Streaming Flow Policy

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

Massachusetts Institute of Technology

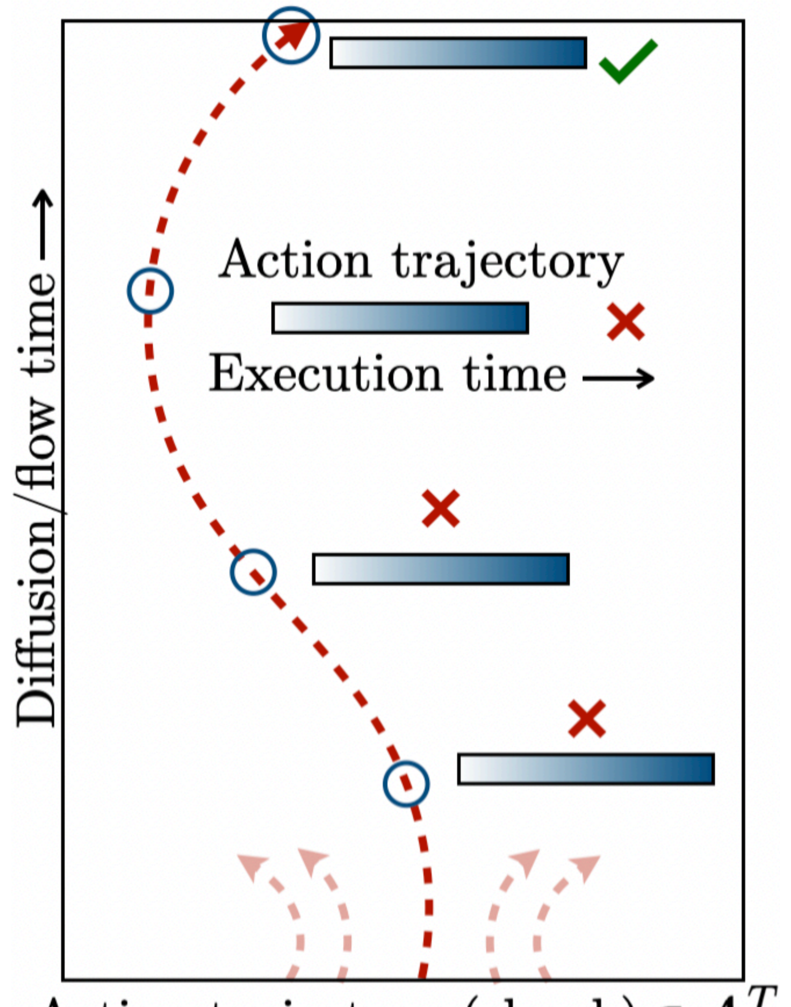
Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Kaelbling, Siddharth Ancha

