We compute the directional change and absolute log norm ratio for each pair, then measure the 1-Wasserstein distance between the student and teacher distributions. We denote these distances by W_{dir} and W_{norm}, respectively; lower values indicate closer alignment with the teacher's velocity-change distributions.

Table 5 shows that the teacher velocity is locally smooth but not constant across timesteps: its small D_{adj} and R_{adj} indicate gradual changes in direction and magnitude, whereas the larger D_{span} and R_{span} reveal substantial evolution over 25 teacher steps. This property motivates explicitly modeling velocity changes within each inference interval. ArcFlow is designed to learn these dynamics through mode-specific momentum factors that change the relative contributions of basic velocities across

Table 5: Velocity changes on 1,000 HPSv2 prompts. D and R compare adjacent timesteps ($\Delta = 1$) and timesteps separated by 25 teacher steps ($\Delta = 25$). W_{dir} and W_{norm} compare the distributions of adjacent velocity changes with the teacher.

Method	D_{adj}	R_{adj}	D_{span}	R_{span}	$W_{\text{dir}} \downarrow$	$W_{\text{norm}} \downarrow$
Teacher (50-step)	0.0117	0.0157	0.1503	0.1010	–	–
Qwen-Image-Lightning	0.0012	0.0004	0.0577	0.0190	0.01094	0.01534
ArcFlow	0.0077	0.0194	0.0741	0.0790	**0.00782**	**0.01157**

Table 6: Student–teacher trajectory alignment on 1,000 HPSv2 prompts. Both students use 2 NFEs and the same initial noise as the teacher.

Method	Latent deviation MSE ↓	Update-direction cosine ↑
Qwen-Image-Lightning	0.0429	0.62
ArcFlow	**0.0178**	**0.85**

timesteps, allowing both direction and magnitude to evolve. Its adjacent and span statistics show that this parameterization can effectively reproduce the teacher's changes in velocity direction and magnitude at both scales. By learning and reproducing these dynamics, ArcFlow achieves closer alignment with the teacher than the piecewise-constant Qwen-Image-Lightning student under the same 2-NFE inference budget, as further reflected in its lower W_{dir} and W_{norm} for the velocity-change distributions.

E.1.2 Pointwise Trajectory Alignment

We next examine whether reproducing the teacher's velocity-change properties translates into alignment along individual trajectories. Using 1,000 HPSv2 prompts and shared initial noise, we sample latent states from the teacher and student trajectories at a common set of timesteps $\{t_i\}_{i=0}^{T}$. Let $\mathbf{x}_{p,i}^{\text{tea}}$ and $\mathbf{x}_{p,i}^{\text{stu}}$ denote these sampled states for prompt p. We use these samples to characterize trajectory alignment through latent-state MSE and the cosine similarity between corresponding latent updates, averaged over prompts and sampled timesteps:

$$\mathcal{D}_{\text{MSE}} = \frac{1}{NT} \sum_{p=1}^{N} \sum_{i=1}^{T} \|\mathbf{x}_{p,i}^{\text{stu}} - \mathbf{x}_{p,i}^{\text{tea}}\|_2^2, \tag{22}$$

$$\mathcal{S}_{\text{cosine}} = \frac{1}{NT} \sum_{p=1}^{N} \sum_{i=1}^{T} \frac{\langle \Delta\mathbf{x}_{p,i}^{\text{stu}}, \Delta\mathbf{x}_{p,i}^{\text{tea}} \rangle}{\|\Delta\mathbf{x}_{p,i}^{\text{stu}}\|_2 \|\Delta\mathbf{x}_{p,i}^{\text{tea}}\|_2}, \tag{23}$$

where $N = 1{,}000$ and $\Delta\mathbf{x}_{p,i} = \mathbf{x}_{p,i} - \mathbf{x}_{p,i-1}$ is the displacement between consecutive sampled states. Unlike the aggregate velocity-change statistics above, these metrics directly compare corresponding student and teacher states and update directions for each prompt. The checkpoints in this comparison are separate from those used in the momentum and mixture-size ablation below.

Table 6 shows that ArcFlow reduces latent-state MSE by 58.5% relative to Qwen-Image-Lightning and increases update-direction cosine similarity from 0.62 to 0.85. Thus, its closer teacher alignment is reflected in the sampled intermediate states and the directions of the updates between them, not only in the generated-image distribution.

Together, the two analyses show that ArcFlow effectively learns the teacher's trajectory dynamics and achieves closer trajectory alignment. We next examine the components of ArcFlow's parameterization to identify their contributions to trajectory learning and how they interact.

E.2 Momentum Dynamics, Capacity, and Mode Specialization

E.2.1 Disentangling Momentum Dynamics and Mixture Capacity

To separate the effect of momentum dynamics from that of mixture size K, we compare linear and momentum parameterizations at $K \in \{1, 4, 16\}$. The linear variant fixes all $\gamma_k = 1$, so the mixture

Table 7: Ablation separating mixture size K from momentum dynamics. Linear fixes $\gamma_k = 1$ for all modes; momentum uses learnable γ_k.

K	Parameterization	FID ↓	Latent deviation MSE ↓	Direction cosine ↑
1	$\gamma \equiv 1$ (linear)	36.13	0.0518	0.61
	Learnable γ (momentum)	**35.89**	**0.0497**	**0.62**
4	$\gamma \equiv 1$ (linear)	24.74	0.0359	0.70
	Learnable γ (momentum)	**20.97**	**0.0274**	**0.72**
16	$\gamma \equiv 1$ (linear)	17.06	0.0232	0.76
	Learnable γ (momentum)	**14.56**	**0.0204**	**0.79**

velocity $\sum_k \pi_k \mathbf{v}_k$ remains constant within each inference interval and produces a straight trajectory segment. The momentum variant learns γ_k, allowing each mode's contribution $\pi_k \mathbf{v}_k \gamma_k^{1-t}$ to vary across timesteps. Comparing the two variants at a fixed K isolates the contribution of momentum, while varying K within each variant examines the contribution of mixture size and its interaction with momentum.

We use Qwen-Image-20B and 2-NFE generation. All variants use the same training settings and 1,500 training steps. FID is measured on Align5000 against 50-step teacher outputs, and trajectory alignment is evaluated on 1,000 HPSv2 prompts using the metrics in Section E.1.2.

Table 7 shows that mixture capacity and momentum dynamics are complementary. At $K = 1$, momentum only changes velocity magnitude along a fixed direction, explaining the slight gain against the linear mode. At $K = 4$, distinct momentum factors reweight multiple basic velocities across timesteps, enabling changes in both direction and magnitude. The largest observed FID gap occurs at this moderate K (24.74 vs. 20.97), where the mixture is rich enough to exploit reweighting across timesteps but still offers limited compensation for its absence. At $K = 16$, the linear variant reaches 17.06 FID, while the matched-K FID gap between linear and momentum modes narrows from 3.77 to 2.50. This suggests that, with sufficiently many modes, a richer velocity decomposition can partially compensate for the lack of momentum by improving the learned interval-averaged velocity. Crucially, this improves the quality of the linear approximation rather than its expressiveness across timesteps. With learnable momentum, additional modes instead provide complementary velocity components with distinct evolution rates across timesteps, allowing the model to approximate more complex changes in direction and magnitude. Combining mixture diversity with momentum therefore yields the best tested configuration at $K = 16$, achieving FID 14.56 and the best latent-state MSE and direction cosine. Thus, larger mixtures can mitigate the performance cost of a linear approximation, but work best with momentum to capture the teacher's evolving velocity rather than only its average motion.

