FLOW-FACTORY: A UNIFIED FRAMEWORK FOR REINFORCEMENT LEARNING IN FLOW-MATCHING MODELS

Bowen Ping[1], Chengyou Jia[1], Minnan Luo[1], Hangwei Qian[2], Ivor Tsang[2]
[1]Xi'an Jiaotong University [2]CFAR and IHPC, A*STAR
jayceping6@gmail.com

ABSTRACT

Reinforcement learning has emerged as a promising paradigm for aligning diffusion and flow-matching models with human preferences, yet practitioners face fragmented codebases, model-specific implementations, and engineering complexity. We introduce **Flow-Factory**, a unified framework that decouples algorithms, models, and rewards through through a modular, registry-based architecture. This design enables seamless integration of new algorithms and architectures, as demonstrated by our support for GRPO, DiffusionNFT, and AWM across Flux, Qwen-Image, and WAN video models. By minimizing implementation overhead, Flow-Factory empowers researchers to rapidly prototype and scale future innovations with ease. Flow-Factory provides production-ready memory optimization, flexible multi-reward training, and seamless distributed training support. The codebase is available at https://github.com/X-GenGroup/Flow-Factory.

1 INTRODUCTION

Recent advances in diffusion and flow-matching models have revolutionized generative modeling across images, videos, and multimodal content (Song et al., 2020; Lipman et al., 2022; Liu et al., 2022). These models have demonstrated remarkable sample quality and training efficiency by learning continuous-time velocity fields that transport noise distributions to data distributions. However, aligning these powerful generative models with human preferences and real-world constraints remains a fundamental challenge.

Reinforcement learning (RL) has emerged as a promising paradigm for post-training alignment, enabling models to optimize arbitrary reward signals beyond maximum likelihood training (Black et al., 2024; Liu et al., 2025; Xue et al., 2025b). Recent works such as Flow-GRPO (Liu et al., 2025) and DanceGRPO (Xue et al., 2025b) have demonstrated the effectiveness of policy gradient methods for flow-matching models by introducing stochastic differential equation (SDE) formulations that enable RL exploration. Subsequent research has further improved training efficiency through mixed ODE-SDE sampling (Li et al., 2025; He et al., 2025), addressed reward hacking via loss reweighting (Wang et al., 2025a), and explored alternative objectives that align RL with diffusion pretraining (Zheng et al., 2025; Xue et al., 2025a).

Despite this rapid algorithmic progress, practitioners face significant barriers when applying RL to flow-matching models:

Fragmented and Coupled Codebases. As RL algorithms proliferate with their own SDE formulations and training procedures, implementations have become increasingly fragmented across isolated repositories. Moreover, existing codebases frequently couple algorithmic logic with model-specific implementations, making it difficult to transfer algorithms across architectures without extensive rewriting, or to evaluate multiple algorithms on the same model under controlled conditions.

Training Inefficiency and Memory Bottlenecks. RL fine-tuning of flow-matching models requires repeated forward passes for trajectory sampling and reward computation, leading to signif-

icant training overhead. Additionally, large-scale models with multiple frozen components (text encoders, VAEs) consume substantial GPU memory even when only the transformer backbone is trainable, limiting batch sizes and hindering scalability on commodity hardware.

Limited Reward Model Flexibility. Current frameworks typically support only pointwise reward models that score individual samples independently. However, emerging algorithms increasingly rely on groupwise rewards that evaluate relative rankings within sample groups (Wang et al., 2025b), or require combining multiple rewards with different aggregation strategies (Liu et al., 2026). The lack of unified reward interfaces restricts algorithmic exploration and multi-objective optimization.

