SPEED ON DEMAND: FASTER-THAN-DEMONSTRATION IMITATION VIA A NEURAL ROBOT DYNAMICS MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Imitation policies trained on teleoperated demonstrations succeed on a broad range of manipulation tasks, but their execution speed is typically capped by the frequency at which human demonstrations are collected. Naively raising the execution frequency at deployment fails, as the tracking error between commanded and reached joint positions grows with frequency, so the executed trajectory deviates from the demonstration and the policy's observations fall out of distribution. We present **Speed on Demand**, a framework that enables a BC policy to operate faster than the demonstration by learning intentionally offset commands whose *reached* positions match the demonstrated trajectory after passing through the robot–controller dynamics. A neural dynamics model, trained using only 2 minutes of play data, captures the command-to-reached-state mapping of the robot–controller system. By fine-tuning the BC policy with this neural dynamics model in the loop, the policy learns to produce intentionally compensated commands whose resulting reached positions match the demonstrated trajectory under accelerated execution. A single policy, conditioned on a continuous speedup factor, executes at up to $5\times$ the demonstration frequency on CRANE-X7 with 80% task success, where both baselines fall to 27% or below. On Franka Research 3 with impedance control, the same method reaches $3.8\times$ the demonstration frequency, and task success is preserved up to $2.5\times$. Videos and code are available on the project website.

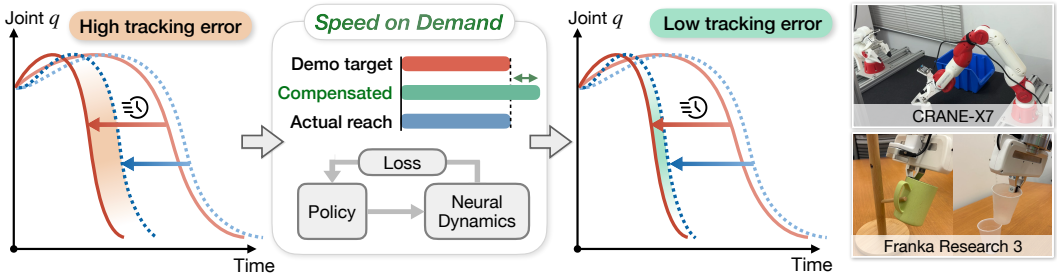


Figure 1: Speed on Demand. **Left**: naively sending action commands faster results in an increased tracking error between commanded action (red) and the actual reach (blue dash) at inference time, resulting in task failure. **Middle**: a neural dynamics model of the robot predicts the actual reach under acceleration during BC policy fine-tuning. The loss between predicted and demonstrated positions is backpropagated into the policy, which learns compensated commands that overshoot or undershoot the demonstrated target whose resulting reached positions match the demonstrated trajectory under accelerated execution. **Right**: the fine-tuned policy reaches the demonstrated trajectory at the raised rate.

1 INTRODUCTION

Imitation learning from teleoperated demonstrations has produced manipulation policies that can handle a broad range of tasks (Chi et al., 2023; Zhao et al., 2023; Black et al., 2025b). In standard deployment, however, execution speed is tied to the frequency at which demonstrations are collected. Human teleoperation is typically performed at roughly $50\,\mathrm{Hz}$, limiting the execution frequency when

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

the policy is deployed at the demonstration frequency. In principle, executing the same policy at 200 Hz could reduce task execution time by up to a factor of four and lower the cost-per-pick of an automation cell.

Naively increasing the execution frequency does not, in general, produce a faster realization of the demonstrated behavior. As the command frequency increases, the state actually reached by the robot increasingly deviates from the commanded state (Figure 1), causing the executed trajectory to diverge from the demonstration and task performance to degrade. This arises because imitation learning implicitly associates each policy command with the state transition observed during data collection, while the robot–controller system induces a different command-to-state mapping when executed at a higher frequency. Real systems exhibit finite-bandwidth execution dynamics arising from actuator response, mechanical dynamics, and low-level control, so the state reached by a command depends on the execution frequency as well as the recent motion history. Consequently, increasing the execution frequency changes not only how quickly commands are issued, but also how those commands are translated into the robot's actual motion. A successful speedup therefore requires accounting for this frequency-dependent command-to-state dynamics rather than simply issuing the learned commands more frequently.

Despite this issue, existing acceleration methods generally assume that the commanded trajectory can be faithfully realized at the target execution frequency. SAIL (Ranawaka Arachchige et al., 2025) addresses the resulting tracking challenge by relying on a high-frequency, high-gain low-level controller at inference time. This requires the control stack to compensate for the mismatch, rather than allowing the policy to account for the frequency-dependent command-to-state dynamics of the robot–controller system.

