TINY RECURSIVE POLICY: RECURSIVE LATENT REFINEMENT FOR VISUOMOTOR CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Behaviour cloning has become a powerful approach for robotic manipulation, but many modern policies reconstruct their internal representation from a fixed observation window at each prediction and do not maintain persistent latent context throughout execution. We introduce **Tiny Recursive Policy (TRP)**, a parameter-efficient behaviour-cloning policy that combines persistent temporal state with recursive latent refinement. At each control step, TRP repeatedly updates high- and low-level latent states using a shared computation module before decoding an action sequence, and propagates the resulting state to subsequent predictions. We evaluate TRP across LIBERO-100, MetaWorld ML45, and the LIBERO-Mem benchmarks. TRP performs competitively across multi-task and few-shot settings and shows substantial gains on long-horizon and context-dependent tasks, while using significantly fewer parameters than large generative baselines. Execution-time perturbations show that reliance on the persistent carry increases with temporal-context demand, while training with carry perturbations substantially improves robustness to both carry and visual corruptions. Ablations reveal a strong interaction between persistent state and recursive refinement: the benefit of carry depends strongly on the amount of within-step computation, and the resulting performance cannot be explained by parameter count alone.

1 INTRODUCTION

Behaviour cloning (BC) has become a central paradigm for learning robotic manipulation policies from demonstrations. Recent advances have significantly improved its ability to model complex and long-horizon behaviours. Action-chunking methods predict temporally coherent sequences of future actions rather than individual control commands Zhao et al. (2023), while discrete latent-action approaches learn reusable representations of temporally extended behaviour Lee et al. (2024); Mete et al. (2024). Generative policies, particularly diffusion- and flow-based methods, have further improved continuous control through iterative action generation and expressive policy parameterisations Chi et al. (2025); Reuss et al. (2023); Pan et al. (2026).

On the other hand, retaining temporal information is another important consideration for robotic policies. In long-horizon manipulation settings, the robot's actions may depend not only on the current observation, but also on the interaction history. Observation histories, action chunking, and temporal abstractions can capture this structure within a fixed temporal window, and many policies reconstruct their internal state from that window at each query. Recurrent models propagate latent context across action steps, allowing temporal interaction information to directly influence later actions and making the policy more robust than a fixed temporal window. This is especially relevant when visually similar observations occur at different stages of execution, where interaction history can provide additional context. This distinction becomes particularly important in partially observable settings, where identical or visually similar observations may require different actions depending on earlier interactions.

Furthermore, recurrence can provide more than temporal persistence. Rather than committing to an action sequence after a single forward pass, recursive refinement lets later computation revise and build on intermediate representations conditioned on the current observation and temporal model state. This provides a parameter-efficient way of increasing policy expressivity without increasing model size. Recent recursive reasoning models have demonstrated the effectiveness of this principle.

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.

Hierarchical Reasoning Model (HRM) Wang et al. (2025) performs repeated computation using interacting high- and low-level recurrent states operating at different timescales, while Tiny Recursive Model (TRM) Jolicoeur-Martineau (2025) shows that a small shared network can achieve strong reasoning performance through repeated latent refinement. These results motivate investigating whether recursive refinement, together with persistent temporal state, can provide an effective and parameter-efficient mechanism for robotic control.

In this work, we introduce **Tiny Recursive Policy (TRP)**, a behaviour-cloning policy that combines persistent temporal state with recursive latent refinement. At each control step, TRP repeatedly updates a pair of high- and low-level latent states using a small shared module before generating future actions. The resulting states are propagated to the next policy step, allowing subsequent predictions to incorporate temporal context. We additionally introduce a training-time carry perturbation objective that increases robustness to imperfect recurrent states. We evaluate TRP across multi-task, few-shot, long-horizon, and context-dependent manipulation settings on LIBERO, MetaWorld ML45, and LIBERO-Mem. TRP performs competitively across multi-task and few-shot benchmarks, while outperforming baselines considerably on long-horizon and non-Markovian tasks. Through execution-time perturbations, we further show that reliance on the persistent carry increases with temporal-context demand: resetting the carry has almost no effect on MetaWorld but substantially degrades performance on LIBERO-LONG and LIBERO-Mem. Carry perturbation makes the resulting policy substantially more robust to both carry and observation corruptions. Finally, ablations over recursion depth, model size, and temporal gradient propagation show that TRP's gains arise from the interaction between persistent state and repeated computation, not parameter count alone.