CoRL 2025

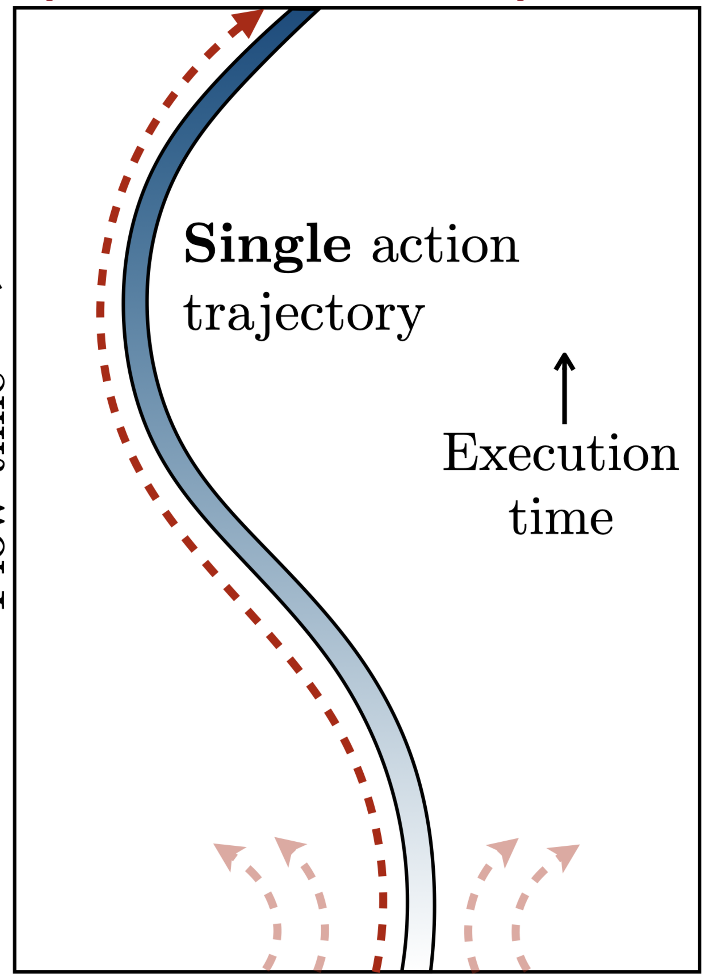
bit.ly/stflp

TLDR; SFP is a new imitation learning method that executes much faster than diffusion policy without reducing accuracy

Diffusion/flow policies are a "trajectory of trajectories"



Streaming flow policy: action trajectories *as* flow trajectories



Streaming flow policy learns a velocity field: $v_\theta(a, t \mid h)$

Produces an action trajectory (chunk) e.g. EE poses tracked by a controller.

Inputs: • h: Observation history • t: Scaled future timestep in [0,1] • a: Action

Generate trajectory:

Start with the most recently executed action a_{prev} predicted from the previous chunk.

Integrate velocity field from a_{prev} to produce future trajectory $a(t)$ for $t \in [0,1]$

$$a(t) = a_0 + \int_0^t v_\theta(a(s), s \mid h)\, ds \quad \text{where} \quad a_0 \sim \mathcal{N}(a_{\text{prev}}, \sigma_0^2)$$

✅ SFP: Can stream actions to controller *during* flow process.
DP: discards intermediate trajectories,
must wait for diffusion process to complete before executing actions.

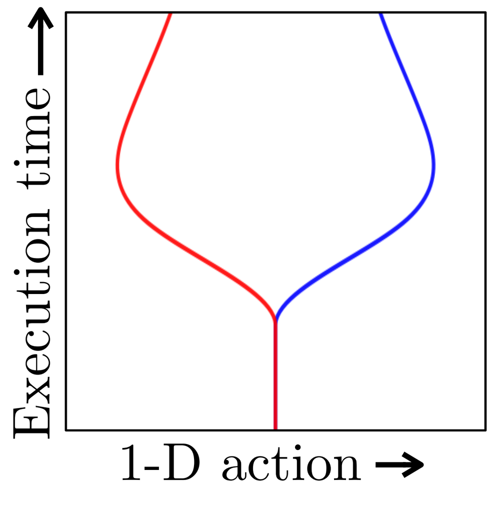
✅ In receding-horizon control where $T_{\text{test}} < T_{\text{train}}$ horizon:
SFP computes only as many as as needed. DP must produce all T_{train} actions.

✅ Minimal modification to DP architecture: only changes output layer dimension.

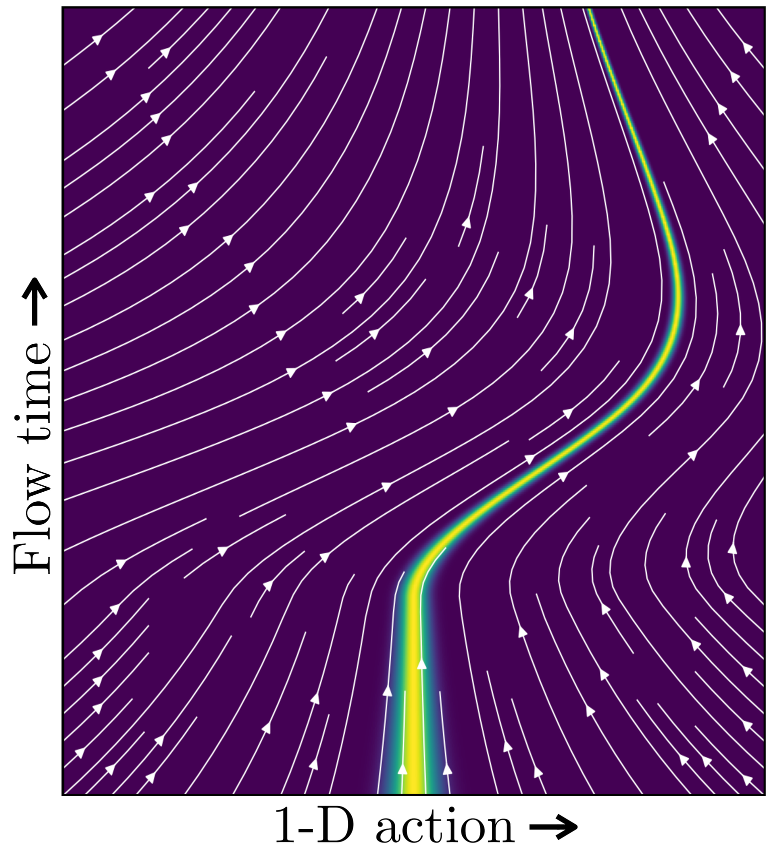
Construct a **conditional** flow around a given demonstration trajectory

