Flow Matching Integrated Decision Transformer for Offline Reinforcement Learning

Asel Nurlanbek kyzy
Future Convergence Engineering
Korea University of Technology and Education
Cheonan, South Korea
aselbaekki@koreatech.ac.kr

Chang-Hun Ji
Future Convergence Engineering
Korea University of Technology and Education
Cheonan, South Korea
koir5660@koreatech.ac.kr

Min-Jun Kim
Future Convergence Engineering
Korea University of Technology and Education
Cheonan, South Korea
june573166@koreatech.ac.kr

Youn-Hee Han
Future Convergence Engineering
Korea University of Technology and Education
Cheonan, South Korea
yhhan@koreatech.ac.kr

***Abstract*—The Decision Transformer has significantly advanced offline reinforcement learning by reframing policy optimization as a sequence modeling problem. This approach offers strong scalability, effective long-term credit assignment, and robust generalization across diverse tasks. However, DT's reliance on sample-based learning from limited offline datasets inherently restricts its coverage of the state–action space and leads to the accumulation of autoregressive prediction errors, especially in long-horizon scenarios. Concurrently, Flow Matching has emerged as a powerful generative modeling framework that learns a continuous-time vector field to transport a simple prior distribution toward a complex target distribution, providing stable training, efficient inference, and distribution-level supervision without stochastic sampling. Motivated by the complementary advantages of these approaches, this paper proposes Decision Transformer with Integrated Flow Matching (DT+FM). This unified framework incorporates Flow Matching into the Decision Transformer architecture to achieve more stable and expressive action generation. By replacing purely sample-level supervision with distribution-based learning, DT+FM expands effective coverage of the state-action space and mitigates compounding prediction errors that arise during long-horizon decision-making. Experimental results on MuJoCo continuous control tasks and RoboMimic manipulation benchmarks demonstrate that DT+FM consistently outperforms the standard Decision Transformer. It offers improved stability, enhanced robustness under limited-data conditions, and substantially better performance in tasks characterized by long episodes and complex dynamics.**

***Index Terms*—Offline Reinforcement Learning, Decision Transformer, Flow Matching.**

I. INTRODUCTION

Offline reinforcement learning (RL) aims to learn optimal decision-making policies entirely exclusively from fixed, pre-collected datasets without further interaction with the environment. This approach is particularly valuable in settings where online data collection is costly, unsafe, or impractical, such as in robotics, autonomous driving, and healthcare [1]–[3]. By removing the necessity for additional exploration, offline RL enhances safety and data efficiency while still enabling the acquisition of complex behaviors. While offline RL traditionally employed temporal-difference methods, recent advancements have reframed it as a sequence modeling problem, most notably realized through the Decision Transformer (DT) [12]. DT models trajectories as autoregressive sequences of return-to-go (RTG), states, and actions. This approach replaces temporal-difference learning with supervised prediction and consequently enables strong performance across diverse benchmarks with stable, scalable training. However, DT remains constrained by its reliance on sample-based supervision, which limits generalization beyond the empirical data distribution [1], and by its autoregressive structure, which can compound prediction errors over long horizons [12], [26]. To address these generalization and stability challenges, Flow Matching (FM) [19] has emerged as a promising generative modeling framework. It learns continuous-time vector fields capable of transporting simple prior distributions toward complex target distributions. FM provides deterministic, simulation-free training of Continuous Normalizing Flows [20] and offers stable optimization and efficient inference without stochastic sampling. When applied to policy learning, FM enables distribution-level modeling that can notably enhance robustness under distributional shift and reduce error accumulation [21].

Motivated by the complementary strengths of sequence modeling and distribution-based learning, this paper introduces the *Decision Transformer with Integrated Flow Matching* (DT+FM). This unified framework augments the standard DT by incorporating a flow-based objective. This integration is designed to leverage FM's distribution modeling capabilities to broaden state-action coverage and significantly stabilize long-horizon predictions within the autoregressive structure of the transformer. Our experimental evaluations, conducted on MuJoCo [33] continuous control and RoboMimic [34] manipulation tasks, demonstrate that DT+FM consistently improves stability, robustness, and overall performance compared to the standard DT, especially in scenarios characterized by long horizons and data-limited conditions.