E.2.2 Visualization of Learned Velocity Modes

The preceding ablations show that multiple momentum modes improve the approximation of the teacher's velocity dynamics. We now investigate what individual modes learn and how their contributions affect the generated image. In Figure 6, we select three cases and visualize four representative modes (2, 4, 12, and 15) from the $K = 16$ modes for each case, alongside the original generated image. The corresponding prompts are included in the figure.

As shown in Figure 6, the modes exhibit distinct visual patterns across these cases. Mode 2 shows noisy, roughly contrast-inverted responses, whereas Mode 4 preserves the overall scene structure and dominant colors. Mode 12 emphasizes contours and fine textures, such as fur, knitted patterns, and facial details. Mode 15 retains image-aligned colors and silhouettes with pronounced fine-scale fluctuations. Thus, different modes capture different patterns within the scene rather than reproducing the same averaged prediction.

We further test their contributions by changing one mode's weight at a time in the apple-and-mug case (Figure 7). Here, π_k is the original learned weight and π'_k is the intervention weight: $\pi'_k = 0$ removes the mode, $\pi'_k = \pi_k$ keeps its original contribution, and $\pi'_k = 2\pi_k$ doubles it. Each row shows one mode followed by these three outputs.

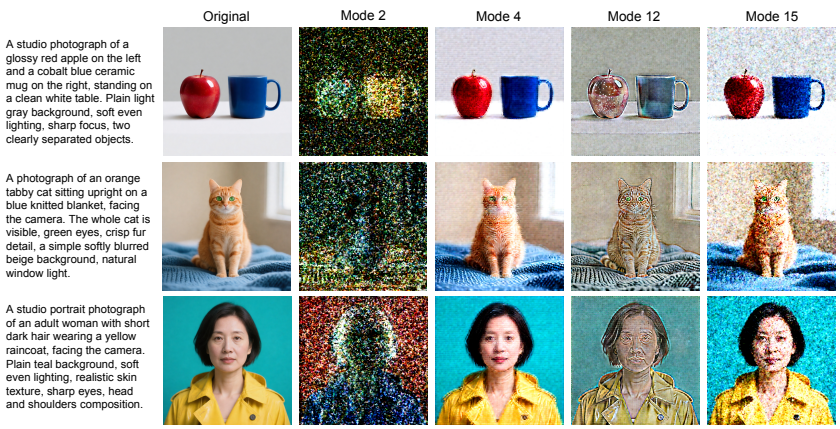

Figure 6: Representative velocity modes for three cases ($K = 16$). Each row shows the prompt, the original output using all modes, and Modes 2, 4, 12, and 15, which exhibit distinct visual patterns.

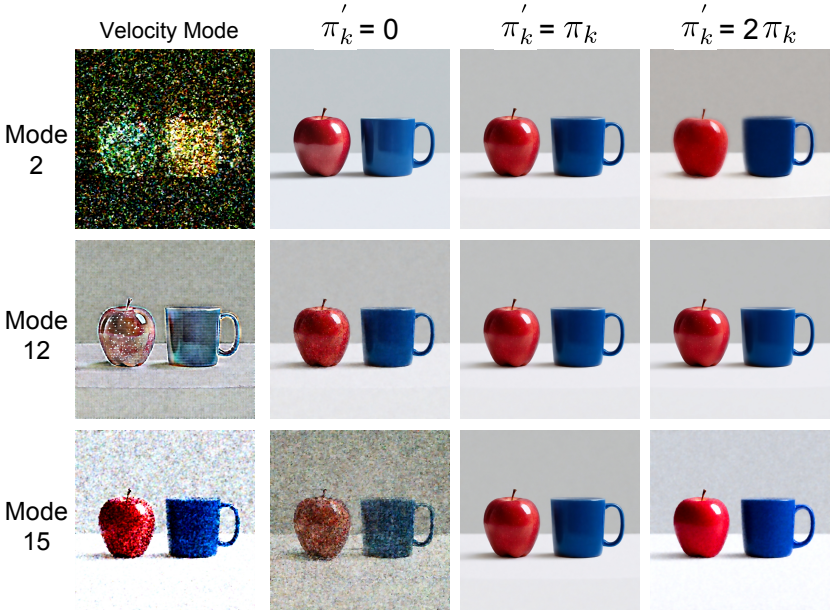

Figure 7: Weight interventions for Modes 2, 12, and 15 in the apple-and-mug case. Each row shows one mode followed by removal ($\pi_k' = 0$), the original contribution ($\pi_k' = \pi_k$), and a doubled contribution ($\pi_k' = 2\pi_k$). Other weights remain unchanged, with no renormalization.

As shown in Figure 7, removing Mode 2 makes tonal transitions less natural, whereas doubling it oversmooths object boundaries, suggesting a role in balancing local smoothness. Removing Mode 12 produces mottled surfaces and less clean contours, while doubling it yields crisper boundaries and smoother surfaces, consistent with edge and texture refinement. Removing Mode 15 introduces dense speckling, while doubling it causes excessive saturation and visible grain, indicating a contribution to residual-noise and color correction.

These distinct intervention effects support complementary, non-redundant contributions among the inspected modes. Crucially, training supervises only the combined velocity, without mode-specific targets or explicit incentives for specialization. This division of labor therefore emerges from the model's own gradient updates, showing that it actively exploits the momentum-mixture inductive bias during optimization rather than merely gaining representational capacity. The emergence of specialization without explicit encouragement suggests that this parameterization provides an

Table 8: Matched-budget comparison on Qwen-Image-20B at 2 NFE after 3,000 updates. FID and pFID use 50-step teacher outputs on Align5000; OneIG-Bench and HPSv2 evaluate generation quality.

Method	OneIG-Bench ↑					HPSv2 ↑	FID ↓	pFID ↓
	Alignment	Text	Diversity	Style	Reasoning			
pi-Flow	0.843	0.736	**0.173**	0.379	0.261	26.56	20.07	8.47
ArcFlow	**0.856**	**0.798**	**0.173**	**0.392**	**0.267**	**27.88**	**13.52**	**5.63**

easier path for gradient-based learning of complex teacher dynamics: individual modes can learn complementary components instead of each having to approximate the full dynamics.

E.3 Learning Efficiency under Matched Training Budgets

The preceding analyses show that momentum mixtures effectively capture the teacher's velocity dynamics, with individual modes learning complementary components during optimization. These findings support our claim that this parameterization reduces the difficulty of learning teacher trajectories, allowing ArcFlow to retain the teacher's image-generation priors and focus on the few-step trajectory approximation. We next test this claim under matched training budgets by examining how quickly teacher-reference FID decreases, how generated images improve during training, and the training cost required to reach a fixed FID.

E.3.1 Controlled Training Protocol

We conduct training of ArcFlow, pi-Flow, and TwinFlow on the same Qwen-Image-20B backbone, prompt dataset, batch size of 16, and 3,000 training steps for 2-NFE generation. We follow the public implementations of pi-Flow and TwinFlow and evaluate teacher-reference FID on Align5000 every 500 steps. ArcFlow and pi-Flow trains LoRA adapters and output heads, while TwinFlow uses full-parameter training. The methods retain their respective training objectives and trainable parameter sets.

For the controlled pi-Flow comparison, both methods use four intermediate supervision states per interval, the same mixed-trajectory guidance schedule, and 2×4 teacher evaluations per iteration. When adapting pi-Flow from its released 4-NFE setting, its two intermediate states per interval are increased to four. All directly comparable trainable components use a learning rate of 10^{-4}; only ArcFlow's momentum-factor head uses 10^{-5} because it predicts $\log \gamma$ (see Appendix B.3 for the detailed rationale).

