Streaming Flow Policy

Simplifying diffusion/flow policies by treating robot trajectories as flow trajectories

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Kaelbling, Siddharth Ancha

Website: https://streaming-flow-policy.github.io†

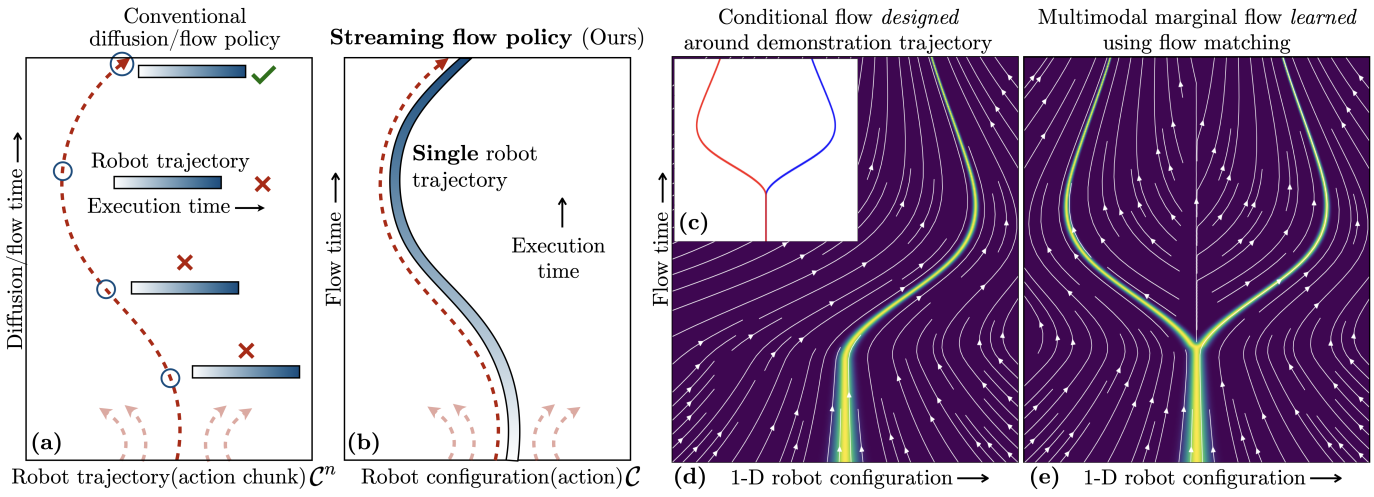

Fig. 1: **(a)** Diffusion policy [1] and flow-matching policy [2] input a history of observations (not shown) to predict a "chunk" of future robot actions. The x-axis represents the action (robot configuration) space, and the $+y$-axis represents increasing diffusion timesteps. Conventional diffusion/flow policies sample a "trajectory of trajectories" *i.e.* a diffusion/flow trajectory of robot trajectories. This framework discards all intermediate robot trajectories that are computed, and must wait for the diffusion/flow process to complete before the first actions can be executed on the robot. **(b)** We introduce an imitation learning framework that simplifies diffusion/flow policies by *treating robot trajectories as flow trajectories*. We develop a novel flow-matching algorithm in C-space that starts from the current robot configuration at and iteratively samples a sequence of configurations that constitutes a *single* robot trajectory. This aligns the "diffusion time" of the flow sampling process with the "execution time" of the robot trajectory. Importantly, the computed actions can be streamed to the robot's actuators on the fly *during* the flow sampling process, enabling significantly faster and reactive policies. **(c)** To illustrate our method, we consider a running toy example of 1-D robot trajectory (x-axis) with two demonstration trajectories shown in blue and red. **(d)** Given a demonstration trajectory from the training set (*e.g.* the blue one), we first *construct* a conditional flow that samples trajectories from a thin tube around the demonstration. **(e)** Then, we use flow matching [3] to *learn* a velocity field that expresses multi-modal trajectories. The marginal distribution over configurations at each time slice induced by the learned velocity field matches the training distribution. **(d)** We show that constructing conditional flows that *stabilize* around demonstration trajectories reduces distribution shift and improves imitation learning performance. Our method is able to represent multi-modal distributions over robot trajectories like diffusion/flow policies, while also being able to stream actions during the flow sampling process.

***Abstract*— Recent advances in diffusion/flow-matching policies have enabled imitation learning of complex, multi-modal action trajectories for robotic control. However, they are computationally expensive because they sample a *trajectory of trajectories*—a diffusion/flow trajectory of robot trajectories. They discard intermediate robot trajectories, and must wait for the sampling process to complete before any actions can be executed on the robot. In this work, we propose a novel framework that simplifies diffusion/flow policies by *treating robot trajectories as flow trajectories*. Instead of starting from pure noise, our algorithm starts from the current robot configuration. Then, it incrementally integrates a velocity field learned via flow matching to produce a sequence of robot configurations that constitute a *single* trajectory. This enables computed actions to be streamed to the robot on-the-fly *during* the flow sampling process, for significantly faster and reactive policies. It is well-suited for receding horizon control where we can adaptively generate only as many actions as are executed on the robot, and no more. Despite streaming, our method retains the ability to model multi-modal behavior. We show that training flows that *stabilize* around demonstration trajectories reduces distribution shift and improves imitation learning performance. Streaming flow policy outperforms prior diffusion/flow policies on imitation learning benchmarks while enabling faster policy execution and tighter sensorimotor loops for learning-based robot control.**

All authors are affiliated with Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Correspondence email: sancha@mit.edu.

†Project website contains (1) qualitative visualizations, (2) an appendix, (3) self-contained notebooks to illustrate our method, and (4) source code.

I. Introduction

Recent advances in robotic imitation learning, such as diffusion policy [1, 4] and flow-matching policy [2, 5, 6] have enabled robots to learn complex, multi-modal action distributions for challenging real-world tasks such as cooking,

laundry folding, robot assembly and navigation [7]. They input a history of observations to generate a sequence of actions (also called an "action chunk"). Conventional diffusion/flow policies are a direct application of diffusion models [8, 9] and flow-matching [3] to robot action sequences — they formulate the generative process as probabilistic transport in the space of action *sequences*, starting from pure Gaussian noise. Therefore, diffusion/flow policies represent a "*trajectory of trajectories*" — a diffusion/flow trajectory of robot trajectories (Fig. 1a). This approach has several drawbacks. The sampling process discards all intermediate robot trajectories, making diffusion/flow policies computationally inefficient. Importantly, the robot must wait for the diffusion/flow process to complete before being able to execute any actions. Therefore, diffusion/flow policies often require careful parameter tunning to admit tight control loops.