To overcome this challenge, we instead make the policy explicitly account for the frequency-dependent robot–controller dynamics by learning through a neural model of the system's command-to-reached-state mapping. A small neural model $\hat{g}_\phi$, conditioned on the recent joint state, the commanded action chunk, and the control period Δt, predicts the reached joint state of the target robot–controller system. The model therefore captures the effective execution dynamics of the entire stack, without requiring us to identify whether the tracking error originates from the actuators, mechanics, or low-level controller. We use $\hat{g}_\phi$ as a differentiable simulator inside the policy fine-tuning loop, minimizing the discrepancy between the predicted reached trajectory and the demonstrated trajectory. Gradients flow through the dynamics model to the policy, allowing it to learn commands that are intentionally offset from the demonstrated positions so that the *reached* positions match the demonstration after passing through the robot-controller dynamics. A frequency-conditioning input lets the same policy adapt its compensation to the desired execution frequency. We refer to this scheme as Speed on Demand.

We make three main contributions:

1. **Dynamics-aware policy acceleration.** We formulate faster-than-demonstration imitation as policy learning under frequency-dependent command-to-reached-state dynamics, rather than assuming faithful tracking at the target frequency.
2. **Neural robot dynamics model for action compensation.** We introduce a neural robot dynamics model that allows the policy to learn commands intentionally offset from the demonstrated positions so that the reached positions match the demonstration at higher execution frequencies.
3. **Continuous, on-demand speed control.** We condition the policy on a continuous execution speed and train it to produce appropriate action compensation across different execution frequencies. A single policy can therefore operate at multiple speeds without frequency-specific retraining. On the CRANE-X7, a single policy executes at up to $5\times$ the demonstration frequency with 80% task success, against 27% or below for both baselines. On the FR3, it accelerates two tasks by up to $3.8\times$ and preserves task success up to $2.5\times$.

2 Related Work

Imitation policies for manipulation Diffusion Policy (Chi et al., 2023) and ACT (Zhao et al., 2023) showed that predicting chunks of future actions, with a denoising diffusion model or a transformer trained as a conditional VAE, is effective for learning from teleoperated demonstrations. Later work used flow matching to predict action chunks in large vision-language-action models (Black et al.,

2025a;b), and real-time chunking kept the execution of these policies smooth under inference delay (Black et al., 2025c). Our imitation-learning stage trains a Diffusion Transformer (Peebles & Xie, 2023) with flow matching to predict action chunks and uses training-time real-time chunking (Black et al., 2025d) to keep consecutive chunks smooth. Our method starts from this behavior-cloned policy and fine-tunes it, conditioned on the desired speedup, through a learned dynamics model of the robot–controller system.

Faster-than-demo policy acceleration Several recent works execute imitation policies faster than the demonstrations. Most approaches accelerate the policy or its reference trajectory without explicitly modeling the frequency-dependent execution dynamics of the robot–controller system. RACE (Kim et al., 2026) uses desired robot states instead of commanded actions as imitation targets and re-times the predicted action chunks at inference to execute them as fast as the robot's physical limits allow. DemoSpeedup (Guo et al., 2025) down-samples demonstrations at rates that increase with the estimated action entropy and trains the policy on the accelerated demonstrations. It states that acceleration requires a high-precision robot controller and increases the gripper gain after finding that some gripper controllers failed to track at high speed. I2RLC (Yamane et al., 2026) instead incrementally accelerates demonstrations and refines their reference trajectories using tracking errors from repeated robot executions before training the imitation policy. B-spline Policy (Han et al., 2026) represents action trajectories as continuous B-spline curves, enabling smooth temporal scaling, while SpeedTuning (Yuan et al., 2025) uses reinforcement learning to learn an execution-speed policy on top of a base manipulation policy. Closest to our work is SAIL (Ranawaka Arachchige et al., 2025), which trains the policy to predict reached rather than commanded poses and tracks these targets with a high-gain controller at inference time. Its speedups therefore depend on a controller gain high enough to track the targets. On the CRANE-X7, raising the gain to the highest stable value made the arm oscillate around the commanded targets and did not raise task success at any speedup (Section 4.3). In contrast, we learn an explicit model of the frequency-dependent command-to-state dynamics of the robot–controller system and fine-tune the policy through this model, so that one frequency-conditioned policy learns compensating commands without per-demonstration refinement on the robot.