E.3.2 Quantitative Convergence Curves and Matched-Budget Quality

Figure 5 shows teacher-reference FID every 500 training steps. Table 8 reports FID, pFID, OneIG-Bench, and HPSv2 for the checkpoints of ArcFlow and pi-Flow after 3,000 steps. ArcFlow reaches FID 13.52 compared with 20.07 for pi-Flow under the same data, batch size, training steps, and intermediate teacher-supervision settings. Together with the trajectory-alignment results, faster convergence using only LoRA adapters and output heads suggests that modeling velocity evolution across timesteps reduces the adaptation needed to align with the pretrained teacher. And the momentum mixture more effectively provides this representation, while its analytic integral computes the corresponding latent transition directly.

E.3.3 Qualitative Convergence

Figure 8 shows images generated from one prompt using the same checkpoints as the FID comparison.

As shown in Figure 8, ArcFlow already exhibits a coherent global structure after only 0.5K training iterations, with most stochastic artifacts and irregular noise largely suppressed. At this early stage, the generated images mainly suffer from mild over-smoothing, rather than structural corruption, indicating that the model has already entered a meaningful generative regime. This behavior suggests that ArcFlow can immediately benefit from its natural compatibility with the pre-trained teacher

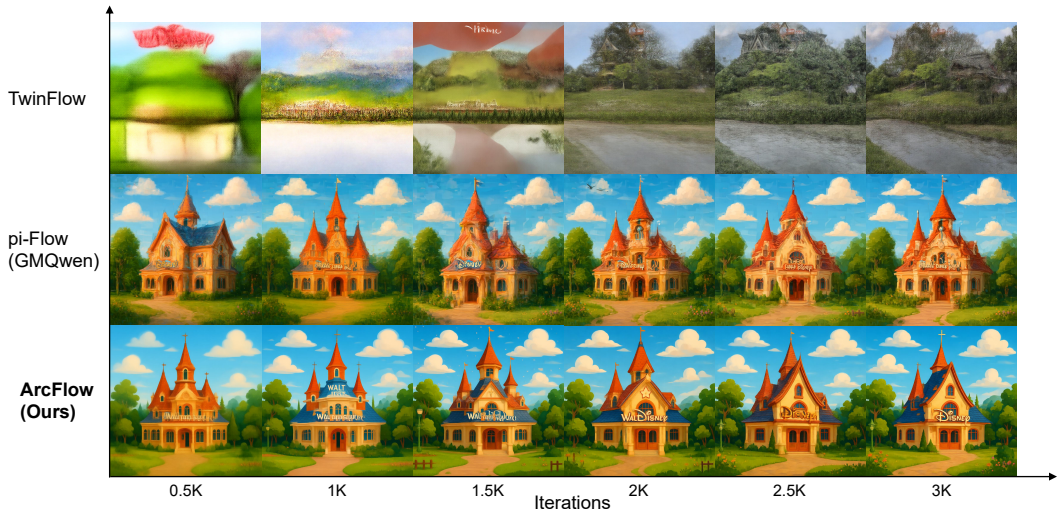

Figure 8: Qualitative convergence using the checkpoints from Figure 5. Columns correspond to 0.5K, 1K, 1.5K, 2K, 2.5K, and 3K training steps.

weights, enabling effective adaptation without requiring extensive retraining. The student can therefore retain the teacher's image-generation priors and focus training on the few-step trajectory approximation, instead of first relearning image structure.

As training proceeds, the visual quality of ArcFlow improves in a stable and monotonic manner. By 3K iterations, the generated images reach a level where no obvious visual defects can be identified by human inspection, demonstrating both fast convergence and strong training stability.

In comparison, pi-Flow preserves reasonable global structure at early iterations; however, it consistently struggles with residual noise artifacts throughout the training process. Even with increased iterations, the presence of scattered noise prevents a clear improvement in perceptual quality, resulting in noticeably slower and less stable convergence.

Moreover, TwinFlow exhibits the weakest convergence behavior. As a linear distillation method, its parameterization conflicts with the teacher's inherently complex trajectory, which prevents effective reuse of the teacher's pre-trained weights at initialization. Consequently, TwinFlow is forced to re-learn a viable representation from a high-error state, leading to significantly slower convergence and inferior visual quality during early and intermediate training stages.

E.3.4 Training Cost and Teacher Supervision

The preceding experiments establish ArcFlow's learning efficiency in terms of training iterations. We now examine how this translates into practical GPU training efficiency through the wall-clock time to reach teacher-reference FID 20 on Align5000 and the peak memory used during training. Table 9 reports iteration timings and the resulting estimates of time to the target on four NVIDIA A100 GPUs with the same batch size. The timings include teacher evaluations and each method's complete update procedure. These efficiency runs are separate from the final-checkpoint training in Section C.

ArcFlow reaches FID 20 in 1,000 updates on both backbones, corresponding to approximately 11.54 hours on Qwen-Image-20B and 10.44 hours on FLUX.1-dev. The corresponding pi-Flow runs use 33.35 and 30.39 hours for 3,000 updates and remain above the target. ArcFlow thus reaches the target with less than 35% of the time already spent by pi-Flow on each backbone. With the GPU count fixed, the same reduction applies to total GPU-hours. Its LoRA-and-head training also has a smaller memory footprint than TwinFlow on Qwen (39.2 vs. 45.1 GB) and SenseFlow on FLUX (27.6 vs. 66.2 GB).

ArcFlow makes effective use of teacher supervision by learning from every sampled trajectory point covered by a single student forward pass. This pass predicts a shared set of momentum-mixture parameters that defines the velocity evolution over the whole interval. Intermediate student velocities

Table 9: Training efficiency on four A100 GPUs. Wall-clock time to teacher-reference FID 20 is estimated as time per iteration multiplied by the required number of training steps. A greater-than sign indicates that the target had not been reached within the reported training budget.

Method	Time/iter (s)	Peak memory (GB)	Steps to FID 20	Time to FID 20 (h)
Qwen-Image-20B students				
TwinFlow	63.35	45.1	$> 3{,}000$	> 52.79
pi-Flow	40.02	33.35	$> 3{,}000$	> 33.35
ArcFlow	41.56	39.2	1,000	11.54
FLUX.1-dev students				
SenseFlow	59.74	66.2	$> 2{,}000$	> 33.19
pi-Flow	36.47	23.2	$> 3{,}000$	> 30.39
ArcFlow	37.58	27.6	1,000	10.44

Table 10: Forward evaluations per training iteration for the implementations timed in Table 9. One SenseFlow iteration includes one generator update and five critic updates.

Method	Teacher calls	Total forward calls
SenseFlow	1	$1 + 1 + 2 \times 5$
TwinFlow	1	$1 + 1 + 1$
pi-Flow	2×4	$1 + 2 \times 4$
ArcFlow	2×4	$1 + 2 \times 4$

and transitions are then computed analytically, allowing all teacher targets to jointly refine the interval trajectory without additional student backbone evaluations. This reuse concentrates the available trajectory information into each student update at low student-side cost. Table 10 shows that ArcFlow pairs one student forward pass with 2×4 teacher evaluations, providing four supervision states per interval; the retrained pi-Flow uses the same budget.

The momentum parameterization further makes the teacher's evolving dynamics easier to learn, enabling rapid alignment with only LoRA adapters and output heads. Thus, ArcFlow combines efficient reuse of multi-point teacher supervision with a lightweight, quickly converging adaptation process. This combination translates into the fewer training updates and lower total wall-clock time observed above.

F Additional Ablation Studies

We evaluate fixed and learnable momentum factors, parameter sharing across modes, the number of modes, and mixed trajectory integration. Each experiment uses the training settings stated below; its checkpoints are separate from those in Section E.2.1.