2 Related Work

Behaviour Cloning Behaviour cloning (BC) learns policies from expert observation-action pairs and has long provided an effective framework for imitation learning Pomerleau (1988); Bain & Sammut (1995); Argall et al. (2009); Osa et al. (2018). However, in robotic manipulation, demonstrations commonly exhibit multimodal action distributions and temporal associations, making direct single-step regression insufficient for many complex tasks. Early approaches addressed these issues using mixture-density policies Mandlekar et al. (2021), implicit energy-based policies Florence et al. (2022), and discretised autoregressive action models such as Behaviour Transformer (BeT) Shafiullah et al. (2022) and VQ-BeT Lee et al. (2024). Another line of work utilises temporal structure by generating extended action sequences or learning temporally abstracted latent skills. Action Chunking Transformer (ACT) predicts action chunks using a Conditional Variational Autoencoder (CVAE) Zhao et al. (2023), while latent-plan and skill-prior approaches compress temporally extended behaviour into reusable representations Lynch et al. (2020); Pertsch et al. (2021). More recent approaches learn discrete latent action spaces, including PRISE Zheng et al. (2024), TAP Jiang et al. (2022), H-GAP Jiang et al. (2024), and QueST Mete et al. (2024), with QueST explicitly learning sequences of discrete skill tokens to improve multi-task and few-shot transfer.

Generative policies have become a dominant approach for continuous Behaviour Cloning. Diffusion Policy introduced conditional action diffusion for visuomotor control Chi et al. (2025), followed by goal-conditioned diffusion policies and extensions incorporating 3D observations, geometric equivariance, and larger policy architectures Reuss et al. (2023); Ze et al. (2024); Ke et al. (2024); Yang et al. (2024); Liu et al. (2025). Flow-based policies provide a related iterative formulation while enabling faster generation Zhang et al. (2025); Hu et al. (2024). Although these approaches are commonly motivated by their ability to represent complex or multimodal action distributions, a recent work Pan et al. (2026) shows that much of their advantage can instead be attributed to stochasticity combined with supervised iterative computation; their Minimal Iterative Policy (MIP) matches flow-based policies using only two regression-based refinement steps.

Recursive Reasoning Recent work on recursive reasoning models has explored iterative latent computation as an alternative to scaling model size or generating explicit chain-of-thought traces. Hierarchical Reasoning Model (HRM) Wang et al. (2025) introduced this direction using high- and low-level recurrent modules operating at different timescales, together with deep supervision and adaptive computation. Tiny Recursive Models (TRM) Jolicoeur-Martineau (2025) showed that much of HRM's architectural complexity was unnecessary, replacing the two-module hierarchy with a single small network that repeatedly refines a latent reasoning state and predicted solution, improving

performance while substantially reducing the parameter count. This recursive-refinement principle has since been extended beyond discrete reasoning: Tiny Recursive Control (TRC) Jain & Linares (2026) applies repeated prediction and trajectory simulation to optimal-control problems, while the Recursive Token Mapper (RTM) Esmaeilzadeh et al. (2026) replaces the latent mapper of style-based image generators with recursive refinement, improving image quality and distribution coverage. However, recursive refinement does not appear to transfer uniformly across settings: Autoregressive TRMs Rauba et al. (2026) show that TRM-style nested within-token refinement provides no advantage over standard recurrent Transformers, suggesting that its benefits may depend on the ability to repeatedly revise a candidate solution rather than committing autoregressively to individual outputs.

3 METHOD

We now formalise TRP. The policy combines within-step recursive refinement with a latent carry propagated across control timesteps. Let the training dataset be composed of robot demonstration trajectories

$$\mathcal{D} = \{\tau^j\}_{j=1}^{N}, \qquad \tau^j = \{(o_t^j, a_t^j)\}_{t=0}^{T_j-1}, \tag{1}$$

where o_t^j denotes the robot observation at time t in trajectory j, and a_t^j denotes the corresponding action. The observation o_t may contain visual observations, task descriptions, proprioceptive state or gripper state. At control time t, the model receives an observation window

$$O_t = \{o_{t-n_o+1}, \ldots, o_t\}, \tag{2}$$

where n_o is the observation horizon. The supervised prediction target is an action sequence

$$A_t = (a_t, a_{t+1}, \ldots, a_{t+H_p-1}), \tag{3}$$

where H_p is the prediction horizon. The observation window is first encoded into a conditioning representation

$$x_t = f_{\text{obs}}(O_t), \tag{4}$$

where f_{obs} denotes the observation encoder. To predict a sequence of future actions, the conditioning representation is converted into a sequence of action-query embeddings,

$$X_t = \{x_t + p_h\}_{h=0}^{H_p-1}, \tag{5}$$

where p_h is a learnable embedding indicating the future action index. Thus, the recurrent computation core operates over the future action slots rather than directly over the raw observation sequence. The core maintains two latent states,

$$c_t = (z_{H,t}, z_{L,t}), \qquad z_{H,t}, z_{L,t} \in \mathbb{R}^{H_p \times d}, \tag{6}$$