Learned dynamics models for robot control Learned dynamics models allow policy training to account for system behavior that is difficult to model analytically. Hwangbo et al. (2019) introduce learned actuator networks that predict joint torques from a history of joint position errors and velocities, capturing actuation and low-level control effects within a rigid-body simulator. Subsequent work incorporates learned actuator models into rough-terrain locomotion (Lee et al., 2020) and massively parallel policy training (Rudin et al., 2022). More recently, Fey et al. (2025) learn corrective torques for simulated arm actuators from real joint encoder data without torque labels, and deploy athletic loco-manipulation policies trained in the corrected simulator on a quadruped manipulator. These approaches improve the simulation dynamics used for reinforcement learning. Our approach instead learns a model of the command-to-reached-state mapping of the robot–controller system, conditioned on the control frequency, and backpropagates through this model to fine-tune an imitation policy against demonstrated joint positions rather than demonstrated actions. This trains the policy to compensate for frequency-dependent tracking errors during accelerated execution.

3 METHOD

3.1 PROBLEM FORMULATION

Consider a behavior cloning (BC) policy π_θ trained on demonstrations collected at a fixed control frequency f_{demo}. Let q_t denote the robot joint position and a_t the commanded joint target. At the demonstration frequency, the command-to-state transition is

$$q_{t+1} = g(q_{t-H:t}, a_t; f_{\mathrm{demo}}), \tag{1}$$

where g is the effective command-to-state transition of the robot–controller system at control frequency f, and H is the history length needed to characterize the dynamics. The BC policy reproduces the demonstrated commands from the observed state, $a_t = \pi_\theta(o_t)$, and implicitly relies on the transition dynamics encountered during data collection. At a higher control frequency $f > f_{\mathrm{demo}}$, the tracking error between the commanded target and the reached joint position grows, and the same

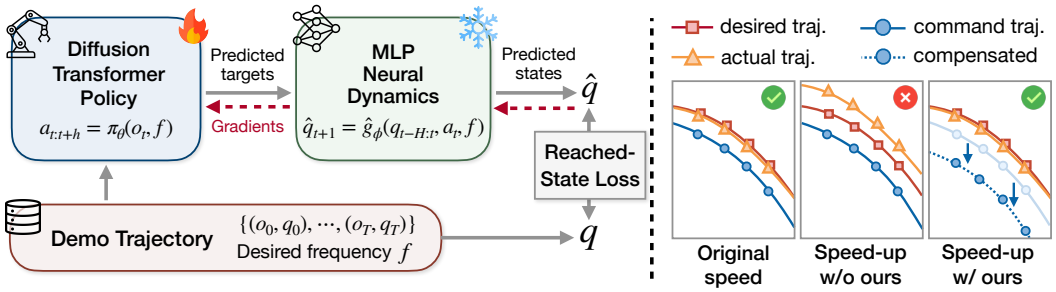


Figure 2: Dynamics-aware fine-tuning. **Left**: the trainable policy π_θ predicts a chunk of joint targets from the observation and the desired speedup. The frozen neural dynamics model $\hat{g}_\phi$ maps each target to the joint position the robot will reach at that control frequency. The reached-state loss $\mathcal{L}_{\text{reach}}$ between predicted and demonstrated joint positions is backpropagated through $\hat{g}_\phi$ into π_θ (red dashed arrows). **Right**: at the demonstration frequency, the reached trajectory (yellow triangles) follows the desired trajectory (red squares). At a higher frequency, the uncompensated command (blue circles) is tracked with a lag and the reached trajectory deviates. The fine-tuned policy commands targets offset from the desired trajectory, so the reached trajectory matches it again.

command no longer produces the demonstrated transition:

$$q_{t+1} = g(q_{t-H:t}, a_t; f) \neq g(q_{t-H:t}, a_t; f_{\text{demo}}). \tag{2}$$

Executing the BC policy faster therefore does not yield a time-scaled version of the demonstrated trajectory. The robot drifts from the demonstrated states, the policy encounters observations absent from the demonstrations, and the resulting distribution shift grows with the execution frequency.

Our goal is a policy trained at f_{demo} that executes the demonstrated behavior at a higher frequency f while preserving the demonstrated reached-state trajectory. Given the demonstrated joint position q_{t+1}^{demo}, we seek a compensated command a_t such that

$$g(q_{t-H:t}, a_t; f) \approx q_{t+1}^{\text{demo}}. \tag{3}$$

The commanded target thus need not coincide with the desired joint position. The policy instead compensates for the execution dynamics so that the reached position matches the demonstration.