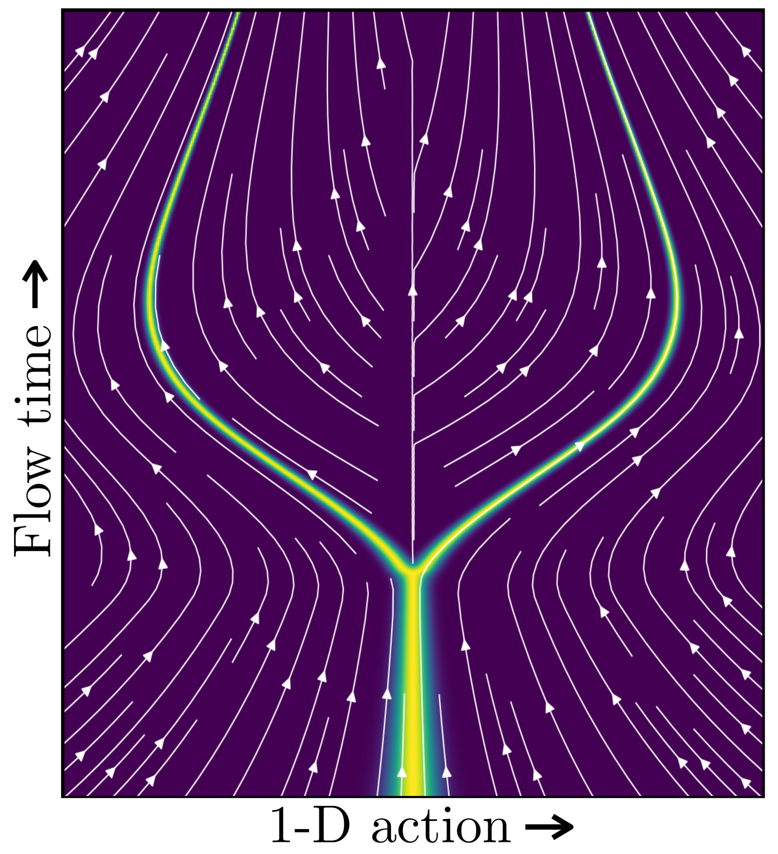
Learn **marginal** flow via flow matching

Toy dataset:
$a \in \mathbb{R}$
2 demos



$\xi : [0,1] \to \mathcal{A}$





Constructing conditional flow around demo ξ

Initial distribution at $t = 0$

$$p_\xi^0(a) = \underbrace{\mathcal{N}\left(a \mid \xi(0), \sigma_0^2\right)}_{\text{Gaussian centered at initial action}}$$

Constructed velocity field

$$v_\xi(a,t) = \underbrace{\dot{\xi}(t)}_{\text{Trajectory velocity}} - \underbrace{k(a - \xi(t))}_{\text{Stabilization term}^*}$$

*Low-level controllers that stabilize around demo trajectories reduce distribution shift and improve IL guarantees [2].