F.1 Momentum and Mixture Configuration

F.1.1 Fixed versus Learnable Momentum Factors

We compare $\gamma \equiv 1$, fixed momentum factors, and learnable momentum factors at $K = 16$ on Align5000 after 1,500 training steps. Table 11a reports FID values of 17.06, 14.77, and 14.56, respectively. With $\gamma \equiv 1$, the predicted velocity is constant within an inference interval. Distinct fixed momentum factors allow the relative contributions of basic velocities to change across timesteps, accounting for the improvement over constant velocity. Learning the factors additionally adapts these changes to each sample and produces the lowest FID. Figure 9(a) shows the corresponding differences in generated images.

Table 11: Momentum-factor and parameter-sharing ablations at $K = 16$ after 1,500 training steps. FID uses 50-step teacher images on Align5000 as the reference.

(a) Momentum-factor settings.

Momentum	FID ↓
$\gamma \equiv 1$	17.06
Fixed γ	14.77
Learnable γ	**14.56**

(b) Parameter sharing across modes.

(N_v, N_γ)	FID ↓
$(K, 1)$	15.08
$(1, K)$	14.97
(K, K)	**14.56**

Table 12: Effect of the number of momentum modes on teacher-reference FID and pFID on Align5000, with all other settings fixed.

K	FID ↓	pFID ↓
8	12.54	4.17
16	12.40	3.78
32	**12.39**	**3.69**

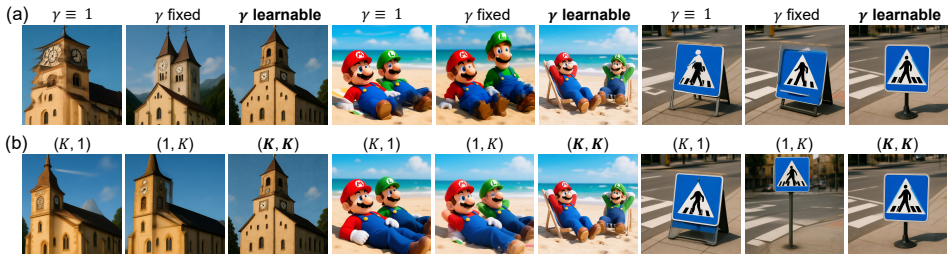

Figure 9: Qualitative ablations at $K = 16$ after 1,500 training steps. (a) Momentum-factor settings. (b) Numbers of independent basic velocities and momentum factors (N_v, N_γ).

F.1.2 Independent Velocity and Momentum Parameters

Let (N_v, N_γ) denote the numbers of independent basic velocities and momentum factors across the $K = 16$ mixture modes. We compare $(K, 1)$, which shares one momentum factor across distinct velocities, and $(1, K)$, which shares one basic velocity across distinct momentum factors, against the default (K, K). Each configuration is trained for 1,500 steps and evaluated using teacher-reference FID on Align5000. Table 11b and Figure 9(b) show that independently parameterizing both components yields the best result.

Sharing one momentum factor makes all basic velocities change by the same factor across timesteps, so their relative contributions cannot change within the interval. Sharing one basic velocity prevents different modes from representing different velocity directions. The default (K, K) configuration removes both restrictions, allowing mode-specific directions and factors to jointly approximate the teacher's changing velocity. Its lower FID supports learning these two components independently rather than requiring one component to compensate for restrictions on the other.

F.1.3 Scalability of the Number of Modes

We evaluate teacher-reference FID and pFID for $K \in \{8, 16, 32\}$ on Align5000 while keeping the other settings fixed. Table 12 shows that increasing K from 8 to 16 reduces FID from 12.54 to 12.40 and pFID from 4.17 to 3.78. Doubling K further to 32 yields smaller improvements, reaching FID 12.39 and pFID 3.69.

Increasing K expands the capacity to represent the teacher's velocity evolution through additional basic velocities and momentum factors. The small FID/pFID gains beyond $K = 16$ indicate diminishing returns from this extra capacity in the evaluated setting. We therefore use $K = 16$ to balance distributional fidelity with the parameter and computation cost of the prediction heads.

Table 13: Mixed-trajectory integration ablation after 3,000 updates. Lower teacher-reference FID indicates closer distributional alignment.

Student	Without mixed integration	With mixed integration
ArcFlow-Qwen	14.04	**13.52**
ArcFlow-FLUX	19.17	**18.21**

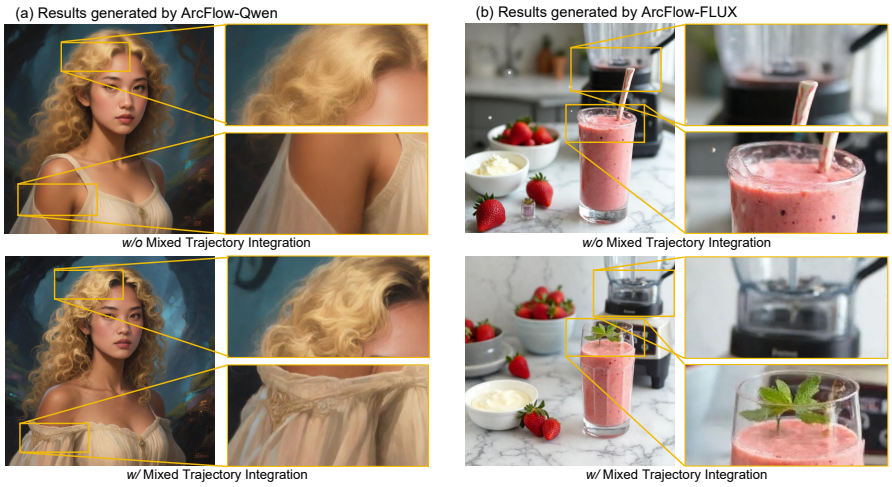

Figure 10: Qualitative mixed-integration ablation on (a) Qwen-Image-20B and (b) FLUX.1-dev, using the training protocol of Table 13.

F.2 Mixed Trajectory Integration

We train ArcFlow with and without mixed trajectory integration on Qwen-Image-20B and FLUX.1-dev. Each run uses 3,000 training steps and batch size 16, with teacher-reference FID evaluated on Align5000. The model parameterization is unchanged; this ablation tests the use of mixed trajectory integration during training, as defined in Section B.2.

Mixed integration improves FID on both backbones and produces richer local details and sharper structures in Figure 10. During early training, teacher-guided integration keeps the sampled latent states closer to the teacher trajectory, reducing the effect of inaccurate student velocity predictions on subsequent states. As λ increases, the student integrates using its own predicted velocities and learns from the states encountered during inference. The lower FID supports this training strategy in addition to the momentum-mixture parameterization itself.

G Additional Evaluation Results

We complement the main comparisons with a direct human preference study and controlled experiments with rCM and MeanFlow. We then report inference time, sensitivity to the inference timestep schedule, generation at different resolutions and aspect ratios, and additional comparisons with Qwen-Image-Lightning.

G.1 Human Preference Study

Teacher-reference FID measures fidelity to the teacher distribution, whereas learned preference scores, such as HPSv2, provide automated estimates rather than direct human judgments. We therefore conduct a pairwise human preference study to test whether viewers prefer ArcFlow outputs over those of competing few-step generators. This evaluation complements the five benchmark metrics in Table 1 and distinguishes perceived generation quality from agreement with teacher-generated references.

Table 14: Pairwise human preferences on 200 held-out prompts using the Qwen backbone at 2 NFE for all methods. Percentages include ties and sum to 100% for each comparison.

Baseline	ArcFlow preferred	Baseline preferred	Tie
TwinFlow	**65.05%**	26.15%	8.8%
Qwen-Image-Lightning	**52.15%**	33%	14.85%

Table 15: Controlled comparison on Qwen-Image-20B after 3,000 updates. FID uses 50-step teacher outputs on Align5000 as the reference; lower is better.