3.2 Neural Robot Dynamics Model

To account for this change in dynamics, we learn a neural model of the robot–controller system from two minutes of play data. Given a history of joint positions, a commanded joint target, and the control frequency, the model predicts the joint position reached at the next timestep:

$$\hat{q}_{t+1} = \hat{g}_\phi(q_{t-H:t}, a_t, f). \tag{4}$$

The joint-position history carries the current motion state, and conditioning on f lets the model capture the frequency-dependent dynamics. We collect the play data at $50\,\text{Hz}$ and replay the recorded command sequences on the robot at 50, 100, and $150\,\text{Hz}$, which yields commanded targets and the corresponding reached joint positions at each frequency, up to $3\times$ the demonstration frequency. The model minimizes the squared prediction error

$$\mathcal{L}_{\text{dyn}} = \|\hat{q}_{t+1} - q_{t+1}\|_2^2. \tag{5}$$

Once trained, $\hat{g}_\phi$ is frozen and used as a differentiable simulator during policy fine-tuning (Figure 2).

3.3 Behavior Cloning

We first train π_θ at the demonstration frequency f_{demo}. The policy is a Diffusion Transformer (DiT) (Peebles & Xie, 2023) trained with flow matching to predict the velocity field of the demonstrated action distribution. Smooth action trajectories matter for faster execution, because abrupt changes between consecutive action chunks induce large accelerations and tracking errors as the

control frequency increases. We therefore adopt TT-RTC (Black et al., 2025d), which keeps the transitions between consecutive chunks smooth.

Let A_1 be a demonstrated action chunk and $A_0 \sim p_0$ a sample from a standard Gaussian prior. With the interpolation $A_\tau = (1-\tau)A_0 + \tau A_1$ for $\tau \sim \mathcal{U}(0,1)$, the DiT predicts the velocity field $v_\theta(A_\tau, \tau, o_t)$ and is trained with the flow matching objective

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{A_0, A_1, \tau}\left[\left\|v_\theta(A_\tau, \tau, o_t) - (A_1 - A_0)\right\|_2^2\right]. \tag{6}$$

This policy reproduces the demonstrated behavior with temporally smooth chunks but does not account for the execution dynamics at higher control frequencies. It is the initialization for the fine-tuning stage.

3.4 Dynamics-Aware Fine-Tuning

Starting from the BC policy, we fine-tune π_θ to output commands that are offset from the demonstrated positions so that the reached positions match the demonstration at higher control frequencies (Figure 2). The frozen dynamics model $\hat{g}_\phi$ from Section 3.2 predicts where the robot will reach when a command is executed at frequency f. For an observation o_t, the policy predicts a command $a_t = \pi_\theta(o_t, f)$ conditioned on the desired frequency, and the dynamics model predicts the resulting reached position:

$$\hat{q}_{t+1} = \hat{g}_\phi(q_{t-H:t}, a_t, f). \tag{7}$$

In practice the policy outputs an action chunk $\hat{a}_{t:t+T_c}$ of T_c steps, recovered from the predicted velocity field by integrating the flow from the noised chunk to $\tau = 1$, and we apply the dynamics model to every chunk step in parallel with the demonstrated joint positions as the input history. Feeding the demonstrated history rather than the model's own predictions (teacher forcing) keeps the input of $\hat{g}_\phi$ on the states it was trained on and prevents prediction errors from compounding across the chunk. Rather than matching the policy output to the demonstrated action, we penalize the predicted reached state against the demonstrated joint position q_{t+1}^{demo}:

$$\mathcal{L}_{\mathrm{reach}} = \left\|\hat{q}_{t+1} - q_{t+1}^{\mathrm{demo}}\right\|_2^2, \tag{8}$$

averaged over the chunk steps after the TT-RTC prefix and excluding the gripper joint, since $\hat{g}_\phi$ is trained on free-space play data without contact. Because $\hat{g}_\phi$ is differentiable, the gradient reaches the policy through the predicted dynamics:

$$\nabla_\theta \mathcal{L}_{\mathrm{reach}} = \frac{\partial \mathcal{L}_{\mathrm{reach}}}{\partial \hat{q}_{t+1}} \frac{\partial \hat{q}_{t+1}}{\partial a_t} \frac{\partial a_t}{\partial \theta}. \tag{9}$$

The policy therefore learns to command targets that, after the execution dynamics at frequency f, reach the demonstrated positions. When the robot lags behind its command at a higher frequency, for example, the policy learns to command a target ahead of the demonstrated position.

We condition the policy on the continuous speedup factor $r = f/f_{\mathrm{demo}}$ and sample f for each training batch from $\{50, 100, 150, 200\}$ Hz, so a single policy adapts its commands to the desired speed without per-frequency fine-tuning.