II. Background and Preliminaries

Offline RL aims to learn decision-making policies solely from fixed datasets, eliminating the need for further interaction with the environment. We consider a Markov Decision Process (MDP) with state space S, action space A, transition kernel P, reward function R, and discount factor γ. The objective of a policy π is to maximize the expected discounted return:

$$J(\pi) = \mathbb{E}_{\pi,\,P}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)\right]. \tag{1}$$

In offline RL, the agent has no further interaction with the environment and must instead learn from a fixed dataset

$$\mathcal{D} = \{(s_t^{(i)}, a_t^{(i)}, r_t^{(i)}, s_{t+1}^{(i)})\}_{i,t},$$

collected by an unknown behavior policy. While this setting is attractive in safety-critical or cost-sensitive domains [1], [3], it introduces a distributional mismatch between the dataset and the state-action distribution induced by the learned policy, which can lead to extrapolation and compounding errors [27].

Classical offline RL approaches, such as Behavior Cloning (BC) [4], offer stability and simplicity but often suffer from severe distribution shift in long-horizon tasks. To mitigate out-of-distribution action selection, value-regularized algorithms such as Batch-Constrained Q-Learning (BCQ) [5] and Conservative Q-Learning (CQL) [6] constrain the learned value function. Policy regularization methods such as BRAC [7] and AWAC [8] further enforce closeness to the behavior policy to reduce extrapolation error. More recent frameworks like Implicit Q-Learning (IQL) [9] improve stability through expectile regression and advantage-weighted regression. However, most offline RL methods remain sensitive to approximation errors and often lack mechanisms to capture global trajectory structure. Model-based approaches such as MOPO and COMBO [10], [11] attempt to broaden data coverage with synthetic rollouts but are limited by model bias.

DT [12] reconceptualizes offline RL as a sequence modeling problem using a causal Transformer [28]. It enables stable training by predicting actions autoregressively from RTG, states, and past actions. A trajectory of length $\mathcal{T}$ is represented as an interleaved sequence:

$$\mathcal{T} = (g_1, s_1, a_1, g_2, s_2, a_2, \ldots, g_T, s_T, a_T), \tag{2}$$

where the RTG at time t is defined as

$$g_t = \sum_{t'=t}^{T} r_{t'}. \tag{3}$$

Given a context window of length K, DT parameterizes a conditional policy

$$\pi_{\mathrm{DT}}\big(a_t \mid s_{t-K:t}, a_{t-K:t-1}, g_{t-K:t}; \theta\big),$$

implemented by a GPT-style causal Transformer that attends only to past tokens. Training reduces to a supervised regression objective over the offline dataset:

$$\begin{aligned}\mathcal{L}_{\mathrm{DT}}(\theta) &= \mathbb{E}_{(s,a,g)\sim\mathcal{D}}\left[\|a_t - \hat{a}_t\|^2\right], \\ \hat{a}_t &= \pi_{\mathrm{DT}}\big(\cdot \mid s_{t-K:t}, a_{t-K:t-1}, g_{t-K:t}; \theta\big).\end{aligned} \tag{4}$$

At evaluation time, a target RTG g_1 and an initial state s_1 are provided, and actions are generated autoregressively. After each step, the RTG is updated as

$$g_{t+1} = g_t - r_t. \tag{5}$$

While this formulation enables scalable sequence-based control, extensions such as MGDT [13], Prompt-DT [14], and CDT [15] explore multi-task, prompt-conditioned, and constrained learning settings. Structural variants such as GDT [16] and dynamic extensions [17], [18] further demonstrate DT's adaptability. Nonetheless, DT remains vulnerable to restricted dataset coverage and the accumulation of autoregressive prediction errors, which degrade performance in long-horizon tasks due to its purely sample-based supervision and autoregressive structure [12], [26].

FM [19] provides a complementary approach for distribution-aware generative modeling. FM learns a time-dependent vector field that transports samples from a simple base distribution p_0 to a complex target distribution p_1. Here, $x_0 \sim p_0$ represents a sample drawn from the base distribution, while $x_1 \sim p_1$ denotes a sample from the target distribution. Specifically, FM defines a flow map $\phi_\tau(x)$ through the ordinary differential equation (ODE):

$$\frac{d}{d\tau}\,\phi_\tau(x) = v_\tau\big(\phi_\tau(x)\big), \qquad \phi_0(x) = x, \tag{6}$$

where v_τ is a neural vector field. When v_τ is learned correctly, the pushforward of p_0 under ϕ_1 matches p_1.