where $z_{H,t}$ is a high-level latent state and $z_{L,t}$ is a low-level latent state. The pair c_t is the persistent carry. At the beginning of an episode, it is initialised as

$$c_0 = (z_H^{\text{init}}, z_L^{\text{init}}), \tag{7}$$

and after each policy evaluation, it is updated and reused at the next timestep. Let $\mathcal{T}_\theta$ be a shared compute module. In practice, this can be implemented as a small transformer block, but the same block is reused for both low-level refinement and high-level update. We define an update operator

$$\mathcal{U}_\theta(z, u) = \mathcal{T}_\theta(z + u), \tag{8}$$

where z is the latent state being updated and u is the conditioning input injected into the update. Given the incoming carry $c_t^{\text{in}} = (z_{H,t}^0, z_{L,t}^{0,0})$, the recursive computation block performs R high-level cycles. Each high-level cycle contains K low-level refinement steps. The low-level state is carried between high-level cycles, such that $z_{L,t}^{1,0} = z_{L,t}^{0,0}$ and $z_{L,t}^{r,0} = z_{L,t}^{r-1,K}$ for $r > 1$:

$$z_{L,t}^{r,\ell} = \mathcal{U}_\theta\left(z_{L,t}^{r,\ell-1}, z_{H,t}^{r-1} + X_t\right), \quad z_{H,t}^{r} = \mathcal{U}_\theta\left(z_{H,t}^{r-1}, z_{L,t}^{r,K}\right), \tag{9}$$

where $\ell = 1, \ldots, K$ and $r = 1, ..., R$. The low-level state z_L performs fast repeated refinement conditioned on the current high-level state and the observation. These equations describe the forward recursive computation. However, during training, we follow the truncated gradient scheme of

Algorithm 1 Stateful recursive policy step

Require: Observation window O_t, incoming carry $c_t^{\text{in}} = (z_H, z_L)$
Require: Shared computation module $\mathcal{U}_\theta$, action head h_ϕ
Require: Number of high-level cycles R, number of low-level refinement steps K
$x_t \leftarrow f_{\text{obs}}(O_t)$ ▷ Encode current observation window
$X_t \leftarrow (x_t + p_0, \ldots, x_t + p_{H_p-1})$ ▷ Add future action index
for $r = 1, \ldots, R$ **do**
  for $k = 1, \ldots, K$ **do**
    $z_L \leftarrow \mathcal{U}_\theta(z_L, z_H + X_t)$ ▷ Fast low-level refinement
  end for
  $z_H \leftarrow \mathcal{U}_\theta(z_H, z_L)$ ▷ Slow high-level update
end for
$\hat{A}_t \leftarrow h_\phi(z_H)$ ▷ Predict future action sequence
$c_t^{\text{out}} \leftarrow (z_H, z_L)$ ▷ Updated persistent latent carry
return $\hat{A}_t, c_t^{\text{out}}$

Jolicoeur-Martineau (2025): the first $R-1$ high-level cycles are evaluated without gradient tracking, and their resulting latent states are detached before the final cycle. Therefore, gradients propagate only through the K low-level refinement steps and the high-level update of the final cycle. At inference time, all R cycles are evaluated identically, as gradient tracking is not required.

After the recursive updates, the final states are decoded into an action sequence:

$$\hat{A}_t = h_\phi(z_{H,t}^R), \tag{10}$$

where h_ϕ is the action head and

$$\hat{A}_t = (\hat{a}_t, \hat{a}_{t+1}, \ldots, \hat{a}_{t+H_p-1}). \tag{11}$$

We experimented with different carry latent combinations to feed into the action head and observed that, while there isn't a large difference, using z_H only generally yielded better performance than other options (concatenation, summation, z_L only), so we use it as the default for all evaluations.

The output carry is defined as

$$c_t^{\text{out}} = (z_{H,t}^R, z_{L,t}^{R,K}). \tag{12}$$

This carry is passed to the next policy iteration:

$$c_{t+1}^{\text{in}} = \text{sg}\left(c_t^{\text{out}}\right), \tag{13}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. Thus, the policy is stateful at inference time, since the latent states persist across timesteps, but during training, the carry is detached between consecutive timesteps, preventing gradients from propagating across policy evaluations while retaining recurrent state information in the forward pass. The action prediction objective is standard supervised behaviour cloning over action sequences:

$$\mathcal{L} = \mathbb{E}_{\tau \sim \mathcal{D},\, t}\left[\sum_{h=0}^{H_p-1} ||\hat{a}_{t+h} - a_{t+h}||_2^2\right]. \tag{14}$$

Equivalently, the model learns a deterministic sequence policy

$$\hat{A}_t = \pi_\Theta(O_t, c_t^{\text{in}}), \tag{15}$$

where $\Theta = \{\theta, \phi\}$ together with the observation encoder parameters.