4 Experiments

We evaluate **Speed on Demand** along four questions:

1. **Speedup:** Can a policy trained at the demonstration frequency achieve faster execution by compensating for robot–controller dynamics?
2. **Task performance:** Can dynamics-aware compensation preserve performance as the execution frequency increases?
3. **Generalization:** Does the approach transfer across different robot–controller systems?
4. **Action compensation:** Does the learned policy exhibit the predicted dynamics-aware compensation?



(a) CRANE-X7 object sorting (b) FR3 cup insertion (c) FR3 mug hanging

Figure 3: The three evaluation tasks.

4.1 Experimental Setup

We evaluate **Speed on Demand** on two distinct robot–controller systems (Figure 3). The first is the CRANE-X7, a low-cost robotic arm equipped with Dynamixel actuators and position control. The second is the Franka Research 3 (FR3), a higher-performance robotic arm with an impedance controller. We evaluate object sorting on the CRANE-X7 and cup insertion and mug hanging on the FR3. These systems differ substantially in their hardware and low-level control dynamics, allowing us to evaluate whether the proposed approach can account for different command-to-reached-state dynamics.

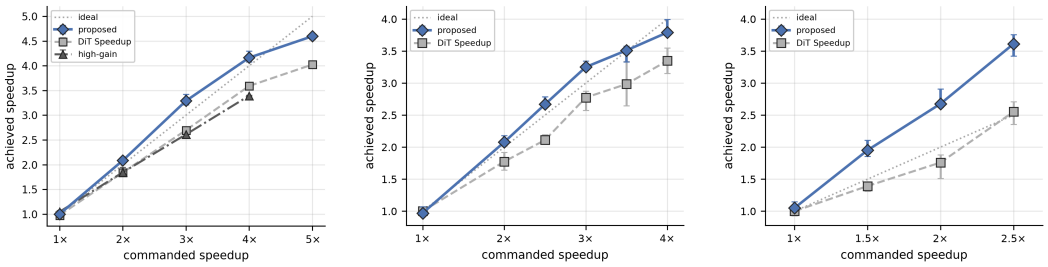
For each task, we collect demonstrations at a fixed demonstration frequency, with initial object positions varied by approximately $2\,\mathrm{cm}$. We collect 50 demonstrations for the CRANE-X7 object-sorting task and 30 demonstrations for each of the two FR3 tasks. We train a behavior cloning policy using only demonstrations collected at the corresponding demonstration frequency, and evaluate the same policy at higher control frequencies without collecting additional demonstrations at the target frequencies. For each robot–controller system, the neural dynamics model is trained using approximately two minutes of play data collected on the corresponding system. The desired speedup is provided to the policy as a continuous conditioning variable, allowing the same policy to adapt its commands across control frequencies.

We evaluate both execution speed and task performance. We measure the achieved speedup relative to the execution time at the demonstration frequency. For the CRANE-X7 we report the whole-task speedup, gripper motion included. For the FR3 we report the arm-only speedup, because the FR3 gripper cannot be accelerated: during evaluation the policy runs at $50\,\mathrm{Hz}$ while the gripper is moving and at the commanded frequency otherwise, and the speedup is measured over the arm-motion segments only. Task performance is evaluated using the success rate over 15 rollout episodes at each control frequency.

We compare against two baselines that represent alternative approaches to faster execution. **DiT Speedup** directly executes the behavior-cloned DiT policy with TT-RTC at higher control frequencies, without adapting the policy to the changed execution dynamics. **DiT High-Gain** executes the same policy with a higher low-level control gain to improve tracking at higher frequencies, following the controller-side strategy used by SAIL (Ranawaka Arachchige et al., 2025). On the CRANE-X7, we raise the Dynamixel position P gain from its default of 800 to 900, the highest value at which the arm remained stable at the demonstration frequency. We use this component of SAIL as a baseline because our focus is specifically on the command-to-reached-state dynamics mismatch induced by changing the execution frequency, rather than reproducing the full SAIL system, which additionally incorporates action inference, adaptive speed modulation, and action scheduling.

4.2 Faster-than-Demonstration Execution

We first evaluate whether **Speed on Demand** can accelerate policy execution beyond the demonstration frequency. Figure 4a shows the achieved whole-task speedup as a function of the commanded speedup for the CRANE-X7. The ideal speedup corresponds to proportional acceleration of the robot motion as the control frequency is increased.



(a) CRANE-X7 object sorting (b) FR3 cup insertion (c) FR3 mug hanging

Figure 4: Achieved speedup as a function of the commanded speedup on (a) CRANE-X7 object sorting (whole task) and (b) FR3 cup insertion and (c) FR3 mug hanging (arm motion only, see Section 4.1). The dotted line is the ideal proportional speedup. Markers are medians and error bars interquartile range over the successful rollouts at each rate ($n \leq 15$).