In this work, we propose a novel imitation learning framework that harnesses the temporal structure of robot trajectories. We simplify diffusion/flow policies by treating *robot trajectories as flow trajectories* (see Fig. 1b). We leverage two key properties specific to robotics: (1) robot action chunks are usually represented as trajectories in configuration space which are tracked by a low-level position controller, (2) the current robot configuration can be accurately measured via proprioceptive sensors (*e.g.* joint encoders). Our aim is to learn a flow transport in the robot's *configuration space* (as opposed to trajectory space). The sampling process starts from the current measured robot configuration, and iteratively generates a sequence of configurations that forms a *single* robot trajectory. The "flow time" — indicating progress of the flow sampling process — coincides with execution time of the sampled robot trajectory. Iteratively generating the sequence of configurations allows the trajectory to be *streamed* to the robot's position controller on-the-fly *during* the flow generation process, significantly improving the policy's speed and reactivity.

We show how a streaming flow policy with the above desiderata can be learned using flow matching [3]. Given a demonstration trajectory from the training set (Fig. 1c), we *construct* a velocity field conditioned on this example that samples paths in a narrow Gaussian tube around the demonstration (Fig. 1d). Our training procedure is remarkably simple — we regress a neural network $v_\theta(q, t \mid h)$ that takes as input (1) an observation history h, (2) flow timestep $t \in [0, 1]$, and (3) configuration q at time t, to match the constructed velocity field. We re-use existing architectures for diffusion/flow policy and only modify the output dimension of the network. Flow matching guarantees that the *marginal* flow learned over all training trajectories, as shown in Fig. 1e, is multi-modal. Specifically, the marginal distribution of configurations at *each timestep* t matches that of the training distribution. Our approach is able to retain diffusion policy's ability to represent multi-modal trajectories while allowing for streaming trajectory generation.

How should we construct the target velocity field? Prior work [10] has shown that low-level stabilizing controllers can reduce distribution shift and improve theoretical imitation learning guarantees. We leverage the flexibility of the flow matching framework to construct velocity fields that *stabilize* around a given demonstration trajectory, by adding velocity components that guide the flow back to the demonstration. We find that stabilizing flow significantly improves performance.

The ability to represent multi-modal distributions is primarily motivated by the need to prevent "averaging" distinct but valid demonstrations of the same task [1]. While multi-modality is crucial during training, the learned policy can be run deterministically at test time without loss in performance. For example, ACT [11] sets its variance parameter to zero at test time to produce deterministic behavior. In order to learn multi-modal distributions during training, streaming flow policy requires a small amount of Gaussian noise added to the start configuration. However, we wish to avoid adding noise to configurations at test time, so we execute the policy deterministically. In App. II, we present a variant of streaming flow policy in an extended state space that decouples stochasticity into auxiliary latent variables. This variant allows us to sample multiple modes of the trajectory distribution at test time while deterministically starting the sampling process with the current robot configuration.

In App. III, we discuss some limitations of streaming flow policy. Unlike diffusion/flow policies, streaming flow policy is only guaranteed to match the marginal distribution of configurations at each timestep, but not necessarily the joint distribution. This means that our method can produce trajectories that are compositions of segments of training trajectories, even if the composition was not part of the training dataset. While this may be seen as a limitation of our method, we argue that for most robotics tasks, compositionality is not only valid, but a desirable property that requires fewer demonstrations. Furthermore, while streaming flow policy is unable to capture global constraints that can only be represented in the joint distribution, it can learn local constraints such as joint constraints, and convex velocity constraints; see App. III for more details. In practice, we find that streaming flow policy performs no worse than diffusion policy while being significantly faster.

II. Background & Problem Formulation

We consider the problem of imitating future robot trajectories from histories of observations as input. Specifically, we assume access to a training set $\mathcal{D} = \{(h_i, \xi_i)\}_{i=1}^N$ of N tuples, where $h_i \in \mathcal{H}$ is a history of observations, and ξ_i is a demonstration trajectory of *future* robot configurations $q \in \mathcal{C}$. The time horizon $T_p \in \mathbb{R}_{\geq 0}$ of the trajectory to be predicted can be an arbitrary hyperparameter. For simplicity, we re-scale the time interval by dividing by T_p; therefore $\xi_i : [0, 1] \to \mathcal{C}$. See Table I for a complete list of notation. Our aim is to learn a policy that outputs a potentially multi-modal distribution over future trajectories ξ given a history of observations h.

We formulate *streaming flow policy*, with model parameters θ, as a history-conditioned velocity field $v_\theta(q, t \mid h)$. For a given history $h \in \mathcal{H}$, $t \in [0, 1]$, and configuration $q \in \mathcal{C}$, the model outputs a velocity in the tangent space $T\mathcal{C}$ of $\mathcal{C}$.

Symbol	Description	Type
t	Flow time / normalized execution time	$[0,1]$
q	Robot configuration	$\mathcal{C}$
v	Velocity	$T\mathcal{C}$
h	Observation history	$\mathcal{H}$
$\xi \sim \mathcal{D}$	Demonstration trajectory	$[0,1] \to \mathcal{C}$
$\xi(t)$	Demonstration configuration at t	$\mathcal{C}$
$\dot{\xi}(t)$	Demonstration velocity at $t = \frac{d}{dt}\xi(t)$	$T\mathcal{C}$
$v_\theta(q, t \mid h)$	Learned marginal velocity field	$T\mathcal{C}$
$v_\xi(q,t)$	Conditional vel. field for demonstration ξ	$T\mathcal{C}$
$p_\xi(q \mid t)$	Marginal probability distribution over q at time t induced by v_ξ	$\Delta(\mathcal{C})$
$v^\star(q, t \mid h)$	Optimal marginal velocity field	$T\mathcal{C}$
$p^\star(q \mid t, h)$	Marginal probability distribution over q at time t induced by $v^\star$	$\Delta(\mathcal{C})$
k	Stabilizing gain	$\mathbb{R}_{\geq 0}$
σ_0	Initial standard deviation	$\mathbb{R}_{\geq 0}$

TABLE I: Mathematical notation used throughout the paper. $\mathcal{C}$ is the robot's configuration space, $T\mathcal{C}$ is tangent space of $\mathcal{C}$ that define velocities, and $\Delta(\mathcal{C})$ is the space of probability distributions over $\mathcal{C}$.