Method	NFE	Teacher-reference FID ↓
rCM	2	62.93
ArcFlow	2	**13.52**

We randomly sample 200 prompts from a held-out set and generate images with ArcFlow, TwinFlow, and Qwen-Image-Lightning, all using the Qwen backbone and evaluated at 2 NFE. We compare ArcFlow separately with each baseline. For each prompt, the corresponding image pair is displayed side by side in randomized left–right order. Twenty human annotators are asked to indicate their preference based on overall quality, visual fidelity, and prompt alignment. We report the proportions of ArcFlow preferences, baseline preferences, and ties, retaining ties in the denominator.

Table 14 summarizes the results. Against TwinFlow, ArcFlow is preferred in 65.05% of judgments, compared with 26.15% for TwinFlow and 8.8% ties. Against Qwen-Image-Lightning, ArcFlow receives 52.15% preference, compared with 33% for the baseline and 14.85% ties. ArcFlow thus receives a majority preference in both comparisons even when ties are included, with the larger margin observed against TwinFlow.

These direct human judgments support ArcFlow's perceptual advantage at 2 NFE. Together with the COCO real-image FID results in Table 2, they provide complementary evidence for improved generation quality beyond teacher-reference fidelity.

G.2 Controlled Comparison with rCM and MeanFlow

To extend the controlled evidence beyond pi-Flow and TwinFlow in Section E.3, we examine rCM (Zheng et al., 2025) and the original MeanFlow (Geng et al., 2025) objective. These experiments broaden coverage of consistency- and MeanFlow-related approaches while separating training-protocol effects from algorithmic effects. Unlike the external reference comparisons in Table 1, the rCM experiment matches the backbone, data, inference budget, and training horizon to ArcFlow's ablation setting.

Controlled rCM protocol. We adapt rCM's publicly available implementation to Qwen-Image and retrain it using the same training data, training protocol, and 3,000-update budget as ArcFlow. Both methods use the Qwen-Image-20B backbone and are evaluated at 2 NFE on Align5000, while retaining their respective algorithmic objectives. Following Sections C.2 and E.3, we compute teacher-reference FID against 50-step teacher outputs, rather than real images. The ArcFlow checkpoint is the 3,000-update ablation checkpoint also reported in Table 8, not the final 7,500-step checkpoint.

Results at the fixed budget. As shown in Table 15, ArcFlow achieves a teacher-reference FID of 13.52, compared with 62.93 for rCM. This substantially smaller discrepancy indicates stronger alignment with the teacher distribution at the same training horizon, supporting ArcFlow's learning efficiency without attributing the difference to a larger backbone, more training data, or additional updates. This fixed-budget comparison measures progress after 3,000 updates, rather than establishing the relative performance of fully converged models.

Separate MeanFlow adaptation. We separately adapt the original MeanFlow objective to FLUX under the same 2-NFE setting and 3,000-update budget as ArcFlow. After 3,000 updates, this run still

Table 16: Mean inference time over 100 runs at 1024×1024 and 2 NFEs. Comparisons are grouped by backbone.

Method	Inference time (s)
Qwen-Image-20B students	
Qwen-Image-Lightning	1.693
TwinFlow	1.370
pi-Flow (GM-Qwen)	1.451
ArcFlow-Qwen	1.408
FLUX.1-dev students	
SenseFlow	1.435
pi-Flow (GM-FLUX)	1.465
ArcFlow-FLUX	1.459

Table 17: Sensitivity to the 2-step inference schedule on Align5000.

Schedule	FID ↓
$1.0 \to 0.76 \to 0$	12.40
$1.0 \to 0.60 \to 0$	12.52
$1.0 \to 0.50 \to 0$	12.48
$1.0 \to 0.80 \to 0$	12.65

produces mostly blurry color blobs and noise, without stable, coherent image structure. We therefore report this outcome qualitatively and do not include a quantitative FID/pFID comparison for this run. The observation concerns this direct adaptation within the tested budget, rather than an intrinsic limitation of MeanFlow.

G.3 Inference Latency

We measure inference time at 1024×1024 resolution and 2 NFEs using the same prompt, and average over 100 runs per model. As shown in Table 16, ArcFlow takes 1.408 seconds on Qwen and 1.459 seconds on FLUX, compared with 1.451 and 1.465 seconds for pi-Flow. These comparable inference times are consistent with the small prediction-head FLOPs increase in Section D.3. The momentum-mixture representation models velocity changes across timesteps while its analytic transition avoids additional backbone evaluations within each inference interval.

G.4 Robustness and Sensitivity Analysis

G.4.1 Guidance-free Inference

During distillation, ArcFlow learns to match teacher velocities computed with CFG (e.g., $w = 3.5$). The student therefore predicts the guided velocity directly and does not require an additional classifier-free guidance evaluation at inference.

G.4.2 Flexibility of Timestep Schedules

We change the intermediate timestep for 2-NFE generation on Align5000 without retraining the model. Table 17 reports teacher-reference FID. FID varies from 12.40 to 12.65 across the tested schedules, indicating that teacher-distribution alignment changes little within this range of intermediate timesteps.

G.4.3 Resolution and Aspect-Ratio Generalization

We also test generation at higher resolutions (e.g., 2048×2048) and different aspect ratios (e.g., 16:9 and 9:16) without additional training. The generated samples retain global structure and local detail, suggesting that the momentum-mixture parameterization is not restricted to the training resolution or aspect ratio. Additional generated samples are provided in Section L.

G.5 Discussion of Qwen-Image-Lightning

Although Qwen-Image-Lightning achieves competitive performance on prompt-alignment benchmarks, its inferior FID and pFID indicate clear degradation in generation quality at the distribution level. Its higher teacher-reference FID and pFID, together with the zoomed-in comparisons in Figure 4, reflect distorted local statistics and weakened high-frequency details, such as blurred textures and over-smoothed structures, which are not captured by prompt-alignment metrics that primarily emphasize semantic correctness and global visual saliency. As a result, although Qwen-Image-Lightning can generate images that remain semantically consistent with the prompt, it deviates noticeably from the teacher's generation distribution in fine-grained structure and detail.

Moreover, this discrepancy reveals a fundamental difference in distillation efficiency and objective. Qwen-Image-Lightning focuses on achieving perceptual and semantic alignment under limited steps, but does not explicitly enforce high-precision alignment with the teacher's underlying generation trajectory, leading to a partial loss of the teacher's original generative prior. As shown in Figure 12, Qwen-Image-Lightning exhibits unstable performance in our test cases (especially in the first column), proving our assumption that it sacrifices the overall distribution correctness for higher semantic alignment. In contrast, ArcFlow directly distills the teacher's non-linear velocity field and preserves its trajectory non-linearity through momentum-based parameterization. This design allows ArcFlow to rapidly converge to a high-precision teacher alignment with minimal training cost, thereby maintaining both strong prompt-following ability and high distributional fidelity in the few-step regime.

H Discussion on comparison with pi-Flow

ArcFlow is motivated by a dynamical view of few-step generation: the continuous trajectory over a long timestep interval should be the object of distillation, with its velocity evolution and accumulated displacement governed by the same learned dynamics. Denoising velocities are correlated across timesteps, yet their directions and magnitudes evolve as image structure and detail emerge. The student therefore needs more than an accurate tangent at the interval start. It should describe a coherent, non-linear curve whose tangents at different timesteps are coupled, whose displacement can be evaluated directly, and whose shape can be refined jointly by teacher observations throughout the interval.

We realize this motivation through momentum parameterization: multiple velocity modes evolve at different rates, and their superposition defines the whole interval from a single backbone prediction. The same momentum modes determine both the continuous velocity curve and its analytic integral. Teacher observations across timesteps jointly update these modes, refining the trajectory used for generation. This connects what ArcFlow predicts, how it learns from the teacher, and how it generates images. We discuss the differences from pi-Flow in inference and teacher supervision below, followed by an analysis of the experimental results.