Optionally, the model can be trained with carry perturbation to improve robustness to imperfect latent states. Let c_t^{out} be the clean carry and let $\epsilon_c \sim \mathcal{N}(0, I)$. A perturbed carry can be formed as

$$\tilde{c}_t^{\text{out}} = (1-\lambda)c_t^{\text{out}} + \lambda\epsilon_c, \tag{16}$$

where λ controls the perturbation strength. A second supervised prediction from the perturbed carry gives an auxiliary loss,

$$\mathcal{L} = \mathcal{L}_{\text{clean}} + \beta\mathcal{L}_{\text{perturbed}}, \tag{17}$$

where β weights the auxiliary objective. This encourages the recurrent policy to remain stable when the persistent latent carry is noisy or imperfect, without changing the inference-time policy. The proposed policy is also summarised in Algorithm 1 which describes one stateful policy evaluation.

Table 1: **Multi-task and few-shot performance on LIBERO-100.** Success rates are reported for LIBERO-90 and LIBERO-LONG. Few-shot models are pretrained on LIBERO-90 and fine-tuned using five demonstrations per LIBERO-LONG task. Parameters represent the backbone parameter count.

	Multi-task		Few-shot	
Model	**LIBERO-90**	**LIBERO-LONG**	**LIBERO-LONG**	**Params (M)**
ResNet-T Liu et al. (2023)	0.844	0.441	0.519	4.8
DP Chi et al. (2025)	0.916	0.849	0.688	336.3
ACT Zhao et al. (2023)	0.704	0.500	0.240	13.87
VQ-BET Lee et al. (2024)	0.724	0.538	0.370	6.43
Quest Mete et al. (2024)	0.861	0.704	0.427	18.48
MIP Pan et al. (2026)	0.918	0.846	0.641	336.3
FLOW Pan et al. (2026)	0.899	0.897	0.558	336.3
TRP	0.929	**0.919**	**0.704**	**4.2**
TRP-perturbed	**0.935**	0.913	0.696	**4.2**
TRP-no-carry	0.733	0.783	0.374	**4.2**

4 Experimental Results

We evaluate TRP across LIBERO-100 Liu et al. (2023), MetaWorld ML45 Yu et al. (2020), and LIBERO-Mem Chung et al. (2026). On LIBERO-100 and MetaWorld, we consider both multi-task learning and few-shot generalisation. For MetaWorld, we follow the ML45 evaluation protocol of Mete et al. (2024). The benchmark contains 45 training tasks and five held-out tasks. We collect 100 demonstrations per training task using the scripted policies provided by MetaWorld and evaluate each task over 50 rollouts. For few-shot evaluation, the multi-task model is fine-tuned using five demonstrations from each of the five held-out tasks. The reported results are over 3 seeds.

We evaluate three variants of our method: **TRP**, our full model with persistent latent carry across timesteps; **TRP-no-carry**, which discards the carry after each policy evaluation; and **TRP-perturbed**, which applies the carry perturbation objective during training. We compare against behaviour-cloning baselines from three families. Among policies that predict actions directly, **ResNet-T** Liu et al. (2023) is the LIBERO reference policy, a ResNet encoder followed by a temporal transformer over recent observations; **LSTM-GMM** Mandlekar et al. (2021) is a recurrent policy with a Gaussian-mixture action head and, like TRP, carries state across timesteps; and **ACT** Zhao et al. (2023) predicts chunks of actions with a transformer trained as a conditional VAE. Two baselines act through discrete codes: **VQ-BeT** Lee et al. (2024) tokenises action chunks with residual vector quantisation and predicts the tokens with a GPT-style transformer, and **QueST** Mete et al. (2024) learns a vocabulary of skill tokens with finite scalar quantisation and predicts them autoregressively. The remaining baselines refine actions iteratively: **Diffusion Policy** (DP) Chi et al. (2025) denoises an action sequence over 100 DDPM steps with a 1-D convolutional U-Net, and **MIP** and **Flow** Pan et al. (2026) use the same U-Net with a two-step iterative update and rectified flow, respectively; DP's transformer variant (**DiT**) is also included in our model-size study. All models use the same vision encoder where possible for fairness.

4.1 Multi-Task Learning

We first evaluate whether TRP can learn a single policy across a diverse set of manipulation tasks. On LIBERO, models are trained using all 50 demonstrations for each task and evaluated over 50 rollouts. We report results on both LIBERO-90, which primarily contains shorter-horizon tasks, and LIBERO-LONG, which contains longer manipulation behaviours. We additionally evaluate on MetaWorld ML45 following the protocol of Mete et al. (2024).