Direct supervision of v_τ is intractable, so FM constructs simple conditional probability paths, typically via linear interpolation:

$$x_\tau = (1-\tau)x_0 + \tau x_1, \tag{7}$$

for which the target velocity has a closed-form expression:

$$v_\tau(x_\tau \mid x_1) = \frac{x_1 - x_\tau}{1-\tau}. \tag{8}$$

A neural approximation $\hat{v}_\tau$ is trained using the Conditional Flow Matching (CFM) objective:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{x_0\sim p_0,\, x_1\sim p_1,\, \tau\sim U[0,1]}\left[\|\hat{v}_\tau(x_\tau;\theta) - v_\tau(x_\tau \mid x_1)\|^2\right], \tag{9}$$

which provides unbiased gradients without solving the ODE during training. For conditional generation, the vector field is augmented with context c, yielding $\hat{v}_\tau(x_\tau, c; \theta)$ and a conditional CFM objective [21]. At inference time, sampling reduces to integrating the learned ODE.

Compared to diffusion models, FM offers deterministic, simulation-free training and typically requires fewer function evaluations during sampling, making it attractive for distribution-aware policy modeling [22], [24]. Conditional FM has recently been explored in RL, leading to flow-based methods such as Flow Q-Learning [22], Flow Policy Optimization [23], and ReinFlow [24], as well as latent-space formulations such as LFM [25]. While flow-based approaches provide expressive, distribution-aware modeling, they typically

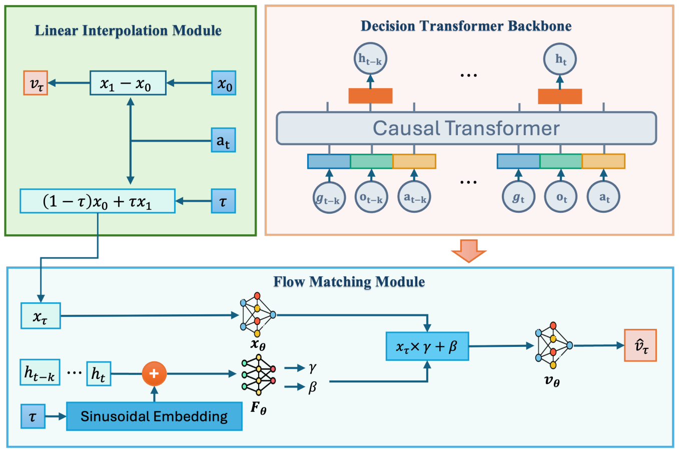


Fig. 1: Conceptual overview of DT+FM.

lack the long-horizon temporal reasoning capabilities offered by Transformer-based sequence models.

This complementary relationship motivates hybrid frameworks that integrate sequence modeling with distributional action generation. The proposed DT+FM framework builds on this idea by conditioning FM policy on Transformer representations, combining long-range temporal reasoning with continuous, distribution-level supervision to improve stability and generalization in offline RL.

III. Decision Transformer with Integrated Flow Matching (DT+FM)

We propose DT+FM, a unified offline RL framework that augments the DT with a FM module. DT provides sequence-level context through a causal Transformer, whereas FM provides distribution-level supervision by learning a continuous-time vector field that transports samples from a simple prior to the data-supported action distribution along a conditional probability path [19], [21]. This integration targets two limitations of standard DT: (i) *sample-based bias* due to restricted dataset coverage, and (ii) *autoregressive error accumulation* in long-horizon rollouts. In DT+FM, the Transformer hidden representation h_t conditions the FM vector field, enabling a sequence-aware denoising process that maps noisy action samples to clean, data-consistent actions, thereby reducing trajectory drift and improving stability under distribution shift [24].