The velocity field is a neural ordinary differential equation (ODE) [12] that induces flow trajectories $q(t)$ in configuration space. The velocity field specifies the instantaneous time-derivative of the flow trajectory $\dot{q}(t) = v_\theta\left(q, t \mid h\right)$, where $\dot{q}(t) = \frac{dq}{dt}(t)$. When endowed with an initial probability distribution over $q(0)$, $v_\theta\left(q, t \mid h\right)$ represents a *continuous normalizing flow* [12, 13] that transforms the initial distribution to a new distribution $p_\theta\left(q \mid t, h\right)$, for every $t \in [0,1]$, in a deterministic and invertible manner. We want streaming flow policy to start sampling from the current robot configuration q_{curr}. However, invertible flows require a non-zero probability density on the domain $\mathcal{C}$ to be well behaved. Therefore, we chose a narrow Gaussian distribution around q_{curr} with a small variance σ_0^2 during training; we set $\sigma_0 = 0$ at inference time. A trajectory is generated by sampling from the initial distribution and integrating the velocity field as:

$$q(t) = q_0 + \int_0^t v_\theta\big(q(s), s \mid h\big)\, ds \quad \text{where} \quad q_0 \sim \mathcal{N}\big(q_{\text{curr}}, \sigma_0^2\big)$$

Importantly, standard ODE solvers can perform forward finite-difference integration auto-regressively, where integration at time t depends only on previously computed configurations $q(s), s \leq t$. This property allows us to stream the trajectory *during* the integration process, without needing to wait for the full trajectory to be computed. Next, we describe how we construct conditional velocity fields from trajectories $\xi \sim \mathcal{D}$ in the dataset, and learn $v_\theta\left(q, t \mid h\right)$ using flow matching.

III. Designing Conditional Velocity Fields

Given an observation history and demonstration trajectory (h, ξ) sampled from the training dataset $\mathcal{D}$, we first analytically construct a stabilizing *conditional* flow that travels closely along ξ. In particular, we construct a velocity field $v_\xi(q,t)$ and an initial distribution $p_\xi^0(q)$ the such that the induced marginal probability distributions $p_\xi(q \mid t)$ is a thin Gaussian tube around ξ. This is illustrated in Fig. 1(c, d). We construct the stabilizing conditional flow as:

$$v_\xi(q,t) \;\coloneqq\; \underbrace{\dot{\xi}(t)}_{\text{Demonstration velocity}} - \underbrace{k(q - \xi(t))}_{\text{Stabilization term}} \tag{1}$$

$$p_\xi^0(q) \;\coloneqq\; \mathcal{N}\left(q \mid \xi(0), \sigma_0^2\right) \tag{2}$$

The initial distribution is a narrow Gaussian centered at the initial demonstration configuration $\xi(0)$ with a small standard deviation σ_0. The velocity has two components. The *demonstration velocity* is the velocity of the demonstration trajectory at time t, and is independent of q. This term serves to flow in the direction of the demonstration. The *stabilization term* is a negative proportional error feedback that corrects deviations from the demonstration trajectory. Controllers that stabilize around demonstration trajectories are known to reduce distribution shift and improve theoretical imitation learning guarantees [10]. We empirically observe that the stabilizing term produces significantly more robust and performant policies, compared to setting $k = 0$. We note that our framework leverages time derivatives of demonstration trajectories $\dot{\xi}(t)$ during training, which are easily accessible, in addition to $\xi(t)$. This is in contrast to conventional diffusion/flow policies that only use $\xi(t)$ but not $\dot{\xi}(t)$.

Theorem 1: The stabilizing conditional flow field given by Eqs. 1 and 2 induce the following per-timestep marginal distributions:

$$p_\xi(q \mid t) = \mathcal{N}\left(q \,\middle|\, \xi(t),\ \sigma_0^2 e^{-2kt}\right) \tag{3}$$

Proof: See App. I. The distribution of states sampled at any timestep $t \in [0,1]$ is a Gaussian centered at the demonstration trajectory $\xi(t)$. Furthermore, the standard deviation starts from σ_0 and decays exponentially with time at rate k.

IV. Learning Marginal Velocity Fields

Let $p_{\mathcal{D}}(h, \xi)$ denote the data generating distribution from which samples are drawn to form the training set. Using the conditional velocity fields $v_\xi(q,t)$ constructed in Sec. III as target, we train a neural network $v_\theta(q, t \mid h)$ using a finite-sample estimate of the conditional flow matching loss [3]:

$$\mathcal{L}_{\text{CFM}}(v_\theta, p_{\mathcal{D}}) \coloneqq \mathbb{E}_{(h,\xi)\sim p_{\mathcal{D}}}\ \mathbb{E}_{t\sim U[0,1]}\ \mathbb{E}_{q\sim p_\xi(q\,|\,t)} \left\| v_\theta(q, t \mid h) - v_\xi(q,t) \right\|_2^2 \tag{4}$$

This is simply an L_2 loss between the predicted velocity field $v_\theta(q, t \mid h)$ and the the conditional velocity field $v_\xi(q,t)$ as target. The expectation is over histories and trajectories in the training distribution $p_{\mathcal{D}}(h, \xi)$, time t sampled uniformly from $[0,1]$, and configuration q sampled from the constructed conditional flow known in closed-form in Eq. 3. The following theorem characterizes the per-timestep marginal distributions induced by the minimizer of this loss:

Theorem 2: The minimizer $v^\star \coloneqq \arg\min_v \mathcal{L}_{\text{CFM}}(v, p_{\mathcal{D}})$ induces the following per-timestep marginal distribution:

$$p^\star(q \mid t, h) = \int_\xi p_\xi(q \mid t)\, p_{\mathcal{D}}(\xi \mid h)\, \mathrm{d}\xi \tag{5}$$