Table 1: Success rate (%) over 15 rollouts at each control rate. Demonstrations were collected at 50 Hz for every task. Best method per task and rate in bold; dashes mark rates that were not evaluated.

	CRANE-X7, object sorting			FR3, cup insertion		FR3, mug hanging	
Rate (Hz)	DiT Speedup	DiT High-Gain	Ours	DiT Speedup	Ours	DiT Speedup	Ours
50 (1×)	**100**	73	**100**	**100**	**100**	87	**100**
75 (1.5×)	–	–	–	–	–	60	**100**
100 (2×)	87	60	**93**	**93**	**93**	47	**87**
125 (2.5×)	–	–	–	87	**93**	20	**47**
150 (3×)	40	20	**87**	**93**	47	–	–
175 (3.5×)	–	–	–	**87**	47	–	–
200 (4×)	13	13	**87**	**93**	33	–	–
250 (5×)	27	0	**80**	–	–	–	–

DiT Speedup falls increasingly short of the commanded speedup as the execution frequency increases, reaching about $4.0\times$ at a commanded $5\times$. This reflects the mismatch between the command-to-reached-state dynamics at the demonstration and execution frequencies. DiT High-Gain does not close this gap: despite its higher controller bandwidth, its achieved speedup matches DiT Speedup up to $4\times$. **Speed on Demand** tracks the commanded speedup up to $4\times$ and reaches about $4.6\times$ at a commanded $5\times$.

To examine whether this behavior extends beyond the specific dynamics of Dynamixel-based position control, we repeat the experiment on the FR3 with joint impedance control. Unlike the CRANE-X7, the FR3 has more high-torque actuator and its impedance control provides unique dynamics not seen in position control. Figure 4b shows the achieved speedup across control frequencies. **Speed on Demand** stays close to the ideal up to $4\times$, reaching about $3.8\times$, while DiT Speedup reaches about $3.4\times$. On the mug-hanging task (Figure 4c), **Speed on Demand** reaches $3.6\times$ at a commanded $2.5\times$, while DiT Speedup reaches about $2.6\times$.

4.3 Task Performance Under Speedup

Increasing the execution frequency is only useful if the policy can maintain task performance. We therefore evaluate the task success rate of each method across increasing control frequencies. Table 1 reports the number of successful rollouts out of 15 at each frequency.

On the CRANE-X7, DiT Speedup degrades from 87% at $2\times$ to 40% at $3\times$ and 13% at $4\times$. DiT High-Gain does not help: its success rate is equal to or below that of DiT Speedup at every rate. The higher gain made the arm oscillate around its targets, and this oscillation degraded task performance more than the tracking lag it was meant to remove. **Speed on Demand** stays at 87% or higher up to $4\times$ and reaches 80% at $5\times$, where both baselines are at or below 27%.

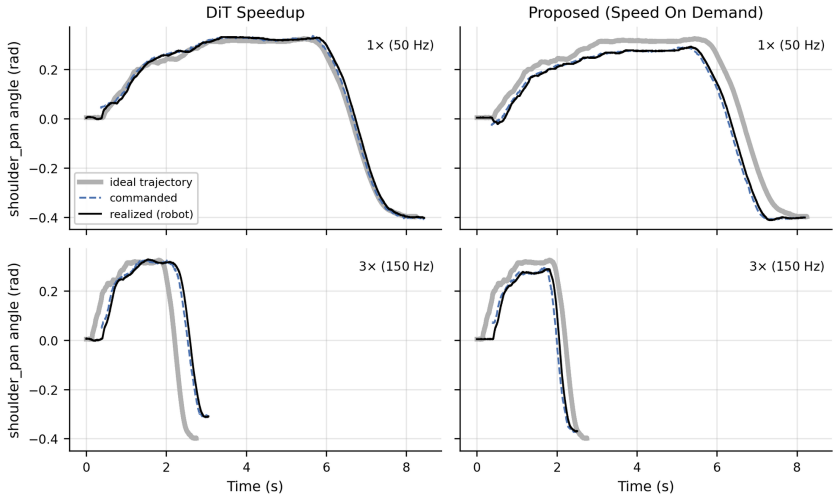


Figure 5: Angle of one joint of the CRANE-X7 during one rollout of DiT Speedup (left) and Speed on Demand (right) at $1\times$ (top) and $3\times$ (bottom). Gray: reference trajectory, the median-duration $1\times$ rollout of Speed on Demand with time divided by the commanded speedup. Dashed: commanded target. Black: realized joint position.