As illustrated in Fig. 1, a trajectory window

$$\mathcal{T}_t = (g_{t-K+1}, o_{t-K+1}, a_{t-K+1}, \ldots, g_t, o_t, a_t) \tag{10}$$

is embedded and processed by a masked self-attention Transformer to produce contextual embeddings h_t. Unlike vanilla DT, which decodes h_t with a deterministic action head, DT+FM uses h_t as conditioning input to a flow model. Concretely, we treat the terminal point of the flow as the action itself ($x_1 = a_t$). During training, we sample $x_0 \sim \mathcal{N}(0, I)$ and $\tau \sim U[0,1]$, and construct an intermediate point on the conditional path

$$x_\tau = (1-\tau)x_0 + \tau x_1, \tag{11}$$

which matches the standard linear interpolation used in FM [19]. FM module predicts a context-conditioned vector field $\hat{v}_\tau = v_\theta(x_\tau, \tau, h_t)$ and is trained by regressing to the corresponding target field under CFM [19], [21]. To make the vector field explicitly depend on both time and trajectory context, DT+FM applies Feature-wise Linear Modulation (FiLM) [32]: a time embedding $\psi(\tau)$ and context h_t produce per-feature scale and shift parameters (γ, β), which modulate intermediate activations in the vector-field network.

The overall optimization jointly updates the DT backbone parameters ϕ and the FM parameters θ by minimizing the FM regression loss, and an optional auxiliary DT action regression loss from Eq. (4) can be added without changing the sampling procedure. At inference time, DT+FM follows an autoregressive rollout as in DT, but replaces deterministic decoding with flow-based action generation: given h_t and an initial noise sample x_0, the action is obtained by integrating the learned ODE

$$\frac{dx(\tau)}{d\tau} = v_\theta(x(\tau), \tau, h_t), \qquad \tau \in [0,1], \tag{12}$$

using a standard numerical solver such as Euler or Runge–Kutta [29]–[31], and taking the terminal state as the executed action $\hat{a}_t = x_1$. RTG is updated as in DT, $g_{t+1} = g_t - r_t$. By generating each action through a context-conditioned denoising flow trained to map noisy samples toward data-supported actions, DT+FM reduces deviation from the empirical behavior distribution and mitigates compounding errors over long horizons, yielding more stable offline policy execution.

IV. Experiments

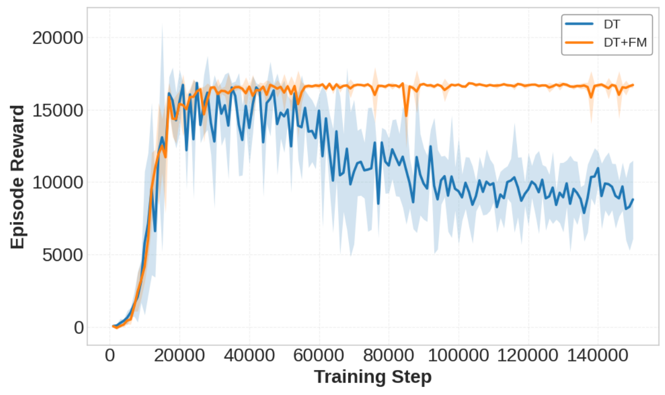
A. Environments and Datasets

The proposed DT+FM framework is evaluated on two standard offline RL domains: continuous-control locomotion tasks from MuJoCo [33] and robotic manipulation tasks from RoboMimic [34]. For locomotion, expert-level datasets from the Minari benchmark [35] are used for HalfCheetah, Walker2d, Ant, and Humanoid. These environments span a wide range of state and action dimensionalities and exhibit increasing control complexity. For manipulation, we consider the *Lift* and *Can* tasks from RoboMimic using the Proficient-Human (PH) datasets, which consist of high-quality teleoperated demonstrations and require precise long-horizon manipulation.

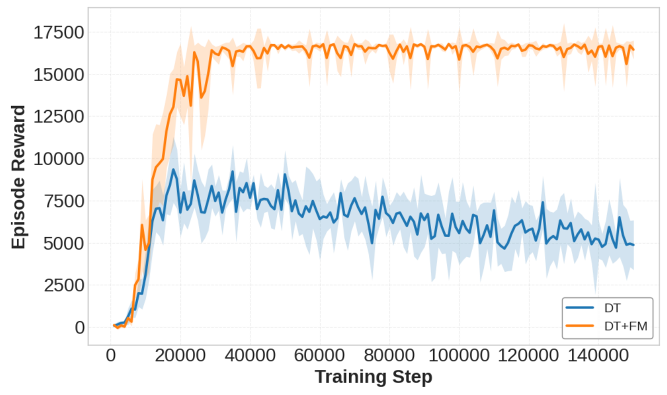
B. Baseline and Evaluation Protocol

To isolate the contribution of FM, DT+FM is compared exclusively against the standard DT [12]. Both methods use identical Transformer backbones, optimization settings, and training schedules. Two evaluation regimes are considered:

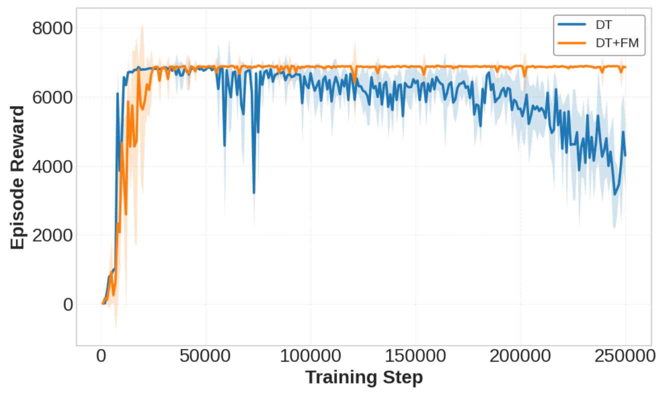
- **Full Dataset:** Both models are trained on the complete expert-level dataset.
- **50% Dataset:** The dataset size is reduced by 50% to simulate limited offline data. Training on a randomly subsampled half of the dataset to assess robustness under data scarcity.



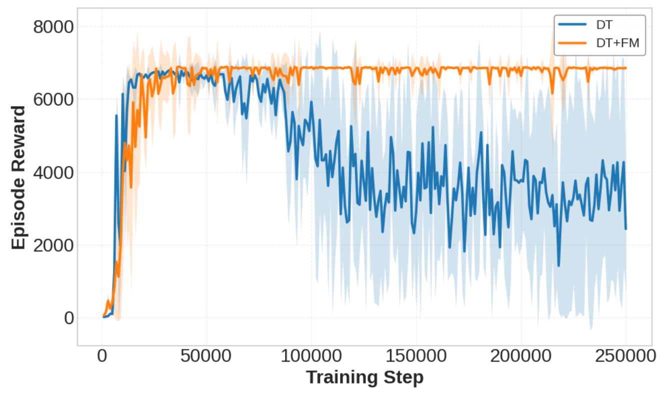
(a) HalfCheetah Expert (Full Dataset)



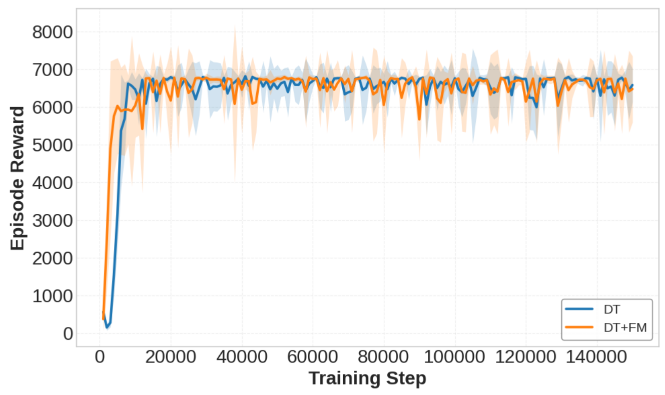
(b) HalfCheetah Expert (50% Dataset)



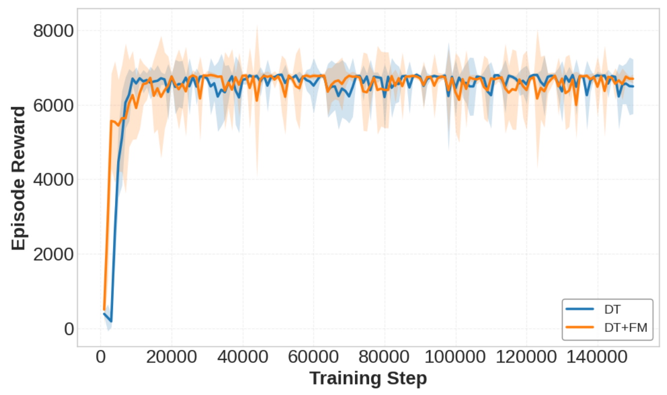
(c) Walker2d Expert (Full Dataset)