We present **Flow-Factory**, a unified and extensible framework addressing these challenges through principled software design. Our key contributions include:

- **Registry-based modular architecture.** Flow-Factory decouples models, algorithms, rewards, and schedulers via a unified plug-and-play registry. This modular design minimizes implementation overhead, allowing researchers to integrate new architectures or algorithms with minimal code changes. Any supported component can be cross-combined through configuration alone, facilitating principled benchmarking and rapid prototyping.
- **Preprocessing-based memory optimization.** Flow-Factory preprocesses and caches condition embeddings before training, enabling frozen components (text encoders, VAEs) to be offloaded from GPU memory. During training, only the target transformer resides on GPU, dramatically reducing memory usage while improving throughput by eliminating redundant encoding operations.
- **Flexible multi-reward system.** Flow-Factory provides unified interfaces for both *pointwise* and *groupwise* reward models, with automatic model deduplication and configurable advantage aggregation strategies. This flexibility supports diverse algorithmic requirements and enables principled multi-reward training with weighted combinations.

2 Framework Design

Figure 1 illustrates the overall architecture of Flow-Factory, which consists of three core design principles: registry-based component decoupling, preprocessing-based memory optimization, and a flexible multi-reward system.

2.1 Registry-Based Component Decoupling

Existing RL fine-tuning codebases typically interleave model-specific logic with algorithmic implementations, forcing researchers to reimplement algorithms when switching architectures. To address this, Flow-Factory introduces a registry-based architecture with four decoupled component types: `BaseAdapter` for model operations (encoding, forward passes, checkpointing), `BaseTrainer` for algorithm logic (sampling, advantage computation, optimization), `BaseRewardModel` for reward computation, and `SDESchedulerMixin` for stochastic sampling with log-probability computation.

By adhering to these standardized interfaces, integrating a new model or algorithm merely requires implementing the specific abstract methods of the corresponding base class. This design reduces the integration complexity from $O(M \times N)$ to $O(M + N)$ for M models and N algorithms. Consequently, components are registered via a global system and instantiated through YAML configurations (Figure 1, top), enabling seamless cross-architectural inheritance and allowing practitioners to extend the framework to future innovations with negligible engineering overhead. This modularity ensures that Flow-Factory remains future-proof as new foundation models and reinforcement learning paradigms emerge.

2.2 Preprocessing-Based Memory Optimization

Modern flow-matching pipelines contain large frozen components, such as text encoders, image encoders, and VAEs, which consume substantial GPU memory even when only the transformer is trainable, while RL fine-tuning repeatedly encodes the same prompts across iterations. Flow-Factory addresses both issues through a two-phase approach: before training, we precompute and cache