H.1 Analytic Integration

At the starting state $(\mathbf{x}_{t_s}, t_s)$, ArcFlow predicts momentum modes $\Theta = \{\pi_k, \mathbf{v}_k, \gamma_k\}_{k=1}^{K}$ that describe the continuous velocity evolution over the interval (Equation (2)). Their relative contributions change with timesteps, allowing the resulting trajectory to curve even though Θ remains fixed. The same modes also determine the analytic transition Φ in Equation (4). Both intermediate velocities and displacements over arbitrary sub-intervals can therefore be evaluated directly, without numerical substeps or additional student backbone evaluations.

This analytic structure is used throughout the method. During training, Φ constructs the student portions of mixed teacher–student trajectories (Section B.2); at inference, it computes the complete interval transition as a single weighted sum (Algorithm 2). In comparison, pi-Flow's Gaussian-mixture variant (Chen et al., 2025a), evaluated in our experiments, predicts a lightweight state-dependent velocity policy and traverses the interval through numerical substeps. These policy evaluations are inexpensive relative to the backbone, but the latent transition is obtained by successively executing the policy. The operational distinction is therefore a numerical policy rollout versus a direct analytic interval transition: ArcFlow's prediction specifies not only the velocity evolution but also the explicit map used to move between latent states.

H.2 Teacher Supervision of the Continuous Trajectory

We use teacher velocities at sampled timesteps to supervise the predicted continuous trajectory. At sampled rollout states $(\mathbf{x}_{t_i}, t_i)$, the targets $\mathbf{u}(\mathbf{x}_{t_i}, t_i)$ jointly supervise the same momentum modes through Equation (6). Because these modes determine both the continuous velocity curve and its integral, a teacher observation supplies gradients to parameters that control the trajectory throughout the interval, rather than affecting only the velocity at the observed timestep.

This coupling can be seen directly from the momentum formulation. Write $\mathbf{v}(t;\Theta)$ for the velocity induced by the fixed interval-start prediction. Holding the other predicted parameters fixed, a change $\delta\mathbf{v}_k$ in the k-th basic velocity gives

$$\begin{aligned} \delta\mathbf{v}(t;\Theta) &= \pi_k \gamma_k^{1-t}\, \delta\mathbf{v}_k, \\ \delta\Phi(\mathbf{x}_{t_s}, t_s, t_e;\Theta) &= \pi_k \mathcal{C}(\gamma_k, t_s, t_e)\, \delta\mathbf{v}_k, \end{aligned} \tag{24}$$

where $\mathcal{C}$ is defined in Equation (5). Thus, an update driven by a tangent error at one sampled time reshapes velocities across the interval and changes their accumulated displacement. Updates to the gating weights and momentum factors similarly adjust the mode contributions and their evolution across timesteps. Multiple teacher observations consequently act together to refine one continuous trajectory, with the same fitted dynamics determining the transition used at inference.

In the training computation, the sampled rollout states are detached before velocity matching (Section 3.3 and Algorithm 1). Gradients propagate through the analytic velocity expression directly to Θ and the network predicting it, without back-propagating through a chain of latent-state updates. The resulting optimization problem is a joint fit of continuous interval dynamics: teacher constraints update the modes governing the whole curve and its displacement (Equation (24)). Thus, one student backbone prediction supports multi-timestep supervision of the same interval transition that will be executed at inference.

H.3 Experimental Evidence

To assess whether ArcFlow provides a more effective distillation method, we test its generation quality and learning efficiency. To test whether analytic integration improves generation quality, we use the same trained ArcFlow checkpoint and compare analytic and numerical integration at inference. To test learning efficiency, we train ArcFlow and pi-Flow with the same training and teacher-supervision budgets and compare their convergence.

Analytic versus numerical integration. ArcFlow's momentum parameterization allows analytic integration of the predicted velocity field. To test whether this improves generation quality, we use the same trained ArcFlow checkpoint for inference with either analytic or numerical integration (Table 18). The numerical version uses 64 substeps per interval. Both versions use the same model weights, backbone NFEs, timestep schedule, and evaluation protocol. No retraining is performed.

Both versions use the same trained model, so numerical integration does not change the learned velocity field or add modeling capacity. It introduces an approximation when computing the latent displacement. The lower FID with analytic integration shows that avoiding this approximation improves generation quality for the same learned dynamics. This advantage comes from ArcFlow's momentum parameterization, which makes the predicted velocities analytically integrable. In comparison, pi-Flow's state-dependent Gaussian-mixture policy does not provide the corresponding closed-form interval transition. ArcFlow therefore uses the same learned momentum modes both to fit the teacher's continuous velocity evolution and to compute latent displacements without numerical substeps.

Implications for learning and generation. The matched-budget 2-NFE study in Section E.3 evaluates how efficiently ArcFlow learns the teacher's trajectories. The faster convergence supports our motivation for using momentum parameterization: teacher velocities at different timesteps jointly refine the same continuous trajectory, making the teacher's velocity evolution easier to learn. The complementary 4-NFE comparisons in Tables 1 and 2 show that the resulting model remains competitive at pi-Flow's established inference budget, supporting the usefulness of this trajectory formulation beyond a reduced-step evaluation.

Table 18: Inference-only integration ablation of a single trained ArcFlow checkpoint, without retraining. Only the integration rule changes; model weights, backbone NFEs, the timestep schedule, and the evaluation protocol are fixed. FID uses teacher generations as the reference on Align5000.

Integration scheme	Numerical substeps per interval	FID ↓
Analytic	–	**12.40**
Numerical	64	12.93



Figure 11: A representative failure of ArcFlow: 1-NFE inference produces blurry results.

Table 19: Per-trajectory oracle fitting of teacher velocities.

Interval	K	Oracle velocity MSE ↓
Full interval $[0, 1]$	16	0.01257
Full interval $[0, 1]$	32	0.01101
Half interval (2-NFE reference)	16	0.00450

I Limitations and Future Work

I.1 Diagnosing 1-NFE Generation

In 1-NFE generation, ArcFlow must predict all trajectory parameters $\{\pi_k, \mathbf{v}_k, \gamma_k\}_{k=1}^K$ from pure noise at $t = 1$ and use them over the entire interval $[0, 1]$. Unlike the 2-NFE schedule $1 \to 0.76 \to 0$ (Section G.4), it receives no second state-conditioned evaluation at $t = 0.76$, where the latent already contains meaningful signal. This extreme setting can severely degrade generation quality and fail to produce meaningful structure, as the blurry example in Figure 11 illustrates. In particular, the momentum factors γ_k become highly sensitive and challenging to estimate when one prediction must govern the full trajectory.

To distinguish limitations of the parameterization from difficulties in learning its parameters, we examine three diagnostics: per-trajectory oracle fitting, increased trainable capacity, and explicit endpoint supervision. Oracle velocity MSE measures fitting residuals, not generation quality. Generation FID follows the Align5000 teacher-reference convention in Section C.2 and measures alignment with the teacher's image distribution.

I.1.1 Parameterization: Per-trajectory Oracle Fitting

First, we ask whether the momentum mixture in Equation (2) can approximate teacher velocities over a full interval when network prediction is removed. For each of 1,000 teacher trajectories, we directly optimize a free parameter set $\{\pi_k, \mathbf{v}_k, \gamma_k\}_{k=1}^K$ to match teacher velocities at 50 dense timesteps along that trajectory. These are per-trajectory optimization variables, not network predictions. We compare full-interval fits at $K = 16$ and $K = 32$ with a half-interval, $K = 16$ fit serving as a 2-NFE reference.

Table 19 shows that the full-interval mixture can fit teacher velocities with a relatively small residual, with only a modest improvement from doubling K. The substantially lower half-interval residual also shows that covering the entire trajectory is a more demanding approximation task. These results

Table 20: Trainable-capacity ablation for 1-NFE generation after 5,000 training steps.