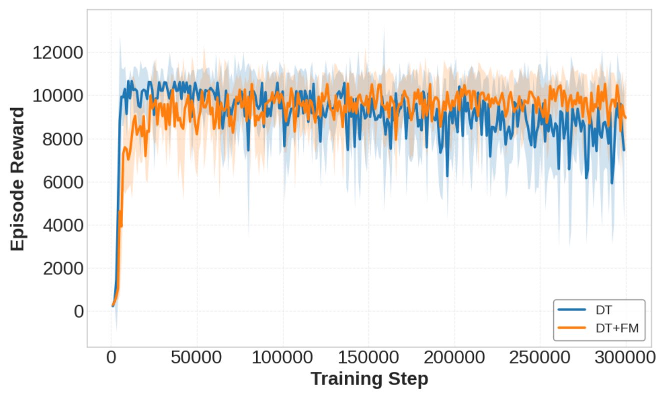
(d) Walker2d Expert (50% Dataset)



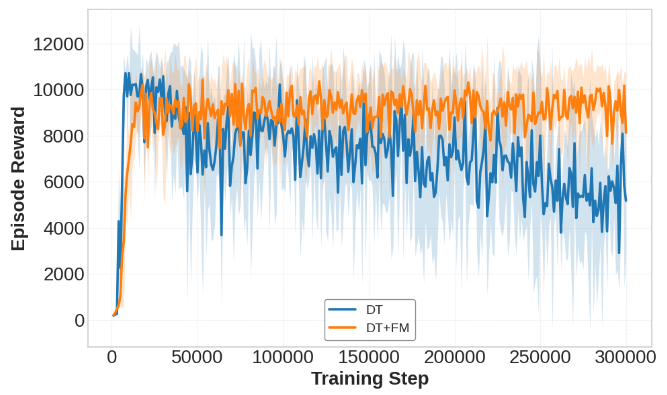
(e) Ant Expert (Full Dataset)



(f) Ant Expert (50% Dataset)



(g) Humanoid Expert (Full Dataset)



(h) Humanoid Expert (50% Dataset)

Fig. 2: Performance comparison of DT and DT+FM across MuJoCo tasks with full datasets (left column) and reduced datasets (50%, right column). Each row corresponds to a different environment: (a,b) HalfCheetah, (c,d) Walker2d, (e,f) Ant, (g,h) Humanoid, (g,h). Evaluation was performed with target RTGs set to expert-level returns: HalfCheetah = 20000, Walker2d = 10000, Ant = 10000, Humanoid = 12000.

TABLE I: Key hyperparameters for DT and DT+FM models.

Component	Parameter	Value
Transformer (DT)		
Embedding dimension		256
Number of layers		10
Attention heads		2
Sequence length		20
Flow Matching (DT+FM only)		
Architecture		3-layer MLP
Hidden dimension		512
Time embedding		Sinusoidal
Optimization		
Optimizer		AdamW
Learning rate		1×10^{-4}
Weight decay		1×10^{-3}
Training Setup		
Batch size		256
Training steps		150,000
Dataset fraction		1.0 / 0.5
Evaluation		
Evaluation frequency		Every 1,000 steps
Episodes per evaluation		5

Performance is reported using normalized episodic returns for MuJoCo and task success rates for RoboMimic. All results are averaged over five random seeds and reported with 95% confidence intervals.

C. Implementation Details

All models are implemented in PyTorch and share a common architecture consisting of a 10-layer causal Transformer with 256-dimensional embeddings and 2 attention heads. The Flow Matching module in DT+FM is implemented as a 3-layer multilayer perceptron with 512 hidden units, sinusoidal time embeddings, and FiLM-based conditioning [32]. Training is performed using AdamW with a learning rate of 1×10^{-4} and batch size 256 for 150k gradient steps. During inference, the learned flow ODE is solved using fixed-step Euler integration with 50 steps [29]–[31]. All key architectural and optimization hyperparameters are summarized in Table I.

D. Results on MuJoCo Locomotion

Across all four MuJoCo environments, DT+FM consistently outperforms the baseline DT under both full-data and reduced-data regimes, as shown in Fig. 2. DT+FM converges more rapidly, exhibits lower variance across random seeds, and maintains stable performance over long horizons. In contrast, DT experiences notable performance degradation under the 50% dataset setting-particularly in Walker2d and Humanoid-indicating sensitivity to compounding errors. These results demonstrate that FM improves distributional generalization and effectively mitigates autoregressive error accumulation in offline locomotion control.