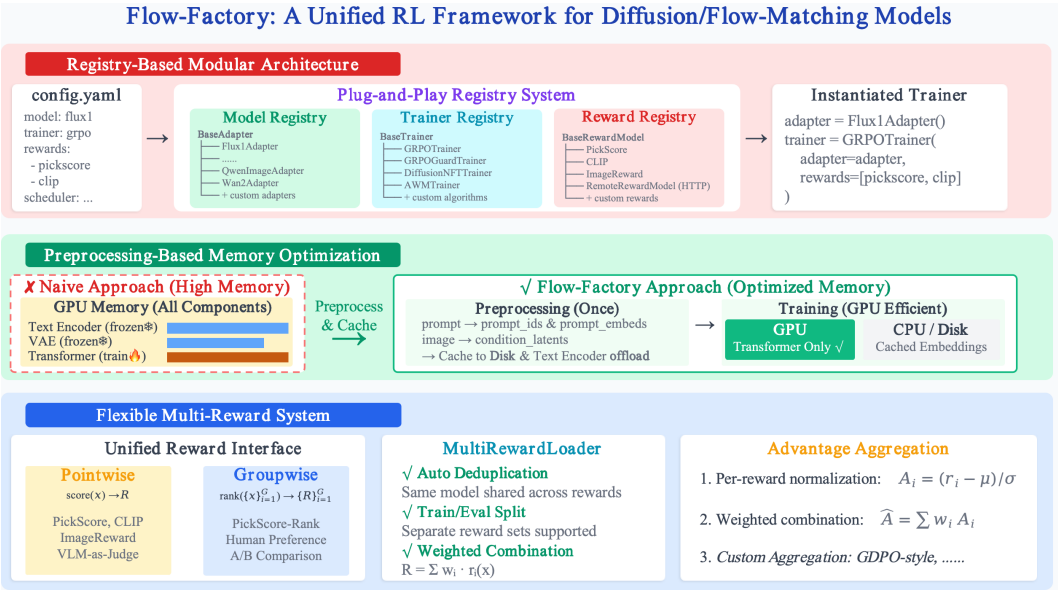

Figure 1: Flow-Factory architecture overview. **Top:** The registry-based design decouples Models, Trainers, and Rewards, enabling flexible combinations through YAML configuration. **Middle:** Preprocessing optimizes memory by caching embeddings and offloading frozen components. **Bottom:** The multi-reward system supports both pointwise and groupwise rewards with automatic deduplication and configurable advantage aggregation.

all condition embeddings (prompt embeddings, pooled embeddings, VAE latents) to disk; during training, these cached embeddings are loaded directly, allowing frozen components to be completely offloaded. As shown in Figure 1 (middle), this transforms the memory layout to *transformer-only on GPU*,, enabling larger batch sizes while eliminating redundant encoding operations to improve throughput.

2.3 Multi-Reward System

Existing frameworks typically support only pointwise rewards that score samples independently, yet emerging algorithms such as Pref-GRPO (Wang et al., 2025b) require groupwise rewards for ranking-based evaluation or multi-objective optimization combining heterogeneous signals, which often results redundant model loading when configurations share the same backbone. Flow-Factory provides a unified reward system (Figure 1, bottom) with three mechanisms: (1) *unified interfaces* for both `PointwiseRewardModel` (score$(x) \rightarrow \mathbb{R}$) and `GroupwiseRewardModel` (rank$(\{x_i\}) \rightarrow \mathbb{R}^k$); (2) *automatic deduplication* via `MultiRewardLoader` that loads each unique model once regardless of how many reward configurations reference it; (3) *configurable advantage aggregation* supporting both weighted-sum and GDPO-style (Liu et al., 2026) per-reward normalization. Implementing new aggregation strategies only requires a new `compute_advantages` method in the trainer class. This design supports diverse algorithmic requirements while minimizing memory overhead and simplifying custom reward integration.

3 Supported Algorithms

Flow-Factory provides unified implementations of state-of-the-art RL algorithms for flow-matching models. All algorithms share the same model adapter and reward interfaces, enabling direct comparison under controlled conditions.

3.1 FLOW-GRPO AND VARIANTS

Flow-GRPO (Liu et al., 2025) extends the deterministic ODE sampling of flow-matching models with stochastic noise injection to enable RL exploration. The standard ODE update $x_{t+dt} = x_t + v_\theta(x_t, t)dt$ is replaced by an SDE formulation:

$$\boldsymbol{x}_{t+dt} = \boldsymbol{x}_t + \left[\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) + \frac{\sigma_t^2}{2t}(\boldsymbol{x}_t + (1-t)\boldsymbol{v}_\theta(\boldsymbol{x}_t, t))\right] dt + \sigma_t\sqrt{dt}\,\epsilon \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, I)$ and σ_t controls noise injection. This formulation enables log-probability computation required for policy gradient optimization.

Flow-Factory implements multiple SDE dynamics through a unified `SDESchedulerMixin` interface, as summarized in Table 1. Users can switch between formulations via a single configuration parameter, facilitating systematic comparison of their effects on training stability and sample quality.

Table 1: Supported SDE dynamics in Flow-Factory.

Dynamics	**Noise Schedule** σ_t	**Reference**
Flow-SDE	$\eta\sqrt{t/(1-t)}$	Flow-GRPO (Liu et al., 2025)
Dance-SDE	η (constant)	DanceGRPO (Xue et al., 2025b)
CPS	$\sigma_{t-1}\sin(\eta\pi/2)$	FlowCPS (Wang and Yu, 2025)
ODE	0 (deterministic)	For NFT (Zheng et al., 2025)/AWM (Xue et al., 2025a)

Beyond vanilla GRPO, Flow-Factory supports two important variants. **MixGRPO** (Li et al., 2025)(i.e., *Flow-GRPO-Fast*) applies SDE updates to only 1–2 selected timesteps while using ODE for the rest, significantly reducing computational cost while maintaining performance. **GRPO-Guard** (Wang et al., 2025a) addresses the negatively-biased ratio distribution inherent in SDE formulations through timestep-dependent loss reweighting, mitigating reward hacking and stabilizing training.