On the FR3 with impedance control, the picture is mixed. The speedup results of Section 4.2 carry over: **Speed on Demand** accelerates both FR3 tasks, and on mug hanging it also keeps a higher success rate than DiT Speedup at every rate (87% versus 47% at $2\times$, 47% versus 20% at $2.5\times$). On cup insertion, however, its success rate drops from 93% at $2.5\times$ to 47% at $3\times$ and 33% at $4\times$, while DiT Speedup stays between 87% and 93%. In the failed rollouts, the policy outputs over-compensated commands whose realized trajectories deviate from the demonstration at higher frequencies. We attribute this to the dynamics model: its history window of three steps is too short to capture the dynamics of the impedance controller, so its next-state predictions are inaccurate at high rates and the fine-tuning loss trains the policy against a poor teacher. A dynamics model with a longer history window is future work. Together with the speedup results, these results show that the proposed method increases execution speed on all three tasks and preserves task performance on the CRANE-X7 and on FR3 mug hanging, but not on FR3 cup insertion above $2.5\times$.

4.4 Dynamics-Aware Action Compensation

The quantitative results show that **Speed on Demand** accelerates execution and, on the CRANE-X7, preserves task performance. We next examine whether the policy achieves this by learning the intended dynamics-aware action compensation. Figure 5 visualizes representative rollouts at $1\times$ and $3\times$ the demonstration frequency on the CRANE-X7. We compare the commanded and realized trajectories with a reference trajectory, the median-duration $1\times$ rollout of Speed on Demand with time divided by the commanded speedup. We use a rollout rather than a demonstration as the reference because the demonstrations contain pauses from VR teleoperation that a $1\times$ rollout does not reproduce, and because TT-RTC makes the rolled-out trajectory smoother than the demonstration.

For DiT Speedup, the commanded trajectory closely follows the desired trajectory, while the realized trajectory deviates from it during rapid motion. In particular, the robot lags behind the desired trajectory during the transition, reflecting the command-to-reached-state mismatch at the higher control frequency.

In contrast, **Speed on Demand** produces commands that are intentionally offset from the desired trajectory. After passing through the robot's execution dynamics, these commands result in a realized trajectory that closely follows the desired trajectory. This qualitative observation demonstrates that the policy does not simply reproduce the demonstrated commands at a higher frequency, but instead learns to modify its commands to compensate for the execution dynamics.

Figure 6 quantifies this offset across commanded speedups. For each observation in the demonstrations (every $0.5\,\mathrm{s}$ of all 50 episodes, with two policy samples per observation, 980 samples in total), we project the offset of the commanded action from the demonstrated action onto the demonstrated direction of motion and average it over the moving steps of the 64-step chunk. The fine-tuned policy's offset grows from about zero at $1\times$ to about $55\,\mathrm{mrad}$ at $3\times$ and $60\,\mathrm{mrad}$ at $4\times$, while the base policy stays near zero at every speedup. The policy is trained on frequencies up to $3\times$, so the increasing offset up to $3\times$ reflects learned in-domain compensation, while the offset observed at $4\times$ shows that this behavior extends beyond the training range. This result quantitatively shows that fine-tuning teaches the policy to output compensated actions whose offset increases with speedup, consistent with the increasing tracking error at higher frequencies.

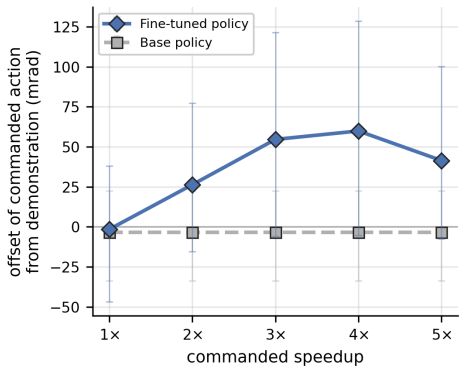


Figure 6: Offset of the commanded action from the demonstration along the demonstrated direction of motion on the CRANE-X7 (medians and interquartile ranges). The dynamics model used for fine-tuning is trained up to $3\times$ the demonstration frequency; thus, $4\times$ and $5\times$ lie outside its training range.

5 Limitations

Environmental dynamics are out of scope $\hat{g}_\phi$ captures the robot's intrinsic servo and controller response in free space. Held objects with non-negligible inertia, sustained contact, and aerodynamic effects are not modeled. We assume that, on sufficiently massive arms, the robot's own dynamics dominate object-induced perturbations. Quantifying this across heavier payloads is future work.

Short history window of the dynamics model $\hat{g}_\phi$ conditions on three past joint positions. This suffices for the position-controlled CRANE-X7, but on the impedance-controlled FR3 the fine-tuned policy over-compensates on cup insertion at $3\times$ and above and success drops (Table 1). A dynamics model with a longer history window is future work.