Training mode	1-NFE FID ↓
LoRA + heads (default)	39.34
Full-parameter fine-tuning	**33.81**

Table 21: Objective ablation for 1-NFE generation.

Objective	1-NFE FID ↓
Velocity matching only	39.34
Velocity matching + endpoint loss	**35.27**

distinguish the ability to fit trajectory-specific parameters from the network's ability to predict them from pure noise in one evaluation, motivating the following capacity and supervision diagnostics.

I.1.2 Trainable Capacity: Full-parameter Fine-tuning

Next, we test whether increasing trainable capacity alleviates this prediction burden. The default configuration updates only LoRA adapters and output heads, comprising less than 5% of parameters. We compare it with full-parameter fine-tuning under the same training schedule for 5,000 training steps. As Table 20 shows, full-parameter training lowers 1-NFE FID from 39.34 to 33.81, indicating a partial benefit from relaxing the trainable-capacity constraint.

A separate preliminary ablation also showed noticeable qualitative improvements when the original linear prediction head was replaced by a linear–ReLU–linear MLP, with the training setting otherwise unchanged. This head-level observation and the full-parameter comparison both support the importance of predictive capacity.

I.1.3 Objective: Explicit Endpoint Supervision

Finally, we test whether supervision matches the full-interval generation task. The velocity-matching objective in Equation (6) supervises instantaneous velocities at intermediate timesteps but does not explicitly constrain the terminal latent. Errors in the predicted velocity evolution can accumulate in the integrated transition without a corrective second evaluation. We therefore augment velocity matching with

$$\mathcal{L}_{\text{end}} = \left\| \mathbf{x}_0^{\text{arc}} - \mathbf{x}_0^{\text{tea}} \right\|_2^2, \tag{25}$$

where $\mathbf{x}_0^{\text{arc}}$ is ArcFlow's analytically computed terminal state using Equation (4), and $\mathbf{x}_0^{\text{tea}}$ is the teacher's terminal state.

Velocity supervision encourages local agreement, whereas the endpoint term directly supervises the result of integrating the full predicted field. This distinction matters because the terminal latent determines the generated output, while no intermediate network evaluation is available to revise the one-step prediction.

In this separate objective ablation, adding endpoint supervision improves 1-NFE FID from 39.34 to 35.27 (Table 21). This reduction supports a complementary role for direct terminal-state supervision, but does not remove the 1-NFE limitation.

Practical implications and future work. Together, these diagnostics identify predictive learnability and supervision as important contributors to 1-NFE degradation. Increased trainable capacity and endpoint supervision each provide partial improvements in separate experiments. In practice, 2-NFE inference remains a favorable trade-off between modeling difficulty and generation quality. Jointly improving prediction architectures and objective design, including more effective estimation of sensitive momentum factors, is a promising direction for closing the remaining 1-NFE gap.

J Human Subjects Data Concern

All training and testing datasets used in this work are publicly available datasets and benchmarks. These datasets contain some images of human subjects but were collected and released by their original creators under specific open-source licenses for research purposes. We have strictly adhered to the terms of use and licensing agreements for each dataset.

K Ethics Concerns

This paper aims to advance the efficiency of image generation models by enabling high-quality few-step inference through improved distillation techniques. Such progress can facilitate broader accessibility and deployment of generative models in practical applications, including creative tools, simulation, and content generation.

At the same time, as with prior work on image generation, our method could potentially be misused for generating misleading or harmful visual content. These concerns are not unique to our approach and are inherent to the broader class of generative image models. We emphasize that responsible deployment, including appropriate content moderation, usage policies, and the development of reliable AI-generated content detection mechanisms, remains important.

We believe that the technical contributions of this work primarily improve inference efficiency and fidelity, without introducing new ethical risks beyond those already present in existing diffusion-based image generation systems.

L Additional Qualitative Results

L.1 Comparison of Few-step Students on Qwen-Image-20B

Figure 12 compares student models distilled from Qwen-Image-20B using the same initial noise. The differences between ArcFlow and Qwen-Image-Lightning are discussed in Section G.5.

L.2 Comparison of Few-step Students on FLUX.1-dev

Figure 13 provides the corresponding qualitative comparison on FLUX.1-dev.

L.3 More High-Resolution Visualizations

In this section, we present additional qualitative examples generated by ArcFlow to further demonstrate its generative performance. All prompts are randomly sampled, and the results are shown directly without any manual selection or filtering. Visualization results are shown in Figure 14, Figure 15, Figure 16.

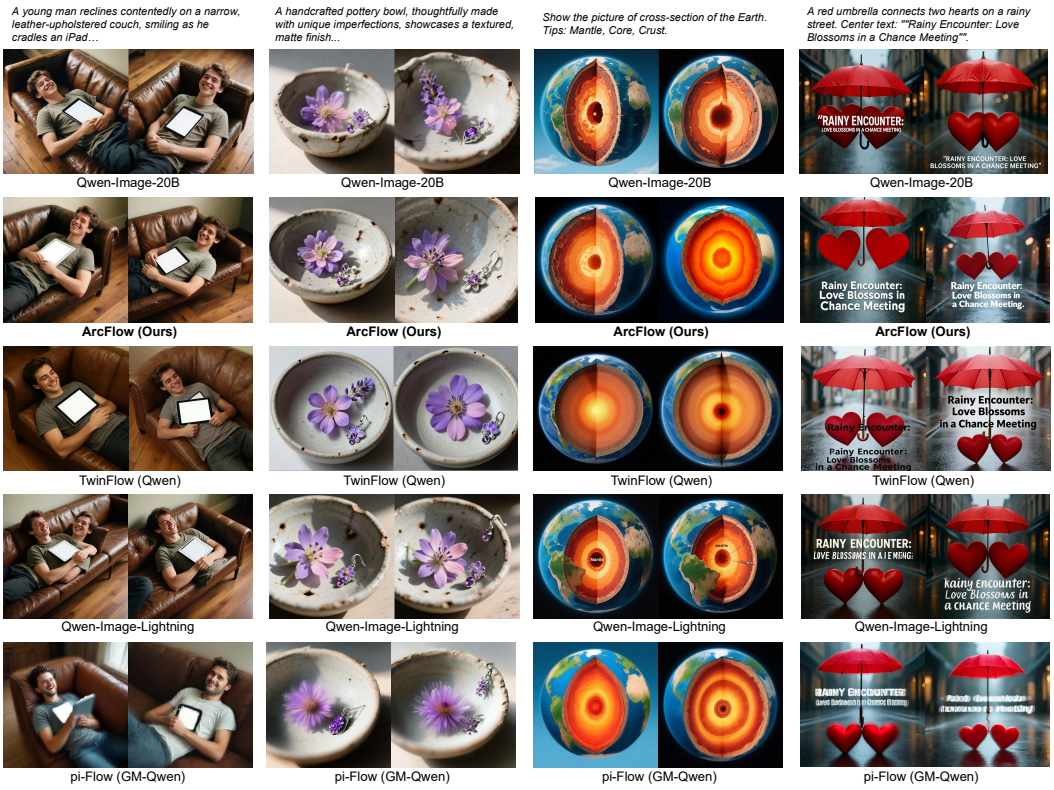

Figure 12: Additional qualitative comparisons between different student models distilled on Qwen-Image-20B. Note that results in each column are generated from the same batch of initial noise.