3.2 DIFFUSIONNFT AND AWM

While GRPO couples sampling dynamics with training timesteps, alternative paradigms exist that decouple them entirely.

DiffusionNFT (Zheng et al., 2025) optimizes a contrastive objective directly on the forward flow-matching process:

$$\mathcal{L}_{\text{NFT}} = \mathbb{E}_{\boldsymbol{c},t}\left[r||\boldsymbol{v}_\theta^+(\boldsymbol{x}_t, \boldsymbol{c}, t) - \boldsymbol{v}||_2^2 + (1-r)||\boldsymbol{v}_\theta^-(\boldsymbol{x}_t, \boldsymbol{c}, t) - \boldsymbol{v}||_2^2\right] \tag{2}$$

where $\boldsymbol{v}_\theta^+$ and $\boldsymbol{v}_\theta^-$ represent implicit positive and negative policies weighted by a reward $r \in [0, 1]$. By contrasting these implicit distributions, DiffusionNFT identifies a policy improvement direction without the need for tractable likelihood estimation or SDE-based sampling.

Advantage Weighted Matching (AWM) (Xue et al., 2025a) further aligns RL optimization with the flow-matching pretraining objective by weighting the standard velocity matching loss with per-sample advantages:

$$\mathcal{L}_{\text{AWM}} = \mathbb{E}_{t,\epsilon}\left[A(\boldsymbol{x}_0) \cdot \|\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - (\epsilon - \boldsymbol{x}_0)\|^2\right] \tag{3}$$

where $A(\boldsymbol{x}_0)$ denotes the advantage computed from reward signals. This formulation incorporates reward-based guidance directly into the velocity matching loss, effectively aligning the optimization target with the original flow-matching objective.

Since both algorithms decouple training from sampling dynamics, they are inherently solver-agnostic: any ODE solver can be used for trajectory generation without modifying the training procedure. Flow-Factory exploits this property by providing configurable timestep sampling strategies (uniform, logit-normal, or discrete) and support for high-order ODE solvers, enabling practitioners to balance computation cost against sample quality during data collection.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate Flow-Factory using FLUX.1-dev (Labs, 2026) with three algorithms: Flow-GRPO (Liu et al., 2025), DiffusionNFT (Zheng et al., 2025), and AWM (Xue et al., 2025a). The reward models are chosen as Text-Rendering and PickScore (Kirstain et al., 2023), following prior works.

4.2 REPRODUCTION OF PUBLISHED RESULTS

To verify the correctness of our implementation and the effectiveness of the decoupled architecture, we replicate the training processes of three state-of-the-art RL algorithms including Flow-GRPO (Liu et al., 2025), DiffusionNFT (Zheng et al., 2025), and AWM (Xue et al., 2025a) using the FLUX.1-dev model as the common backbone.

As shown in Figure 2, Flow-Factory successfully recovers the performance gains reported in the original literature. All three algorithms demonstrate consistent reward growth when optimized against the PickScore reward model, achieving performance parity with their reference implementations within similar step counts. This alignment underscores the high fidelity of our implementations, proving that decoupling the algorithmic logic does not compromise stability or performance.

Furthermore, we provide a qualitative comparison in Figure 3. Compared to the base FLUX.1-dev model, all variants finetuned with the PickScore reward exhibit enhanced visual quality and stronger alignment with human aesthetic preferences. The fact that these consistent improvements are achieved by simply switching the argument `trainer_type` in the configuration validates Flow-Factory as a robust and extensible platform for preference alignment.