Proof: This is a direct consequence of the flow matching theorems (Thms. 1 and 2) in Lipman et al. [3]. Intuitively, the per-timestep marginal distribution induced by the minimizer of $\mathcal{L}_{\text{CFM}}$ is the *average* of per-timestep marginal distributions of constructed conditional flows $p_\xi(q \mid t)$, over the distribution

		Push-T with state input			Push-T with image input		
		Average score ↑	Maximum score ↑	Latency per action ↓	Average score ↑	Maximum score ↑	Latency per action ↓
1	Diffusion policy: 100 DDPM steps [1]	90.7%	92.8%	40.2 ms	83.8%	**87.0%**	127.2 ms
2	Diffusion policy: 10 DDIM steps [1]	81.4%	85.3%	04.4 ms	80.8%	85.5%	10.4 ms
4	Streaming diffusion policy [14]	84.2%	87.0%	26.7 ms	80.5%	83.9%	77.7 ms
5	Streaming flow policy *without* stabilization	84.0%	86.4%	**03.5 ms**	73.9%	77.5%	**08.8 ms**
6	**Streaming flow policy (Ours)**	**92.3%**	**95.2%**	**03.5 ms**	**83.9%**	84.8%	**08.8 ms**

TABLE II: Imitation learning accuracy on the Push-T [1] dataset with state-based and image-based inputs, respectively. Our methods (in green) are compared against baselines (in red), and ablations (in blue) removing or changing one component of our method at a time. Average and maximum scores are computed across 10 best checkpoints, evaluating over 50 episodes [1].

of future trajectories in the training set $p_{\mathcal{D}}(\xi \mid h)$ that share the given observation history h.

Matching the per-timestep marginal distributions is a necessary condition for representing multi-modal distributions. Consider the example in Fig. 1(c, d) that constructs two conditional flows, one that samples states to the right ($q > 0$), and the other that samples states to the left ($q < 0$). In order for a learned model to sample both modes with probability 0.5 each, its per-timestep marginal distribution must match the averaged per-timestep marginal distributions of conditional flows. Unlike flow policies [2, 5, 6] that only require matching the target distributions at $t = 1$, our method leverages the fact that flow matching matches the marginal distributions over configurations at *all* timesteps $t \in [0, 1]$.

V. Experiments

We evaluate our method on the Push-T environment [1], a benchmark dataset for imitation learning. The dataset contains 200 training demonstrations, and 50 evaluation episodes. We compare against 3 baselines: (1) standard diffusion policy [1] that uses 100 DDPM [9] steps, (2) a faster version of diffusion policy that uses 10 DDIM [15] steps, and (3) Streaming Diffusion Policy [14], a recent method that runs diffusion policy in a streaming manner (see Sec. VI-C for details). Following Chi et al. [1], we report the best score across all checkpoints, and the average score for the 10 best checkpoints, and the latency per action for each method in Table II. We conclude from our initial experiments that stabilizing streaming diffusion policy performs no worse than diffusion policy while being significantly faster. Diffusion policy can be sped up by running fewer diffusion steps via DDIM [15], but only at the cost of often significant reduction in accuracy. We are currently working on scaling up to more imitation learning benchmark environments and real robot experiments.

VI. Related Work

A. Learning dynamical systems

Our work is closely related to prior work on learning dynamical systems [16–20]. A key difference is that most prior works assume a single, deterministic future trajectory given an input state. However, we focus on learning multi-modal *distributions* over trajectories, which is known to be essential for behavior cloning in robotics [1]. For example, Sochopoulos et al. [18] learn a neural ODE [12] by minimizing the distance between the predicted and demonstrated trajectories. This is susceptible to undesirable "averaging" of distinct behaviors that achieve the same task. Our approach learns a neural ODE endowed with an initial noise distribution that induces a distribution over future trajectories. This is also known as a continuous normalizing flow [12, 13]. We use the recently proposed flow matching framework [3] to fit the predicted trajectory distribution to training data. An important consequence is that our method aggregates "noised" demonstration trajectories to our training set, whereas prior works train only on states directly present in the demonstrations. Furthermore, most prior works assume a time-invariant velocity field [16–18]. Our velocity field depends on time and allows learning non-Markovian behaviors for tasks like spreading sauce on pizza dough, where the end-effector must rotate a fixed number of times. Finally, most works on learning dynamical systems focus on *point-stability* around goal states [16–18]; we don't assume goal states and construct flows that stabilize around demonstration trajectories.

B. Flow matching

Flow matching [3] is a recent technique for learning complex, multi-modal distributions that has been used to model images [3, 21, 22], videos [23, 24], molecular structures [25–27], and robot action sequences [2, 5, 6, 28, 29]. However, the flow sampling process starts from Gaussian noise, and the distribution of interest is only modeled at the final timestep. Our insight is to treat the entire flow trajectory as a sample from the target distribution over action sequences.

C. Streaming Diffusion Policy

The work most closely related to ours is Streaming Diffusion Policy [14], which is an adaptation of of discrete time diffusion policies [1]. Instead of maintaining all actions in the sequence at the same noise level, Streaming Diffusion Policy maintains a rolling buffer of actions with increasing noise levels. Every diffusion step reduces the the noise level of all actions by one, fully de-noising the oldest action in the buffer that can be streamed to the robot. However, this method requires maintaining an action buffer of the length of the prediction horizon, even if the action horizon is much shorter. Furthermore, there is an up-front cost to initialize the buffer with increasing noise levels. The rolling buffer approach has been applied to video prediction [30] and character motion generation [31]. Our method is more economical since it computes only as many actions are streamed to the robot, without requiring a buffer. We evaluate our method against Streaming Diffusion Policy in Sec. V.

References

[1] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea, July 2023.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control, 2024. *ArXiv Preprint:2410.24164*, 2024.

[3] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[4] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[5] Fan Zhang and Michael Gienger. Affordance-based Robot Manipulation with Flow Matching. *arXiv preprint arXiv:2409.01083*, 2024.

[6] Sean Ye and Matthew C Gombolay. Efficient trajectory forecasting and generation with conditional flow matching. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2816–2823. IEEE, 2024.

[7] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. NoMaD: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[8] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning (ICML)*, pages 2256–2265. PMLR, 2015.

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.