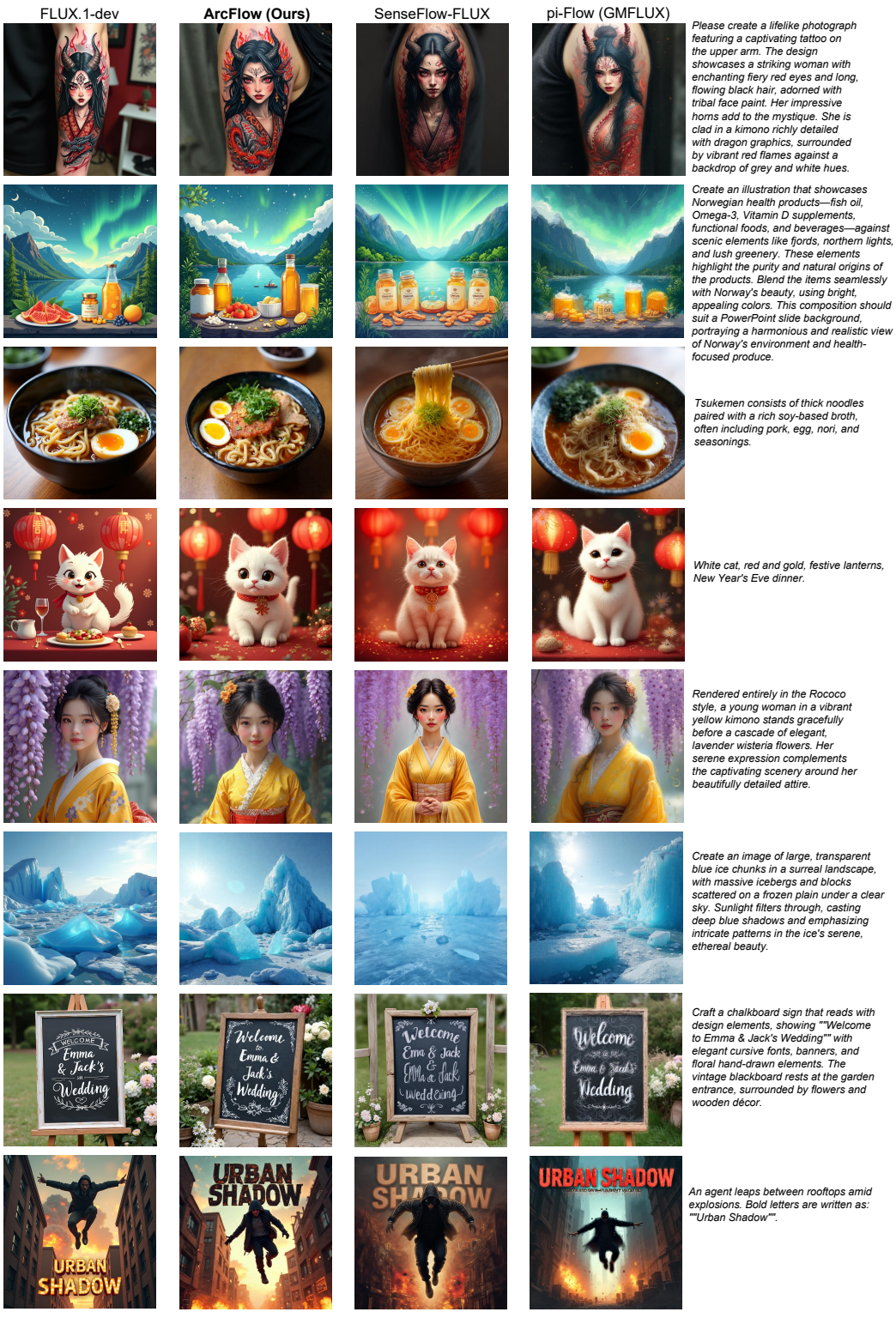

Figure 13: Qualitative comparisons between different student models distilled on FLUX.1-dev.

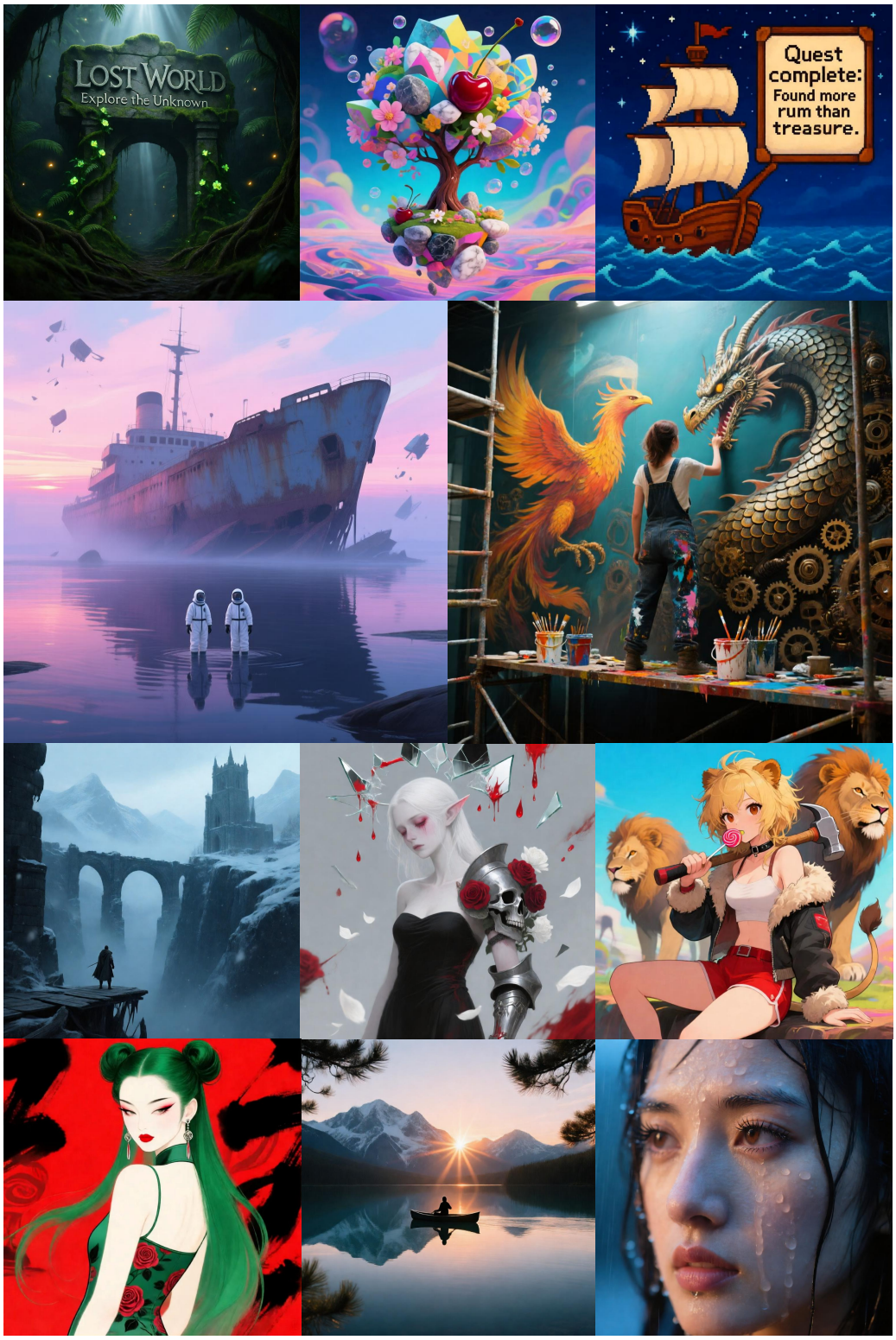

Figure 14: Visualization of ArcFlow-Qwen (NFE=2). Each image is of 1024×1024 resolution.



Figure 15: Visualization of ArcFlow-Qwen (NFE=2). Each image is of 1024×1024 resolution.



Figure 16: Visualization of ArcFlow-FLUX (NFE=2). Each image is of 1024×1024 resolution.