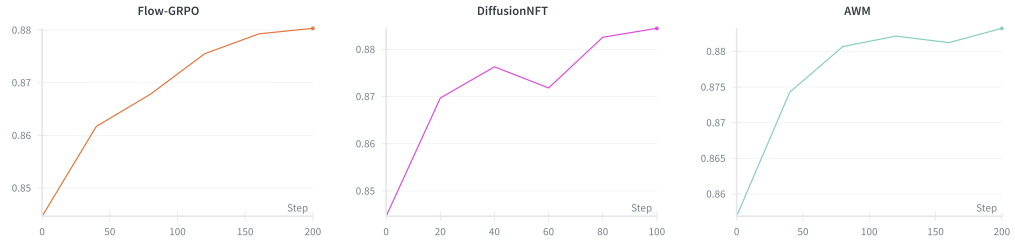

Figure 2: Reproduction of reward curves for Flow-GRPO, DiffusionNFT, and AWM on Flux.1-dev with PickScore reward.

4.3 TRAINING EFFICIENCY

We evaluate the impact of preprocessing-based memory optimization on training efficiency using Flux.1-dev with PickScore reward on 8×H200 GPUs (140GB each). Each experiment runs for 100 training steps. Table 2 summarizes the results.

Table 2: Training efficiency comparison with and without preprocessing on Flux.1-dev (100 steps).

Metric	**Without Preprocessing**	**With Preprocessing**
Peak GPU Memory (per device)	61.08 GB	53.14 GB (↓13.0%)
Time per Step	144.02 s	82.68 s (1.74×)

Preprocessing reduces GPU memory consumption by 13.0% (7.94 GB per device) by offloading frozen text encoders components after caching their outputs. More significantly, eliminating redundant encoding operations during training yields a 1.74× speedup, reducing per-step time from 144s to 82s. These improvements enable larger batch sizes and faster iteration cycles, making RL fine-tuning more accessible on commodity hardware.

Figure 3: Qualitative comparison of different RL algorithms on Flux.1-dev with PickScore reward. Each column shows generations for the same prompt across methods. All RL-finetuned models show improved visual quality compared to the base Flux.1-dev model.

5 Conclusion

We presented Flow-Factory, a unified and extensible framework that decouples algorithms, models, and rewards for reinforcement learning on flow-matching models. Its modular architecture enables flexible cross-combination of components through configuration alone, while preprocessing-based memory optimization and a flexible multi-reward system ensure training efficiency and algorithmic generality. Experiments confirm that Flow-Factory faithfully reproduces state-of-the-art results with significant efficiency gains. We hope Flow-Factory lowers the engineering barrier and accelerates community research on RL-based alignment of generative models.

ACKNOWLEDGMENTS

This work is supported by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM101), the National Natural Science Foundation of China (No. 62272374, No. 62192781), the Natural Science Foundation of Shaanxi Province (No.2024JC-JCQN-62), the State Key Laboratory of Communication Content Cognition under Grant No. A202502, the Key Research and Development Project in Shaanxi Province (No. 2023GXLH-024) **and** by A*STAR Career Development Fund <Project No. C243512010>.

REFERENCES

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2024.

Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. Tempflow-grpo: When timing matters for grpo in flow models. *arXiv preprint arXiv:2508.04324*, 2025.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in neural information processing systems*, 36:36652–36663, 2023.

Black Forest Labs. Introducing flux.1 tools. `https://bfl.ai/blog/24-11-21-tools`, 2026.

Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. *arXiv preprint arXiv:2507.21802*, 2025.

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

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

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

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.

Jing Wang, Jiajun Liang, Jie Liu, Henglin Liu, Gongye Liu, Jun Zheng, Wanyuan Pang, Ao Ma, Zhenyu Xie, Xintao Wang, et al. Grpo-guard: Mitigating implicit over-optimization in flow matching via regulated clipping. *arXiv preprint arXiv:2510.22319*, 2025a.

Yibin Wang, Zhimin Li, Yuhang Zang, Yujie Zhou, Jiazi Bu, Chunyu Wang, Qinglin Lu, Cheng Jin, and Jiaqi Wang. Pref-grpo: Pairwise preference reward-based grpo for stable text-to-image reinforcement learning. *arXiv preprint arXiv:2508.20751*, 2025b.

Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models. *arXiv preprint arXiv:2509.25050*, 2025a.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025b.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025.