As shown in Table 1, TRP performs strongly across both LIBERO multi-task settings. On LIBERO-90, TRP and TRP-perturbed outperform all baselines, with TRP-perturbed obtaining the highest success rate. The difference becomes clearer on LIBERO-LONG, where the margin over the strongest non-TRP baseline widens. The MetaWorld results in Figure 1 show a similar regime. Multi-task

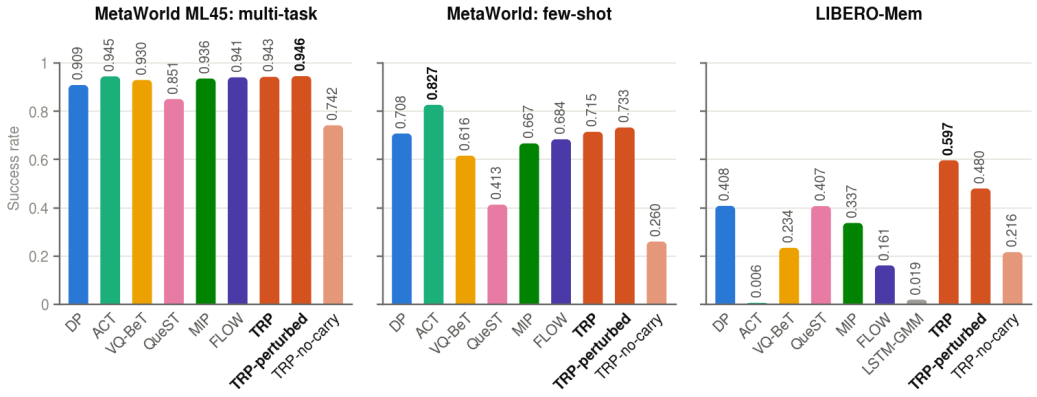


Figure 1: **Multi-task, few-shot, and context-dependent evaluation.** Success rates on MetaWorld ML45 multi-task learning and five-shot transfer to the five held-out ML45 tasks, together with subgoal completion on LIBERO-Mem. MetaWorld follows the evaluation protocol of Mete et al. (2024). The comparison highlights the effect of removing or perturbing the persistent latent carry across increasingly transfer- and context-dependent settings.

ML45 performance is high for several methods, with ACT, MIP, Flow, and the two full TRP variants all concentrated near the top of the range. TRP-perturbed achieves the highest mean, but only by a small margin. In contrast, TRP-no-carry performs substantially worse despite using the same backbone and recursive computation. As examined further in Section 4.4, this gap arises primarily from training with persistent carry: wiping the carry from an already trained TRP has almost no effect on MetaWorld performance. Thus, on these tasks the carry shapes the learned policy during training without being required as online temporal context.

4.2 Few-Shot Generalisation

Next, we evaluate transfer to tasks with very limited task demonstration data. On LIBERO, models are first trained on the complete LIBERO-90 dataset and then fine-tuned using only five demonstrations from each LIBERO-LONG task. On MetaWorld, we again follow Mete et al. (2024): the model trained jointly on the 45 ML45 training tasks is fine-tuned using five demonstrations from each of the five held-out tasks. This setting tests whether representations learned from many source tasks can be adapted to new behaviours using limited data.

On LIBERO-LONG, TRP remains the strongest model in the five-shot setting, reaching 0.704 success, while TRP-perturbed performs similarly. The margins between the full TRP variants and the strongest generative baseline are relatively small, but the comparison with TRP-no-carry is much larger. Removing the persistent state reduces performance to roughly half that of the full model. Interestingly, this degradation is much stronger than in standard multi-task training, suggesting that the carried latent context is useful when the target behaviour is learned from very few examples. On the other hand, while TRP outperforms most of the strong baselines in the MetaWorld few-shot setting, ACT performs best in this setting. However, it should be noted that in the Libero few-shot setting, ACT is the worst performer among the baselines. The no-carry variant also falls short of every policy in this setting, supporting the importance of persistent model state.

4.3 Evaluation on Context-Dependent Tasks

We further evaluate on LIBERO-Mem Chung et al. (2026), a benchmark designed specifically for non-Markovian manipulation under partial observability. LIBERO-Mem contains 10 tasks spanning object motion, object sequence, object relations, and object occlusion. Unlike standard manipulation tasks, the current observation is intentionally insufficient to uniquely determine the correct action: visually similar states can require different behaviours depending on the preceding interaction history. Tasks include repeated manipulation of the same object, ordered interactions with multiple visually