E. Results on RoboMimic Manipulation

Table II reports success rates for the *Lift* and *Can* tasks. Under the full-data regime, DT+FM matches or slightly improves upon DT, indicating that FM does not compromise performance when sufficient expert data is available. Under the 50% dataset setting, DT+FM yields substantial gains, improving success rates from 0.88 to 0.98 on *Lift* and from 0.55 to 0.85 on *Can*. These improvements highlight the importance of distribution-aware supervision in data-limited manipulation tasks, where DT alone struggles with out-of-distribution generalization and long-horizon consistency.

TABLE II: Success rate comparison (mean ± std) of DT and DT+FM on RoboMimic manipulation tasks under full and 50% data regimes.

Task	Data Regime	DT	DT+FM
Lift	Full Data	96.0 ± 2.0	**96.0** ± **2.0**
	50% Data	88.0 ± 3.2	**98.0** ± **1.4**
Can	Full Data	90.0 ± 3.0	**92.0** ± **2.7**
	50% Data	55.0 ± 5.0	**85.0** ± **3.6**
Average	Full Data	93.0 ± 2.5	**94.0** ± **2.3**
	50% Data	71.5 ± 4.1	**91.5** ± **2.5**

F. Summary

Overall, DT+FM achieves higher returns, improved stability, and stronger robustness in low-data regimes across both locomotion and manipulation benchmarks. The experimental results confirm that integrating FM significantly enhances DT by improving distributional modeling and reducing compounding errors in offline RL.

V. Conclusion

This work presented **DT+FM**, a unified offline RL framework that integrates FM with the DT. The approach addresses two major limitations of standard DT: restricted generalization due to sample-based training and the accumulation of autoregressive prediction errors. By learning a context-conditioned vector field in latent action space, FM provides distribution-aware supervision that guides noisy action samples toward data-consistent trajectories.

Experimental results across MuJoCo locomotion and RoboMimic manipulation tasks demonstrate that DT+FM offers improved stability, faster convergence, and stronger performance under both full and data-limited regimes. In particular, DT+FM shows substantial robustness when training data are scarce, indicating enhanced distributional coverage and reduced sensitivity to out-of-distribution states.

Overall, DT+FM offers a simple yet effective enhancement to sequence-modeling approaches for offline RL by combining long-horizon temporal reasoning with continuous, flow-based action generation. Future work will focus on extending the framework to sim-to-real transfer scenarios, including high-fidelity locomotion tasks in IsaacGym using the Unitree Go1 quadruped, as well as incorporating visual observations and online fine-tuning to further improve robustness in real-world robotic deployments.

Acknowledgment

This research was supported by two Basic Science Research Programs through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2023R1A2C1003143 and No. NRF-2018R1A6A1A03025526).

References

[1] S. Levine, A. Kumar, G. Tucker, and J. Fu, “Offline reinforcement learning: Tutorial, review, and perspectives on open problems,” *arXiv:2005.01643*, 2020.

[2] G. Dulac-Arnold, D. J. Mankowitz, and T. Hester, “Challenges of real-world reinforcement learning,” *arXiv:1904.12901*, 2019.

[3] R. Prudêncio, T. Maximo, and L. Aguirre, “A survey on offline reinforcement learning,” *arXiv:2302.07265*, 2023.

[4] D. Pomerleau, “ALVINN: An autonomous land vehicle in a neural network,” *NeurIPS*, 1989.

[5] S. Fujimoto, D. Meger, and D. Precup, “Batch-constrained deep Q-learning,” *ICML*, 2019.

[6] A. Kumar, A. Zhou, G. Tucker, and S. Levine, “Conservative Q-learning for offline reinforcement learning,” *NeurIPS*, 2020.

[7] Y. Wu, G. Tucker, and O. Nachum, “Behavior-regularized actor–critic,” *NeurIPS*, 2019.

[8] A. Nair, M. Dalal, A. Gupta, and S. Levine, “AWAC: Accelerating online reinforcement learning with offline datasets,” *arXiv:2006.09359*, 2020.

[9] I. Kostrikov, H. Nair, and S. Levine, “Offline reinforcement learning with implicit Q-learning,” *NeurIPS*, 2021.

[10] T. Yu, G. Thomas, L. Yu, S. Levine, J. Kim, and P. Abbeel, “MOPO: Model-based Offline Policy Optimization,” in *Adv. Neural Inf. Process. Syst.*, 2020.