Extrapolation beyond the dynamics model's training range The neural dynamics model is trained only up to $3\times$ the demonstration frequency, while the policy is evaluated at higher frequencies. Although the model enables useful extrapolation in our experiments, its predictions are not guaranteed to remain accurate outside the training range. Improving dynamics-model extrapolation to substantially higher execution frequencies is an important direction for future work.

6 Conclusion

We presented *Speed on Demand*, a dynamics-aware approach for executing imitation policies faster than demonstrations. The key idea is to explicitly model the frequency-dependent command-to-reached-state dynamics of the robot–controller system as a neural network and use this model during policy fine-tuning. Rather than assuming that commanded targets are faithfully tracked at higher frequencies, the policy learns to intentionally compensate for execution errors so that the *reached* states remain consistent with the demonstrated behavior.

We demonstrated that a single speed-conditioned policy can adapt its commands across multiple execution frequencies without per-frequency retraining. Experiments on two robot platforms with different control systems show that the approach accelerates execution on all three tasks and maintains task performance on the CRANE-X7 and on FR3 mug hanging, while on FR3 cup insertion success drops above $2.5\times$, which we attribute to the short history window of the dynamics model. These results suggest that explicitly accounting for robot–controller execution dynamics provides a general mechanism for accelerating imitation policies beyond the speed of human demonstrations.

AI USE STATEMENT

In this work, we used generative AI tools for translation assistance, readability editing, and refinement of the manuscript. We have not used generative AI tools for synthetic data generation, theoretical model development, mathematical formulation, proof assistance, hypothesis refinement, research methodology design, or data cleaning. We also used generative AI tools for implementation of evaluation scripts, result interpretation, paper drafting, LaTeX formatting, brainstorming, structural suggestions, and reference formatting. We have reviewed all AI-assisted work. AI-assisted text was reviewed and revised by the authors, and any AI-assisted code was independently inspected and tested by the authors before use. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The neural robot dynamics model, the compensation fine-tuning objective, and the policy backbone are specified in Sections 3.2 to 3.4. The hardware, control rates, dynamics data collection, and evaluation protocol are described in Section 4. We will release the source code, trained models, and relevant data and configuration files upon publication, subject to any applicable hardware or dataset restrictions.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or sensitive personal data. The experiments use physical robot systems and were conducted with appropriate attention to robot safety and the surrounding environment.

REFERENCES

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Conference on Robot Learning (CoRL)*, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025c.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking, 2025d. URL https://arxiv.org/abs/2512.05964.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

Nolan Fey, Gabriel B. Margolis, Martin Peticco, and Pulkit Agrawal. Bridging the sim-to-real gap for athletic loco-manipulation. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.125. URL https://www.roboticsproceedings.org/rss21/p125.html.

Lingxiao Guo, Zhengrong Xue, Zijing Xu, and Huazhe Xu. DemoSpeedup: Accelerating visuomotor policies via entropy-guided demonstration acceleration. In *Conference on Robot Learning (CoRL)*, 2025.

Xiaoshen Han, Haoyu Xiong, Haonan Chen, Chaoqi Liu, Antonio Torralba, Yuke Zhu, and Yilun Du. B-spline Policy: Accelerating manipulation policies via B-spline action representations. *arXiv preprint arXiv:2607.09648*, 2026.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Sunwoo Kim, Jeongjun Kim, and Joseph Lim. Time optimal execution of action chunk policies beyond demonstration speed. In *International Conference on Learning Representations (ICLR)*, 2026.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Nadun Ranawaka Arachchige, Zhenyang Chen, Wonsuhk Jung, Woo Chul Shin, Rohan Bansal, Pierre Barroso, Yu Hang He, Yingyan Celine Lin, Benjamin Joffe, Shreyas Kousik, and Danfei Xu. SAIL: Faster-than-demonstration execution of imitation learning policies. In *Conference on Robot Learning (CoRL)*, 2025.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 91–100. PMLR, 2022. URL https://proceedings.mlr.press/v164/rudin22a.html.

Koki Yamane, Cristian C. Beltran-Hernandez, Steven Oh, Masashi Hamaya, and Sho Sakaino. Refinement of accelerated demonstrations via incremental iterative reference learning control for fast contact-rich imitation learning, 2026. URL https://arxiv.org/abs/2604.16850.

David D. Yuan, Tony Z. Zhao, Kaylee Burns, and Chelsea Finn. Speedtuning: Speeding up policy execution with lightweight reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1184–1192, 2025. doi: 10.1109/ICRA55743.2025.11128753.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.