Benchmark	Model	Clean	Execution-time perturbation		
			Carry wiped	Blackout (short)	Blackout (long)
MetaWorld ML45	TRP	0.943	0.941	0.659	0.515
	TRP-perturbed	**0.946**	0.942	**0.672**	**0.531**
	TRP-no-carry	0.742	–	0.492	0.347
LIBERO-Long	TRP	**0.919**	0.578	0.633	0.438
	TRP-perturbed	0.913	**0.863**	**0.769**	**0.674**
	TRP-no-carry	0.783	–	0.524	0.412
LIBERO-Mem	TRP	**0.597**	0.347	0.366	0.304
	TRP-perturbed	0.480	**0.414**	**0.390**	**0.403**
	TRP-no-carry	0.216	–	0.136	0.117

Table 2: **Closed-loop performance under execution-time perturbations**. *Clean* reports the corresponding headline result. *Carry wiped* resets the recurrent carry before every prediction. *Blackout* blanks all camera inputs for a short or long window mid-episode before restoring them (8/20 environment steps on MetaWorld and 16/32 on LIBERO).

similar objects, and reasoning about objects after occlusion. LIBERO-Mem uses a subgoal-aware evaluation rather than relying only on binary task success, which we follow.

As shown in Figure 1, this setting produces the largest separation between TRP and the comparison methods. Standard TRP achieves a subgoal-completion score of 0.597, while the strongest non-TRP methods remain around 0.41. This contrasts with MetaWorld multi-task learning, where several methods achieve similar performance, and with the standard LIBERO benchmarks, where the margins are smaller. Consequently, TRP is relatively the most advantageous when successful execution depends on information not recoverable from the current observation alone.

The TRP variants provide further evidence for this interpretation. Removing the carry substantially reduces performance, while carry perturbation produces an intermediate result. This ordering is consistent with the persistent latent state carrying useful task context across policy evaluations.

Strong action-generation methods also do not necessarily translate into strong performance in this setting. Diffusion Policy and QueST, for example, are competitive on several of the multi-task and few-shot evaluations but remain substantially below TRP on LIBERO-Mem. This suggests that expressive action distributions or temporal action abstractions alone do not fully substitute for state propagated across successive policy evaluations. Consequently, we view the LIBERO-Mem result as complementary to the long-horizon experiments: LIBERO-LONG tests whether a policy can handle extended execution, whereas LIBERO-Mem more directly tests whether information from earlier interaction can influence later interactions.

4.4 Ablations

Execution-time role of the carry. We first probe how trained policies use the persistent carry during closed-loop execution. Table 2 compares each policy against two execution-time perturbations. In *carry wiped*, we reset the carry to its initial value before each prediction, preventing information

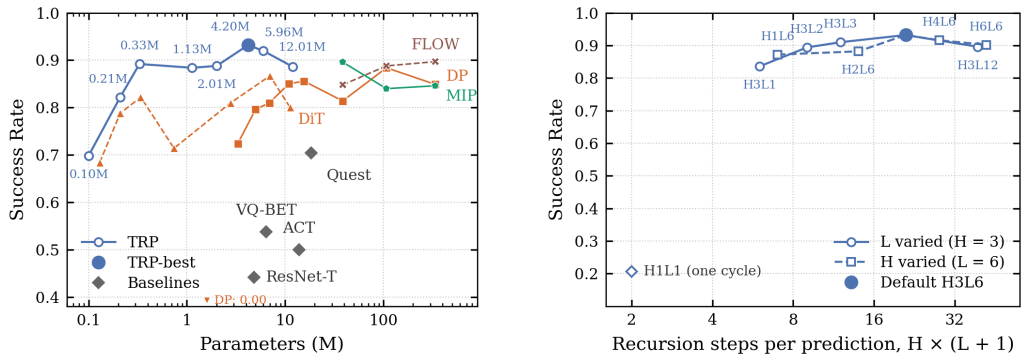
(a) Interaction between persistent carry and recursive refinement.

Carry	Cycles	MetaWorld	LIBERO-LONG	LIBERO-Mem
on	H3L6	**0.943**	**0.919**	**0.597**
on	H1L1	0.630	0.206	0.362
off	H3L6	0.742	0.783	0.216
off	H1L1	0.794	0.642	0.305

(b) Temporal gradient propagation through the carry.

Carry gradient	LIBERO-LONG	LIBERO-Mem
detached (TRP)	**0.919**	**0.597**
TBPTT, $k = 4$	0.882	0.441
TBPTT, $k = 8$	0.816	0.370
TBPTT, $k = 16$	–	0.326

Table 3: **Ablations of the persistent latent carry.** (a) compares the default recursive configuration with a single-cycle model, with and without carry propagation. (b) compares the detached carry used by TRP against truncated backpropagation through time (TBPTT).



(a) Model size.

(b) Recursion depth.