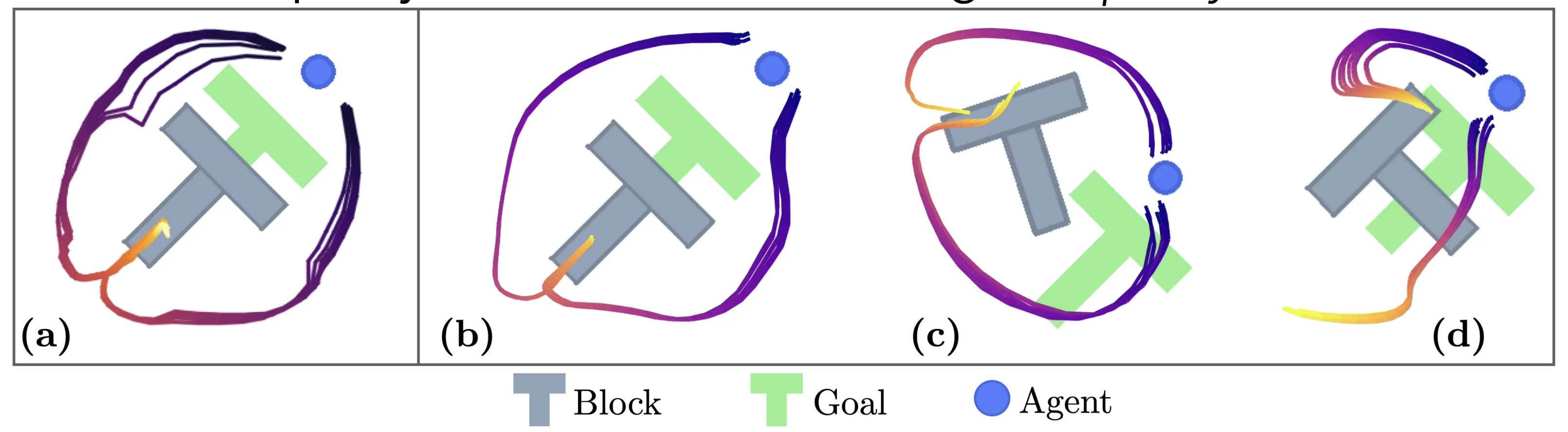
Distributions induced by velocity field: $p_\xi(a \mid t) = \mathcal{N}\left(a \mid \xi(t),\, \sigma_0^2 e^{-2kt}\right)$

Conditional flow matching loss: matches marginal action distributions

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

By flow matching theorem [1], **per-timestep marginal distribution** of learned flow field matches the training distribution.

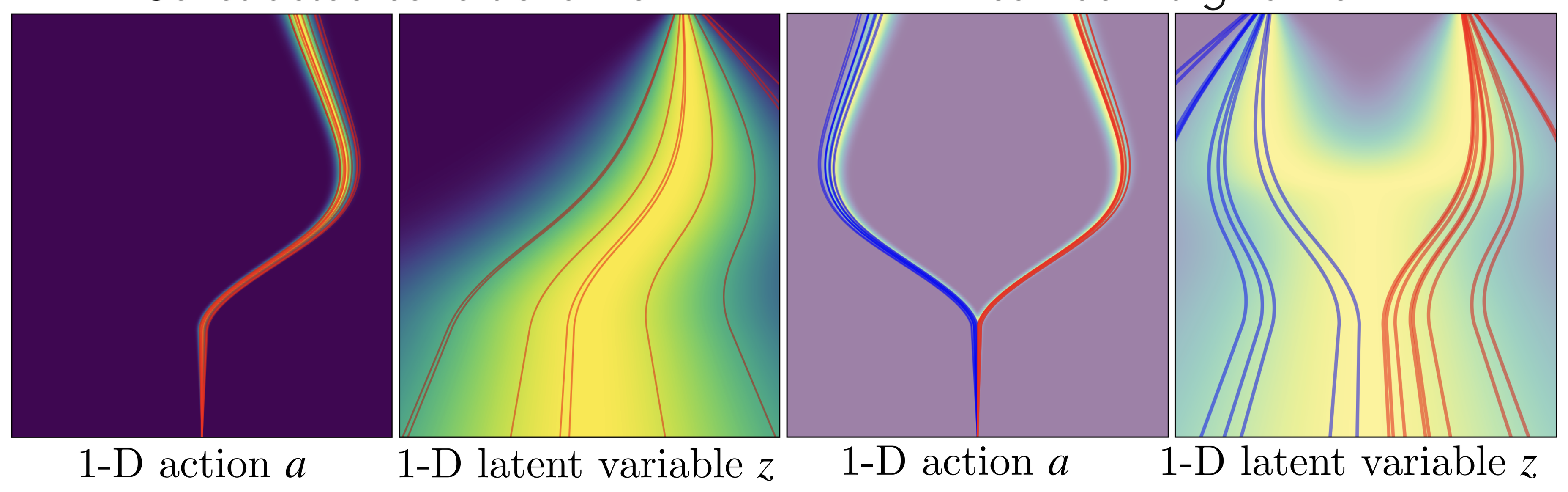
↳ ✅ SFP learns multimodal trajectory distributions!



Alternate formulations: Injecting noise via extra latent variables

We can be creative in applying the flow matching framework to action chunks!