[11] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, “COMBO: Conservative Offline Model-Based Policy Optimization,” in *Adv. Neural Inf. Process. Syst.*, 2021.

[12] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, “Decision Transformer: Reinforcement learning via sequence modeling,” in *Adv. Neural Inf. Process. Syst.*, 2021.

[13] K.-H. Lee, O. Nachum, M. Yang, L. Lee, D. Freeman, W. Xu, S. Guadarrama, I. Fischer, E. Jang, H. Michalewski, and I. Mordatch, “Multi-Game Decision Transformers,” in *Int. Conf. Learn. Represent.*, 2022.

[14] H. Liu, B. Liu, M. Li, Y. Chen, S. Zhang, and C. Yu, “Prompt-based Decision Transformer,” in *Int. Conf. Learn. Represent.*, 2023.

[15] W. Xu, A. R. Shah, Y. Hanada, Y. Lu, and Y. T. Yu, “Constrained Decision Transformer,” in *Int. Conf. Learn. Represent.*, 2022.

[16] Y. Li, L. Wang, and J. Shi, “Graph Decision Transformer,” in *Adv. Neural Inf. Process. Syst.*, 2023.

[17] Q. Zheng, A. Zhang, and A. Grover, “Online Decision Transformer,” *arXiv preprint arXiv:2202.02850*, 2022.

[18] E. H. Jiang, Z. F. Zhang, D. Zhang, A. Lizarraga, C. Xu, Y. Zhang, S. Zhao, Z. Xu, P. Yu, Y. Tang, D. Kong, and Y. N. Wu, “DODT: Enhanced Online Decision Transformer Learning through Dreamer's Actor-Critic Trajectory Forecasting,” *arXiv preprint arXiv:2410.11359*, 2024.

[19] Y. Lipman, R. T. Q. Chen, C. L. A. Song, J. Gao, A. Rajeswaran, and I. Mordatch, “Flow matching for generative modeling,” *arXiv preprint arXiv:2210.02747*, 2023.

[20] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, “Neural Ordinary Differential Equations,” in *Adv. Neural Inf. Process. Syst.*, 2018.

[21] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Q. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat, “Flow Matching Guide and Code,” *arXiv preprint arXiv:2412.06264*, 2024.

[22] S. Park, Q. Li, and S. Levine, “Flow Q-Learning,” in *Int. Conf. Mach. Learn.*, 2025.

[23] D. McAllister, S. Ge, B. Yi, C. M. Kim, E. Weber, H. Choi, H. Feng, and A. Kanazawa, “Flow Matching Policy Gradients,” in *Int. Conf. Mach. Learn.*, 2025.

[24] T. Zhang, C. Yu, S. Su, and Y. Wang, “ReinFlow: Fine-tuning Flow Matching Policy with Online Reinforcement Learning,” *arXiv preprint arXiv:2505.22094*, 2025.

[25] Q. Dao, H. Phung, B. Nguyen, and T. Le, “Flow matching in latent space,” *arXiv preprint arXiv:2307.08698*, 2023.

[26] J. Ho and S. Ermon, “Generative adversarial imitation learning,” *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[27] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” *AISTATS*, 2011.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *NeurIPS*, 2017.

[29] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 3rd ed. Chichester, U.K.: Wiley, 2016.

[30] C. Runge, “Über die numerische Auflösung von Differentialgleichungen,” *Mathematische Annalen*, vol. 46, no. 2, pp. 167–178, 1895.

[31] W. Kutta, “Beitrag zur näherungsweisen Integration totaler Differentialgleichungen,” *Zeitschrift für Mathematik und Physik*, vol. 46, pp. 435–453, 1901.

[32] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, “FiLM: Visual reasoning with a general conditioning layer,” in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2018, pp. 3942–3951.

[33] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, 2012, pp. 5026–5033.

[34] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and A. Zhu, “RoboMimic: A benchmark for imitation learning and offline reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2591–2598, Apr. 2021.

[35] O. G. Younis, R. Perez-Vicente, J. U. Balis, W. Dudley, A. Davey, and J. K. Terry, “Minari: A standard format for offline reinforcement learning datasets,” Zenodo, 2024. [Online]. Available: https://doi.org/10.5281/zenodo.13767625