Figure 2: Ablations on LIBERO-LONG. (a) Success rate against model backbone parameter count. (b) Success rate against recursion steps per prediction

from propagating across policy evaluations while leaving the current observation unchanged. In *blackout*, all camera inputs are blanked for a short or long interval mid-episode and subsequently restored. This removes visual information while also corrupting carry with uninformative observations. TRP-no-carry provides a stateless reference.

Carry reliance across benchmarks. The effect of the carry differs sharply between training and execution. On MetaWorld ML45, wiping the carry from a trained TRP has almost no effect: TRP falls only from 0.943 to 0.941. In contrast, TRP-no-carry, which uses the same reset-before-prediction inference procedure but was trained without inter-step carry, reaches only 0.742. The resulting gap is therefore entirely a training-time effect rather than evidence that MetaWorld requires persistent temporal context at execution. One possible explanation is that propagating state during training provides a useful warm start for the recursive computation or improves the conditioning of the optimisation problem, even though the final policy does not depend on that state. This changes on the more temporally demanding benchmarks. Wiping the carry reduces standard TRP by approximately 37% on LIBERO-LONG and 42% on LIBERO-Mem, showing that the trained policy increasingly relies on propagated state during execution. Thus, carry appears to play two distinct roles: it can shape the learned policy during training even when dispensable at test time, and on temporally dependent tasks it also provides useful online context.

Robustness from carry perturbation. Training with carry perturbations substantially reduces sensitivity to both carry removal and visual blackout on LIBERO-LONG and LIBERO-Mem. Wiping the carry causes only modest degradation for TRP-perturbed compared with standard TRP, and the same pattern holds under short and long blackouts. Under long blackout, for example, TRP-perturbed retains 0.674 success on LIBERO-LONG, compared with 0.438 for TRP, and also remains stronger on LIBERO-Mem. TRP-perturbed additionally degrades less than TRP-no-carry in these settings, suggesting that perturbation training produces a more robust recurrent state when visual evidence is temporarily unavailable. MetaWorld behaves differently: since wiping the carry has negligible effect, blackout primarily removes useful visual input and TRP and TRP-perturbed degrade similarly. These robustness gains come with little cost to clean performance, although standard TRP remains stronger on LIBERO-Mem.

Interaction between carry and recursion. Persistent state is not beneficial independently of recursive refinement. Table 3a shows a strong interaction between the two. With the default recursive configuration, enabling the carry substantially improves performance across all three benchmarks. At shallow recursion, however, the effect can reverse: on LIBERO-LONG, H1L1 drops from 0.642 without carry to 0.206 with carry, and MetaWorld shows the same qualitative pattern. LIBERO-Mem is the exception, where carry remains mildly beneficial even in the shallow model. These results suggest that persistent state must be sufficiently processed before it becomes useful. A shallow model may be unable to reconcile the incoming carry with the current observation, causing stale or poorly adapted latent state to interfere with action prediction. Additional recursive refinement may instead

allow the policy to update, reinterpret, or overwrite the incoming state before decoding an action. We therefore do not view carry and recursion as independently beneficial components: the usefulness of persistent state depends strongly on having sufficient within-step computation to integrate it.

Temporal gradient propagation. Table 3b tests whether the persistent carry should also transmit gradients between successive policy evaluations. The default TRP detaches the carry between timesteps, while the alternatives apply truncated backpropagation through time over increasingly long horizons. Propagating gradients through the carry consistently reduces performance on both LIBERO-LONG and LIBERO-Mem, with larger truncation horizons producing further degradation. Persistent forward state therefore provides the benefit observed by TRP without requiring long-range gradient propagation.

Model capacity. Figure 2a shows whether model capacity alone explains TRP's performance. Performance rises rapidly at small model sizes, with a 0.33M-parameter backbone already approaching the strongest much larger baselines on LIBERO-LONG. Performance peaks at the 4.2M-parameter configuration and then decreases as the backbone grows further. Thus, TRP's gains do not follow a simple scaling trend; recursively using a compact computation module appears more effective than increasing parameter count alone.

Recursion depth. Finally, Figure 2b varies the amount of recursive computation while holding the backbone fixed. Increasing either the number of high-level or low-level refinement cycles initially improves performance, with the default configuration achieving the strongest result. Additional recursion beyond this point provides no further benefit and eventually reduces performance. Together with the model-size ablation, this indicates that TRP benefits from additional computation through parameter reuse, but that the advantage saturates at a moderate recursion depth.