Instead of adding noise to initial action, decouple it as a separate latent variable z. Then perform flow-matching in extended space $(a, z) \in \mathcal{A}^2$

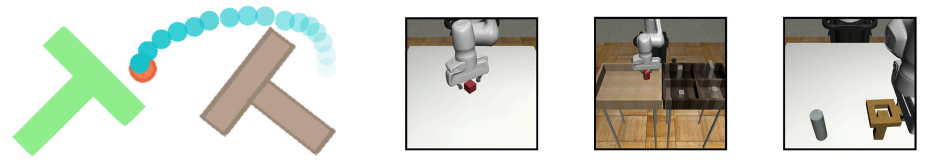


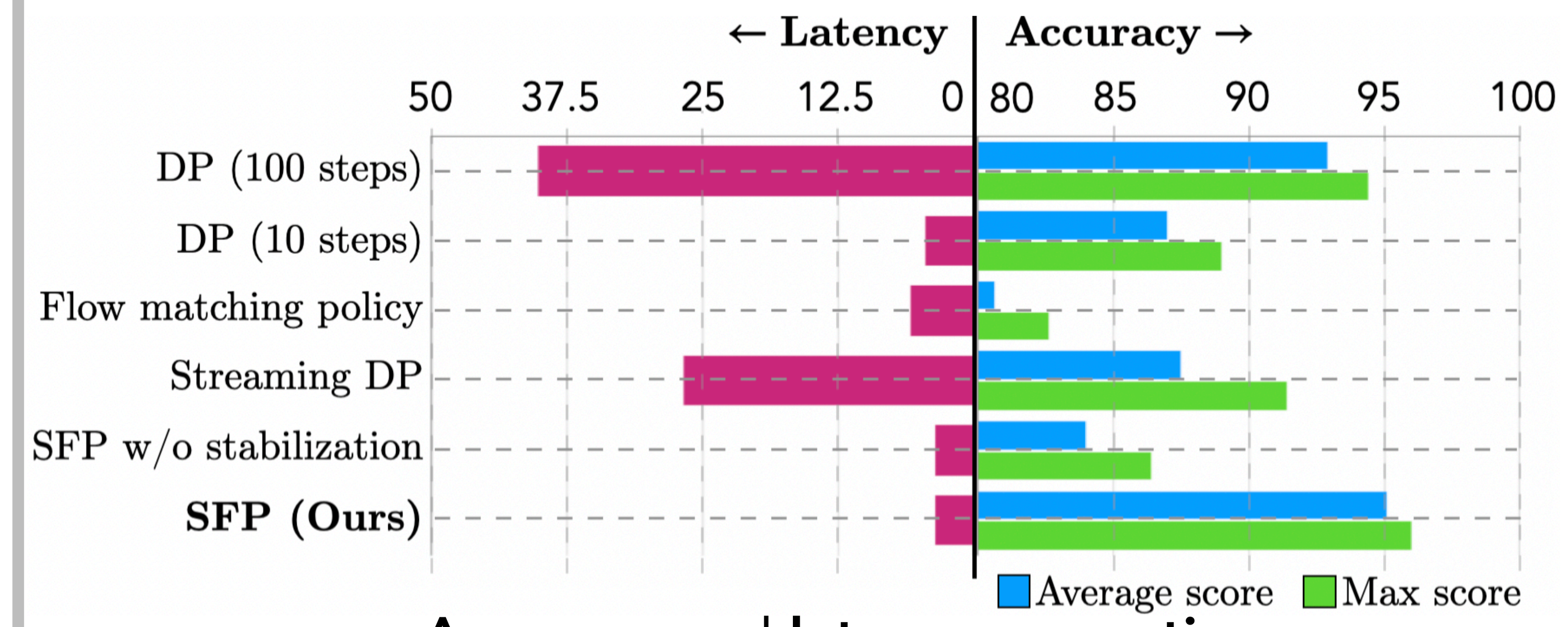
$z_0 \sim \mathcal{N}(0, I)$ $\quad a(t \mid \xi, a_0, z_0) = \xi(t) + (a_0 - \xi(0))\, e^{-kt} + (\sigma_r t) z_0$

$a_0 \sim \delta(\xi(0))$ $\quad z(t \mid \xi, a_0, z_0) = (1 - (1 - \sigma_1)t) z_0 + t\xi(t)$