[10] Adam Block, Ali Jadbabaie, Daniel Pfrommer, Max Simchowitz, and Russ Tedrake. Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 48534–48547, New Orleans, USA, December 2023.

[11] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea, July 2023.

[12] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[13] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *International Conference on Learning Representations (ICLR)*, 2019.

[14] Sigmund H Høeg, Yilun Du, and Olav Egeland. Streaming Diffusion Policy: Fast Policy Synthesis with Variable Noise Diffusion Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, Atlanta, USA, May 2025.

[15] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

[16] S Mohammad Khansari-Zadeh and Aude Billard. Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2676–2683. IEEE, 2010.

[17] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics (T-RO)*, 27(5):943–957, 2011.

[18] Andreas Sochopoulos, Michael Gienger, and Sethu Vijayakumar. Learning deep dynamical systems using stable neural ODEs. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11163–11170. IEEE, 2024.

[19] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[20] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2003.

[21] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[22] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision (ECCV)*, pages 23–40. Springer, 2024.

[23] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for

efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024.

[24] Adam Polyak et. al. Movie Gen: A Cast of Media Foundation Models. *arXiv preprint arXiv:2410.13720*, 2025.

[25] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. AlphaFold meets flow matching for generating protein ensembles. *arXiv preprint arXiv:2402.04845*, 2024.

[26] Avishek Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. *arXiv preprint arXiv:2310.02391*, 2023.

[27] Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:59886–59910, 2023.

[28] Niklas Funk, Julen Urain, Joao Carvalho, Vignesh Prasad, Georgia Chalvatzaki, and Jan Peters. ActionFlow: Equivariant, Accurate, and Efficient Policies with Spatially Symmetric Flow Matching. *arXiv preprint arXiv:2409.04576*, 2024.

[29] Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5144–5151. IEEE, 2024.

[30] David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling Diffusion Models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 42818–42835. PMLR, 21–27 Jul 2024.

[31] Zihan Zhang, Richard Liu, Rana Hanocka, and Kfir Aberman. TEDi: Temporally-entangled diffusion for long-term motion synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

Appendix

Streaming Flow Policy

Simplifying diffusion/flow policies by treating robot trajectories as flow trajectories

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Kaelbling, Siddharth Ancha

Website: https://streaming-flow-policy.github.io†

APPENDIX I
PROOF OF THEOREM 1

Integrating learned velocity fields can suffer from drift since errors accumulate during integration. We introduce a stabilizing variant that actively corrects deviations from the demonstration trajectory. The stabilizing velocity field is:

$$v_\xi(q,t) = \underbrace{-k(q-\xi(t))}_{\text{Stabilization term}} + \underbrace{\dot{\xi}(t)}_{\text{Path velocity}} \tag{6}$$

where $k > 0$ is the stabilizing gain. This results in exponential convergence to the demonstration:

$$\frac{d}{dt}(q-\xi(t)) = -k(q-\xi(t)) \tag{7}$$

$$\implies \frac{1}{q-\xi(t)}\frac{d}{dt}(q-\xi(t)) = -k \tag{8}$$

$$\implies \frac{d}{dt}\log(q-\xi(t)) = -k \tag{9}$$

$$\implies \log(q-\xi(t))\Big|_0^t = -\int_0^t k dt \tag{10}$$

$$\implies \log\frac{q(t)-\xi(t)}{q_0-\xi(0)} = -kt \tag{11}$$

$$\implies q(t) = \xi(t) + (q_0-\xi(0))e^{-kt} \tag{12}$$

Since $q_0 \sim \mathcal{N}\left(\xi(0), \sigma_0^2\right)$, and $q(t)$ is linear in q_0, we have by linearity of Gaussian distributions that:

$$p_\xi(q \,|\, t) = \mathcal{N}(q \,|\, \xi(t), \sigma_0^2 e^{-2kt}) \tag{13}$$

□

APPENDIX II
DECOUPLING STOCHASTICITY VIA LATENT VARIABLES

In order to learn multi-modal distributions during training, streaming flow policy as introduced in Sec. III requires a small amount of Gaussian noise added to the start configuration. However, we wish to avoid adding noise to configurations at test time. We now present a variant of streaming flow policy in an extended state space by introducing a latent variable $z \in C$. The latent variable z decouples stochasticity from the flow trajectory, allowing us to sample multiple modes of the trajectory distribution at test time while deterministically starting the sampling process with the current robot configuration.

σ_0	Initial standard deviation	$\mathbb{R}_{\geq 0}$
σ_1	Final standard deviation	$\mathbb{R}_{\geq 0}$
σ_r	Residual std. deviation = $(\sigma_1^2 - \sigma_0^2)^{1/2}$	$\mathbb{R}_{\geq 0}$

TABLE III: Hyperparameters used in the stochastic variant of streaming flow policy that uses stochastic latent variables.

We now define a conditional flow in the extended state space $(q,z) \in \mathcal{C}^2$. We define the initial distribution by sampling q_0 and z_0 independently. q_0 is sampled from a vanishingly narrow Gaussian distribution centered at the initial configuration of the demonstration trajectory $\xi(0)$, but with a extremely small variance $\sigma_0 \approx 0$. z_0 is sampled from a standard normal distribution, similar to standard diffusion models [9] and flow matching [3].

$$z_0 \sim \mathcal{N}(0, I) \tag{14}$$

$$q_0 \sim \mathcal{N}(\xi(0), \sigma_0^2) \tag{15}$$

We assume hyperparameters σ_0 and σ_1 such that $\sigma_1 \geq \sigma_0$. They correspond to the initial and final standard deviations of the configuration variable q in the conditional flow. Let us define $\sigma_r := \sqrt{\sigma_1^2 - \sigma_0^2}$. Then we construct the joint flow trajectories of (q,z) starting from (q_0, z_0) as:

$$q(t \,|\, \xi, q_0, z_0) = q_0 + (\xi(t) - \xi(0)) + (\sigma_r t) z_0 \tag{16}$$

$$z(t \,|\, \xi, q_0, z_0) = (1 - (1-\sigma_1)t) z_0 + t\xi(t) \tag{17}$$

Intuitively, the variable q starts from q_0 and flows along the shape of the demonstration trajectory $\xi(t)$. However, it uses the sampled noise variable $z_0 \sim \mathcal{N}(0, I)$ to increase the standard deviation from σ_0 around $\xi(0)$ to σ_1 around $\xi(1)$. to sample different modes of the trajectory distribution at test time. On the other hand, the latent variable z starts from the random sample $z_0 \sim \mathcal{N}(0, I)$ but continuously moves closer to the demonstration trajectory $\xi(t)$, reducing its variance from 1 to σ_1. Since the flow is a diffeomorphism, we can invert it and express (q_0, z_0) as a function of $(q(t), z(t))$:

$$z_0 = \frac{z(t) - t\xi(t)}{1 - (1-\sigma_1)t} \tag{18}$$

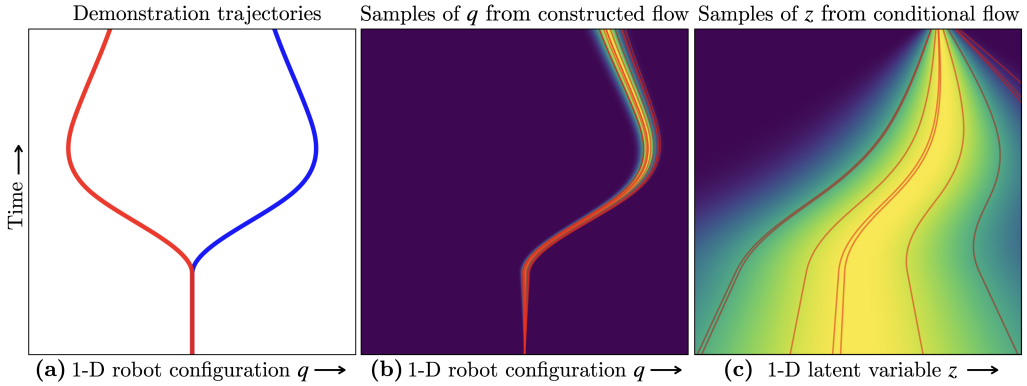

Fig. 2: Constructing a conditional flow using auxiliary stochastic latent variables instead of adding noise to robot configurations. In this toy example, the x-axis represents a 1-D configuration space, and the y-axis represents both trajectory time and flow time. **(a)** A toy bi-modal training set contains two trajectories shown in red and blue; the same as in Fig. 1a. Given a demonstration trajectory from the training set (*e.g.* the demonstration in blue), we design a velocity field $v(q, z, t \mid h)$ that takes as input time $t \in [0, 1]$, the configuration q at time t, as well as an additional latent variable z. The latent variable is responsible for injecting noise into the flow sampling process, allowing the initial robot configuration $q(0)$ to be deterministically set to the initial configuration of the demonstration. The latent variable $z(0) \sim \mathcal{N}(0, 1)$ is sampled from the standard normal distribution at the beginning of the flow process, similar to conventional diffusion/flow policies. The velocity field $v(q, z, t \mid h)$ generates trajectories in an extended sample space $[0, 1] \to \mathcal{C}^2$ where q and z are correlated and co-evolve with time. **(b, c)** Shows the marginal distribution of configurations $q(t)$ and the latent variable $z(t)$, respectively, at each time step. Overlaid in red are the q- and z- projections, respectively, of trajectories sampled from the velocity field. The configuration evolves in a narrow Gaussian tube around the demonstration, while the latent variable starts from $\mathcal{N}(0, 1)$ at $t = 0$ and converges to the demonstration trajectory at $t = 1$; see App. II for a full description of the velocity field.

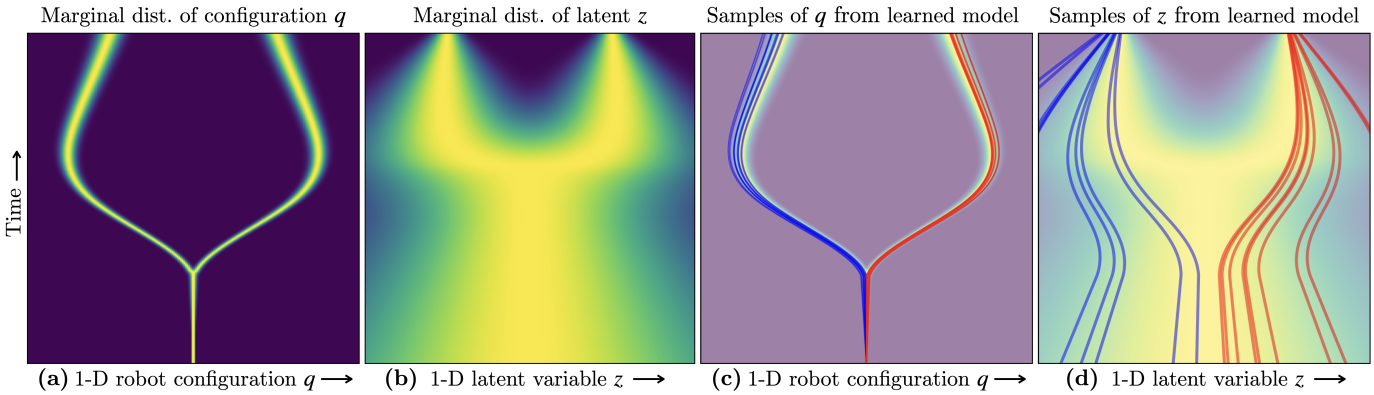

Fig. 3: The marginal velocity flow field $v_\theta(q, z, t \mid h)$ learned using the flow construction in Fig. 2. **(a, b)** shows the marginal distribution of configurations $q(t)$ and the latent variable $z(t)$, respectively, at each time step under the learned velocity field. **(c, d)** Shows the q- and z- projections, respectively, of trajectories sampled from the learned velocity field. By construction, $q(0)$ deterministically starts from the current robot configuration, whereas $z(0)$ is sampled from $\mathcal{N}(0, 1)$. Trajectories starting with $z(0) < 0$ are shown in blue, and those with $z(0) > 0$ are shown in red. The main takeaway is that in **(c)**, even though all samples deterministically start from the same initial configuration (*i.e.* the current robot configuration), they evolve in a stochastic manner that covers both modes of the training distribution. This is possible because the stochastic latent variable z is correlated with q, and the initial random sample $z(0) \sim \mathcal{N}(0, 1)$ informs the direction q evolves in.

$$q_0 = q(t) - (\xi(t) - \xi(0)) - (\sigma_r t) z_0 \tag{19}$$

Differentiating the flow equations with respect to t yields:

$$\dot{q}(t \mid \xi, q_0, z_0) = \dot{\xi}(t) + \sigma_r \, z_0 \tag{20}$$

$$\dot{z}(t \mid \xi, q_0, z_0) = \xi(t) + t\dot{\xi}(t) - (1 - \sigma_1) z_0 \tag{21}$$

To compute the corresponding velocity field, we substitute (q_0, z_0) in terms of (q, z) at time t.

$$v^q_\xi(q, z, t) = \dot{\xi}(t) + \frac{\sigma_r (z - t\xi(t))}{1 - (1 - \sigma_1) t} \tag{22}$$

$$v^z_\xi(q, z, t) = \xi(t) + t\dot{\xi}(t) - \frac{1 - \sigma_1}{1 - (1 - \sigma_1) t} (z - t\xi(t)) \tag{23}$$

Importantly, the evolution of q and z is inter-dependent *i.e.* the sample z_0 determines the evolution of q. Furthermore, the marginal probability distribution $p^q_\xi(q, t)$ has a simple form given by:

$$p_\xi(q \mid t) = \mathcal{N}\left(q \;\middle|\; \xi(t), \sigma_0^2 (1 - t^2) + \sigma_1^2 t^2\right) \tag{24}$$

In other words, q evolves in a Gaussian tube centered at the demonstration trajectory $\xi(t)$ with a standard deviation that varies from σ_0 at $t = 0$ to σ_1 at $t = 1$. The fact that the marginal distribution lies close to the demonstration trajectories, from Eq. 5 ensures that the per-timestep marginal distributions over configurations induced by the learned

velocity field are close to training distributions. However, this formulation allows us to select extremely small values of σ_0, essentially deterministically starting from the current configuration q_{curr}. The stochasticity injected by sampling $z_0 \in \mathcal{N}(0, I)$, as well as the correlated evolution of q and z ensures that we sample a diverse distribution of trajectories in q starting from the same configuration $q_0 = q_{\text{curr}}$. This phenomenon is illustrated via a 1-D toy example in Figs. 2 and 3, with details in captions.

APPENDIX III
CAVEATS

In this section, we discuss some limitations of our approach.

A. SFP does not match joint distribution, only per-timestep marginal distributions

Our flow matching framework ensures that the learned distribution over trajectories conditioned on the history matches the training distribution in terms of marginal distributions of configurations at each timestep $t \in [0, 1]$. We however, do not guarantee that the joint distribution of configurations within a trajectory matches that of the training distribution. This is in contrast to diffusion policy, that is able to match the joint distribution since diffuses over entire trajectories.

Figs. 4 and 5 illustrates a toy example where streaming flow policy matches marginal distributions but not the joint distribution. The x-axis represents 1-D robot configurations, and the y-axis represents flow time ($t \in [0, 1]$). Fig. 4a shows two trajectories in blue and red, of shapes "S" and "2" respectively. The trajectories intersect at $t = 0.5$. The learned flow field is shown in Fig. 4c, and the induced marginal distribution over configurations is shown in Fig. 4d. The marginal distribution of configurations matches the training distribution at each $t \in [0, 1]$. Trajectories sampled from the flow field are shown in Fig. 4d. The trajectory distribution contains two modes of equal probability: trajectories that always lie either in $q < 0$ (shown in blue), or in $q > 0$ (shown in red). The shapes formed by sampled trajectories — "Ɛ" and "3" respectively — do not match the shapes of trajectories in the training data.

A similar phenomenon is illustrated in Fig. 5 using the "stochastic" variant of streaming flow policy (see App. II) trained on the same dataset of intersecting trajectories. While the marginal distribution of configuration again matches with the training distribution, the trajectories contain *four* modes, with shapes "S", "2", "Ɛ" and "3". Due to stochasticity introduced by decoupled latent variables, trajectories in blue (alternatively, red) that start from $q < 0$ (alternatively, $q > 0$) are able to split in either direction at the point of intersection.

B. Streaming flow policies exhibit compositionality

While the loss of fidelity to the joint distribution of configurations can be interpreted as a limitation of streaming flow policy, another perspective is to think of our method as providing *compositionality* over training demonstrations. The sampled trajectories can be composed of pieces across the training data.

For many robotics tasks, compositionality might be both valid and desirable. For example, in quasi-static tasks where the robot moves slowly, if two demonstration trajectories are valid, then the compositions across these trajectories are often also valid. Under this assumption, compositionality allows the flow model to learn many valid combinations of partial trajectories with fewer demonstrations.

What constraints on trajectories reflected in the training data can streaming flow policy learn? Streaming flow policy is unable to capture global constraints that can only be represented in the joint distribution. However, it can learn certain local constraints.

C. SFPs can learn arbitrary position constraints

Robot configurations $q \in Q \subseteq \mathcal{C}$ may be constrained to lie in a subset $Q \subseteq \mathcal{C}$ of the configuration space. For example, Q may reflect joint limits of a robot arm. Then, a well-trained streaming flow policy should learn this constraint as well.

To see why, consider Eq. 5 which that states the learned marginal density of configurations $p^\star(q \mid t, h) = \int_\xi p_\xi(q \mid t)\, p_\mathcal{D}(\xi \mid h)\, \mathrm{d}\xi$ at time t is a weighted average of marginal densities of conditional flows $p_\xi(q \mid t)$. Recall that we construct $p_\xi(q \mid t)$ to be thin Gaussian tubes around demonstration trajectories ξ. Assume that the thickness of the Gaussian tube is sufficiently small that $q \notin Q \implies p_\xi(q \mid t) < \epsilon$, for some small $\epsilon > 0$ and for all ξ, t. Then we have from Eq. 5 that $p_\xi(q \mid t) < \epsilon \implies p^\star(q \mid t, h) < \epsilon$ for all $t \in [0, 1]$. Therefore, the probability of sampling a configuration q that violates the constraint Q is extremely low.

D. SFPs can learn convex velocity constraints

Theorem 2 of Lipman et al. [3] implies that the minimizer of the conditional flow matching loss $v^\star := \arg\min_v \mathcal{L}_{\text{CFM}}(v, p_\mathcal{D})$ has the following form:

$$
\begin{aligned}
v^\star(q, t \mid h) &= \int_\xi v_\xi(q, t)\, \underbrace{\frac{p_\mathcal{D}(\xi \mid h)\, p_\xi(q \mid t)}{\int_{\xi'} p_\mathcal{D}(\xi' \mid h)\, p_{\xi'}(q \mid t)\, \mathrm{d}\xi'}}_{p_\mathcal{D}(\xi \mid q, t, h)}\, \mathrm{d}\xi \\
&= \int_\xi v_\xi(q, t)\, p_\mathcal{D}(\xi \mid q, t, h)\, \mathrm{d}\xi \qquad (25) \\
&\approx \int_\xi \dot{\xi}(t)\, p_\mathcal{D}(\xi \mid q, t, h)\, \mathrm{d}\xi \ \text{(assuming } k \approx 0)
\end{aligned}
$$

Intuitively, the target velocity field $v^\star$ at (q, t) is a weighted average of conditional flow velocities $v_\xi(q, t)$ over demonstrations ξ. The weight for ξ is the Bayesian posterior probability of ξ, where the prior probability $p_\mathcal{D}(\xi \mid h)$ is the probability of ξ given h in the training distribution, and the likelihood $p_\xi(q \mid t)$ is the probability that the conditional flow around ξ generates q at time t.

Under sufficiently small values of k, we have from Eq. 1 that $v_\xi(q, t) \approx \dot{\xi}(t)$. Note that $v^\star$ is then a convex combination of demonstration velocities $\dot{\xi}(t)$. Consider convex constraints over velocities $\dot{\xi}(t) \in C$ *i.e.* $\dot{\xi}(t)$ is constrained to lie in a convex set C for all ξ with non-zero support $p_\mathcal{D}(\xi) > 0$ and for all $t \in [0, 1]$. This is the case, for example, when robot

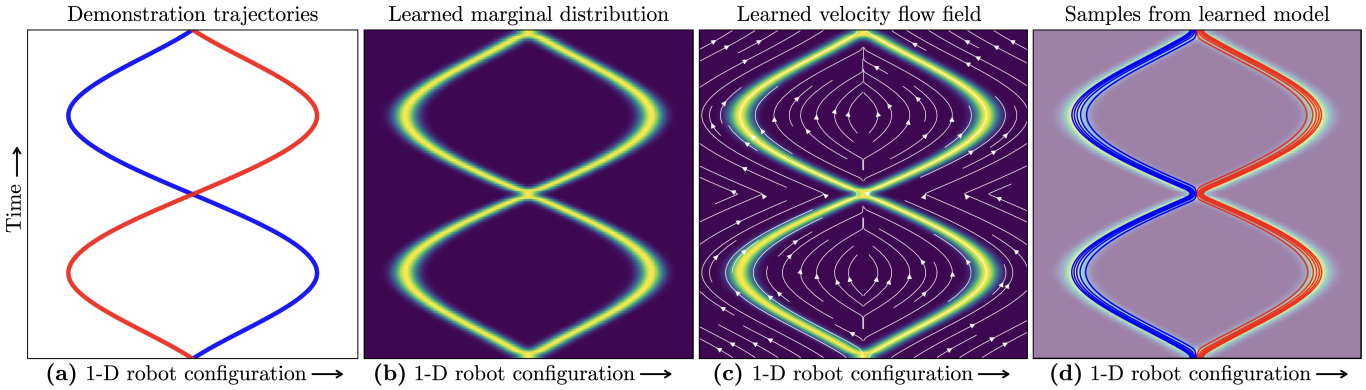

Fig. 4: A toy example illustrating how streaming flow policy matches marginal distribution of configurations in the trajectory at all time steps, but not necessarily their joint distribution. The x-axis represents a 1-D configuration space, and the y-axis represents both trajectory time and flow time. **(a)** The bi-modal training set contains two intersecting demonstration trajectories, illustrated in blue and red, with shapes "S" and "2" respectively. **(b)** The marginal distribution of configurations at each time step learned by our streaming flow policy. The marginal distributions perfectly match the training data. **(c)** The learned velocity flow field $v_\theta(q, t \mid h)$ that yeilds the marginal distributions in (b). **(d)** Trajectories sampled from the learned velocity field. Trajectories that start from $q < 0$ are shown in blue, and those starting from $q > 0$ are shown in red. The sampled trajectories have shapes "Ɛ" and "3", with equal probability. These shapes are different from the shapes "S" and "2" in the training distribution, although their margin distributions are identical.

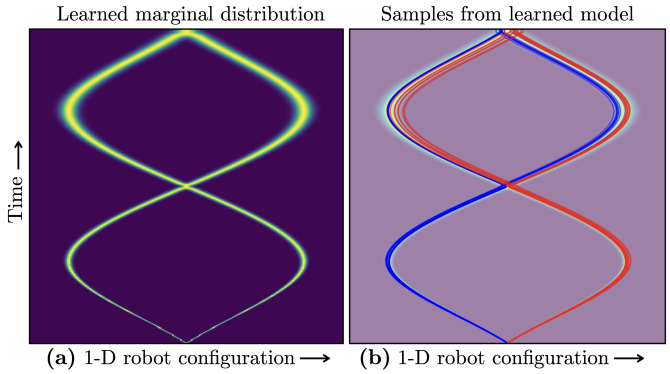

Fig. 5: Different variants of streaming flow policy can produce different joint distributions of configurations that are consistent with the marginal distributions in the training data. This example is produced using the stochastic version of streaming flow policy, described in App. II. **(a)** The marginal distribution of configurations at each time step learned by the stochastic streaming flow policy matches the training data. **(b)** Samples from the trained policy produces *four* modes with shapes "S", "2", "Ɛ" and "3", whereas the training data contains only two modes with shapes "S" and "2". Due to stochasticity introduced by decoupled latent variables, trajectories in blue (alternatively, red) that start from $q < 0$ (alternatively, $q > 0$) are able to split in either direction at the point of intersection.

joint velocities lie in a closed interval $[v_{\min}, v_{\max}]$. Then, Eq. 25 implies that $v^\star$ also lies in C.