5 Conclusion

We introduced Tiny Recursive Policy, a stateful behaviour-cloning policy that combines recursive latent refinement with persistent temporal context for robotic manipulation. Rather than increasing model size or relying on iterative generative inference, TRP repeatedly reuses a compact computation module while propagating its latent state across successive policy evaluations. Across LIBERO, MetaWorld ML45, and LIBERO-Mem, TRP performs competitively in multi-task and few-shot settings and shows its strongest advantages on long-horizon and context-dependent manipulation tasks. Beyond aggregate performance, our execution-time interventions show that the role of the recurrent state depends strongly on the task. Wiping the carry has almost no effect on MetaWorld, but causes substantial degradation on LIBERO-LONG and LIBERO-Mem, indicating increasing reliance on persistent context as temporal dependencies become stronger. Training with carry perturbations substantially reduces this sensitivity and improves robustness to temporary visual blackout. Additional ablations show that persistent state and recursive refinement interact strongly, that propagating gradients through the carry is unnecessary, and that increasing recursion is more effective than simply increasing model capacity up to a moderate computation depth. Together, these results suggest that persistent state, when paired with sufficient recursive refinement, provides a parameter-efficient mechanism for building compact visuomotor policies for temporally demanding control.

Limitations and Future Work While our results demonstrate strong performance, several limitations remain. First, our evaluation is currently restricted to simulation, and real-robot experiments are needed to establish whether the same benefits persist in real-world settings. Second, although our intervention experiments identify when the policy relies on its persistent carry, they do not reveal what information is encoded in the latent state, how individual task variables are represented, or how long particular information is retained. Probing or decoding the carry could more directly characterise the temporal representations learned by TRP. Finally, TRP currently uses a fixed amount of recursive computation for every prediction. Our recursion-depth experiments show that additional computation is beneficial only up to a point, motivating adaptive halting or input-dependent computation mechanisms that allocate refinement according to the difficulty of the current decision.

AI USE STATEMENT

In this work, we used generative AI tools, including AI coding assistants, under human direction to aid and polish writing; draft and edit sections of the manuscript; retrieve, identify, and check relevant literature and references; create and refine figures; and design, implement and orchestrate experiments. We did not use generative AI to generate synthetic datasets. All experimental results reported in the paper were produced by the training and evaluation runs described in the manuscript. The authors reviewed all AI-assisted work, verified generated or modified code through execution and testing, checked literature and references against their original sources, and checked quantitative claims against the recorded experimental results. We take responsibility for the final content of this work, including text, claims, code, and artefacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe the model architecture, training procedure, evaluation protocols, and ablation settings used in our experiments in the main paper. All reported results are averaged over the stated random seeds using the evaluation procedures described in the paper. The training and evaluation code, model configurations, and scripts used to reproduce the experiments will be released publicly upon acceptance.

REFERENCES

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine intelligence 15*, pp. 103–129, 1995.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Nhat Chung, Taisei Hanyu, Toan Nguyen, Huy Le, Frederick Bumgarner, Duy Minh Ho Nguyen, Khoa Vo, Kashu Yamazaki, Chase Rainwater, Tung Kieu, et al. Rethinking progression of memory state in robotic manipulation: An object-centric perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 3407–3415, 2026.

Mehdi Esmaeilzadeh, Alexia Jolicoeur-Martineau, Chirag Vashist, and Ke Li. One pass is not enough: Recursive latent refinement for generative models. *arXiv preprint arXiv:2605.15309*, 2026.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pp. 158–168. PMLR, 2022.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. *Advances in Neural Information Processing Systems*, 37:138836–138858, 2024.

Amit Jain and Richard Linares. Tiny recursive control: Iterative reasoning for efficient optimal control. In *AIAA SciTech 2026 Forum*, pp. 2380, 2026.

Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022.

Zhengyao Jiang, Yingchen Xu, Nolan Wagener, Yicheng Luo, Michael Janner, Edward Grefenstette, Tim Rocktäschel, and Yuandong Tian. H-gap: Humanoid control with a generalist planner. In *International Conference on Learning Representations*, volume 2024, pp. 17774–17791, 2024.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pp. 1113–1132. Pmlr, 2020.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. *Advances in Neural Information Processing Systems*, 37:4062–4089, 2024.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2): 1–179, 2018.

Chaoyi Pan, Giridharan Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, et al. Much ado about noising: Dispelling the myths of generative robotic control. In *International Conference on Learning Representations*, volume 2026, pp. 90575–90614, 2026.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Paulius Rauba, Claudio Fanconi, and Mihaela van der Schaar. Tiny autoregressive recursive models. *arXiv preprint arXiv:2603.08082*, 2026.

Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Jingyun Yang, Zi-ang Cao, Congyue Deng, Rika Antonova, Shuran Song, and Jeannette Bohg. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. *arXiv preprint arXiv:2407.01479*, 2024.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14754–14762, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Ruijie Zheng, Ching-An Cheng, Hal Daumé III, Furong Huang, and Andrey Kolobov. Prise: Llm-style sequence compression for learning temporal action abstractions in control. *arXiv preprint arXiv:2402.10450*, 2024.