} a and z co-evolve

Experiments on Push-T and RoboMimic

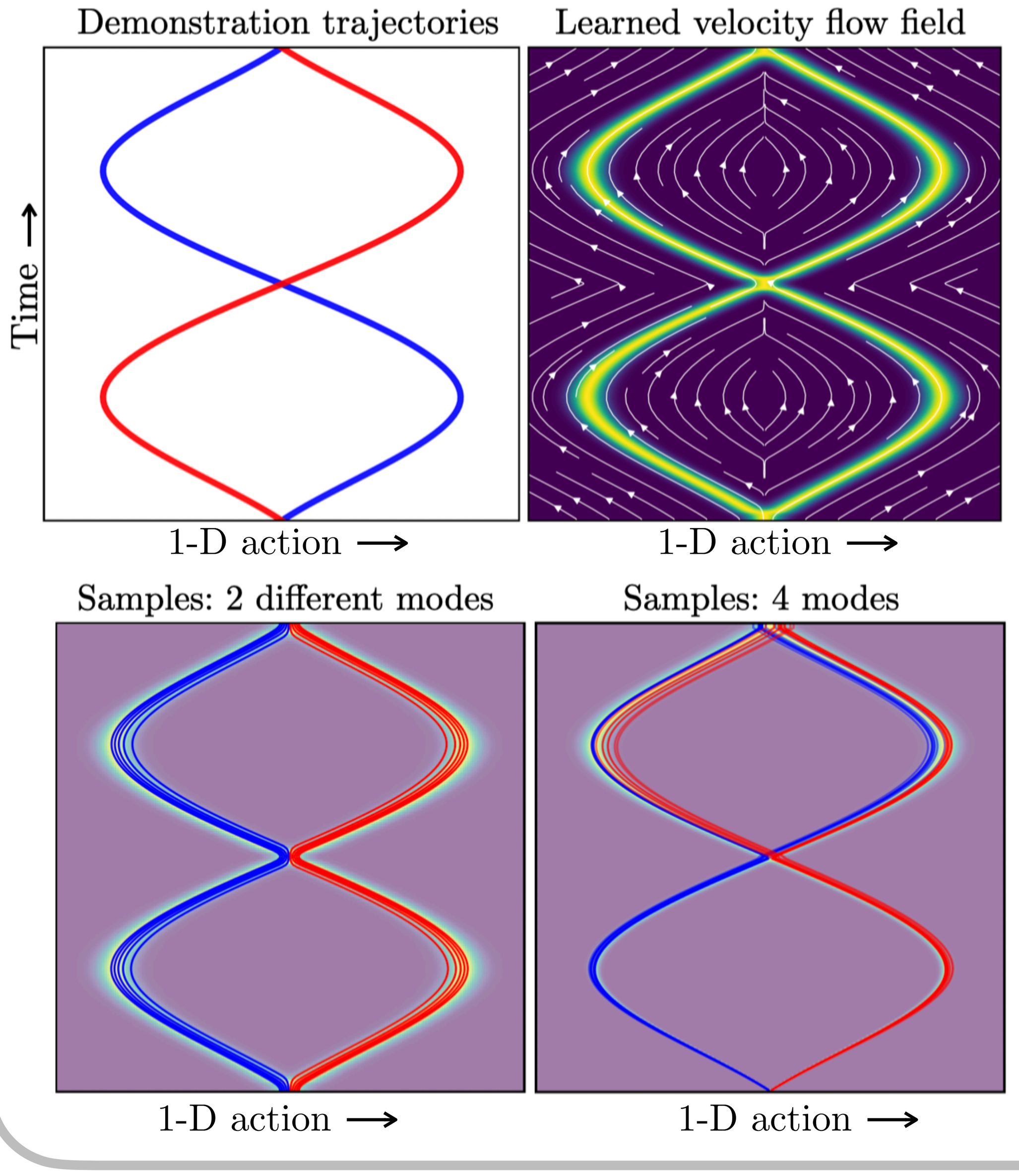




Accuracy and **latency per action**

Caveats

⚠ SFP only matches the *per-timestep marginal action distributions*, not joint distributions.



Training data contains two intersecting demonstration trajectories of shapes: (S, Ƨ)

Sampled trajectories are compositions with shapes: (3, Ɛ, S, Ƨ)

They match per-timestep marginal distribution but not the joint distribution.

[1] Lipman, Yaron, et al. "Flow matching for generative modeling." [2] Block, Adam, et al. "Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior."