

476 A Additional Experiments

477 In addition to the main results presented in §5.1, we conduct three experiments to further validate
 478 the design decisions behind our proposed method.

479 A.1 Which component of the policy benefits most from PTP?

480 To identify which part of the policy is most influenced by PTP, we compare a fully PTP-trained
 481 long-context policy against two ablated variants: *Encoder PTP*, where we first train the visual en-
 482 coder with PTP, then freeze it and train the action decoder without PTP; *Decoder PTP*, where we
 483 conversely train the encoder without PTP, freeze it, and then apply PTP only during decoder train-
 484 ing. As shown in Fig. 12, *Decoder PTP* achieves performance on par with the fully trained PTP
 485 policy, whereas *Encoder PTP* performs significantly worse. This result suggests that the benefits
 486 of PTP primarily stem from improved temporal modeling in the action decoder, rather than from
 487 changes to the visual encoder, directly motivating our multi-stage training recipe that decouples
 488 encoder pretraining from long-context policy learning.

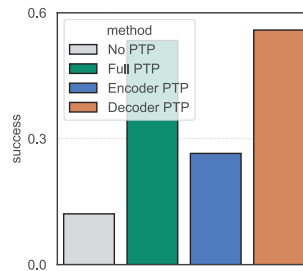


Figure 12: Performance of ablated PTP variants on Push-T. Applying PTP only to the decoder recovers the full PTP policy performance, whereas encoder-only PTP does not.

489 A.2 Does our method reduce reliance on action chunking?

490 Existing short-context policies typically rely on action chunking compensate for limited access to
 491 past observations. However, this common design choice comes at the cost of reduced reactivity. To
 492 assess whether our method alleviates this limitation, we compare the performance of three policy
 493 variants: (i) *short-context short-chunk*, which receives the past 2 frames as input and outputs single-
 494 step actions (chunk size 1); (ii) *long-context short-chunk*, which receives the past 16 frames and also
 495 outputs single-step actions; and (iii) *long-context long-chunk*, which receives the past 16 frames
 496 and outputs action chunks of size 8. As shown in Fig. 13, the *long-context short-chunk* policies
 497 trained by our method substantially outperform the *short-context* counterparts and recover most of
 498 the performance of the *long-context long-chunk* policies. This result demonstrates the effectiveness
 499 of our method in reducing reliance on open-loop action chunking.

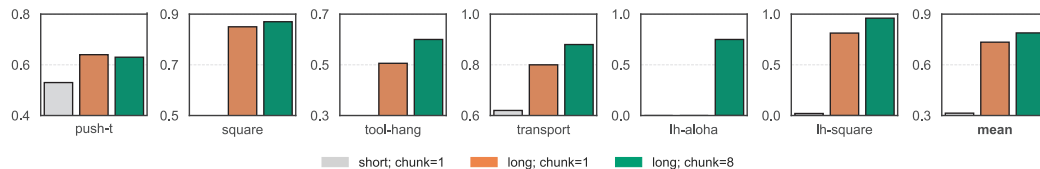


Figure 13: Comparison of policies with different context lengths and chunk sizes. Long-context policies trained with PTP perform significantly better than short-context policies when run in a fully closed-loop setting (chunk size 1). Moreover, they achieve performance comparable to long-context policies that use open-loop chunking (chunk size 8), indicating reduced reliance on chunking during execution.

A.3 Is PTP still critical when conditioning on past actions?

Our earlier analysis in Fig. 6 has shown the importance of PTP in capturing temporal action dependencies when the policy is conditioned on past observations. A natural question is whether PTP remains necessary when the model also has direct access to past actions. To understand this, we augment the input to diffusion policies with the previous 16 actions and compare performance with and without PTP. As shown in Fig. 14, even with access to past actions, the vanilla baseline performs poorly without PTP, while our method consistently yields substantially better results. Consistent with our previous findings, this result highlights the critical role of PTP in enabling diffusion policies to effectively model temporal structure, even when past actions are explicitly provided.

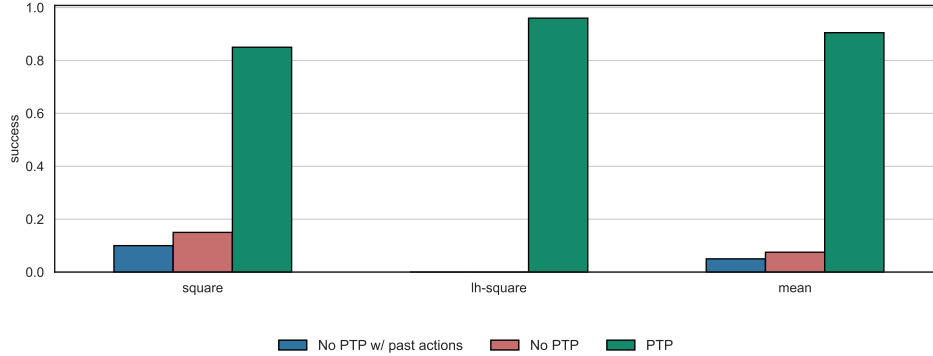


Figure 14: Comparison of adding past actions into the context history without PTP and our baseline of PTP and no PTP without actions. We observe that adding past actions in the observation doesn’t improve performance.

B Environmental Details

B.1 Real-World Tasks

We conduct real-world experiments using two robot platforms: a Franka Panda arm set up, and the ALOHA bimanual system. For the Franka setup, we follow the DROID hardware configuration [47], using a single arm with wrist-mounted RGB cameras and proprioceptive sensing. The observation space includes RGB images and end-effector pose, while the action space consists of 6-DoF Cartesian displacements and gripper commands. We collect 50-200 human demonstrations for each task. For the ALOHA platform, we use the bimanual robot as described in [2], with RGB camera inputs and proprioceptive feedback. The observation space consists of dual-arm RGB views and proprioception, and the action space includes joint displacements for both arms and gripper states. We collect 150 demonstrations for the tape replacement task.

B.2 Simulation Tasks

Our simulation tasks include the Push-T task [6], three existing tasks in the Robomimic benchmark [1], along with two new long-horizon tasks that we introduce:

- *Long-Horizon Square*: A variant of the RoboMimic square task, where the robot must place a block onto the farthest peg from its initial location. We collect 100 noisy scripted demonstrations to prevent the policy from inferring the goal using current pose information alone, thus requiring memory of the initial state.
- *Long-Horizon ALOHA*: A simulated bimanual task where one arm picks up a block, moves it to the center of the workspace, and places it back at its original location. Success requires remembering the block’s starting position, highlighting the need for long-term memory.

Table 1: Hyper-parameters in simulation and real-world experiments.

Hyperparameter	LH Square	LH Aloha	Block Move	Two Scoops	Mug Repl.	Tape Repl.
Epochs	500	1500	500	500	500	1500
# Demos	100	50	50	200	200	150
# Subsampled frames	20	20	1	20	1	24
# Observations	20	32	10	20	32	20

B.3 Implementation Details

B.3.1 Policy Architecture

We build upon the transformer-based Diffusion Policy codebase [6], which supports training and evaluation across multiple Robomimic tasks. All policies are trained for 500 epochs by default, using visual encoders and chunked action prediction. For long-horizon ALOHA tasks, we train for 1500 epochs to accommodate the added difficulty of bimanual coordination and higher-frequency control. To reduce training overhead, all long-context policies are initialized with a frozen short-context encoder, pretrained on 2-frame inputs. This design choice is supported by the analysis in Fig. 12, which shows that freezing the encoder does not impair performance. To further improve training efficiency, we cache visual embeddings during data preprocessing and load them at runtime. This avoids repeatedly passing observations through the encoder and speeds up training.

B.3.2 Subsampling Rate

Real-world tasks often require longer history horizons, but full-length observation sequences can be computationally expensive. To reduce inference latency, we apply temporal subsampling to the input sequence. Specifically, instead of feeding all T observations t_0, t_1, \dots, t_{T-1} , we sample every K th frame, i.e., $t_{K-1}, t_{2K-1}, \dots, t_{T-1}$, where T is a multiple of K . This reduces the effective observation size while retaining broad temporal coverage. Subsampling values are listed in Table 1.

B.3.3 Context Length

When increasing the observation history, we scale the prediction horizon (past and future tokens) proportionally. We empirically find that the prediction length of future tokens has only a minor effect on task performance. Detailed context length configurations are provided in Table 2.

Table 2: Settings for different context lengths

Observation Length	2	4	8	16
Horizon	16	20	24	32
Future Tokens	14	16	16	16

B.3.4 Action Dependency Metric

To quantify how well a policy captures temporal action structure, we use *action predictability* as a proxy metric [24]. Specifically, we measure how accurately the current action a_t can be predicted from a window of $K = 15$ past actions, defined as $p(a_t \mid a_{t-K:t-1})$. We compute this quantity over policy rollouts and compare it to the same metric evaluated on the expert demonstrations. Higher predictability indicates stronger temporal action dependencies captured by the learned policy.

C Additional Results

In addition to the results reported in §5.1, we report per-task success rates across varying temporal context lengths, training conditions, and chunking configurations. Table 3 compares diffusion-based and regression-based baselines on six benchmark tasks, evaluated under different history lengths

561 and with or without PTP. Table 4 presents results in the closed-loop setting (chunk size = 1) across
562 different context lengths.

Table 3: Success rate (%) of diffusion-based and regression-based policies on simulation tasks under different training and history conditions. Results are reported as mean \pm standard deviation across 3 seeds.

Method	Push-T	Square	Tool-Hang	Transport	ALOHA	Long Square
Diffusion (PTP)	0.62 \pm 0.02	0.89 \pm 0.01	0.75 \pm 0.10	0.67 \pm 0.08	0.98 \pm 0.01	0.93 \pm 0.02
Diffusion (no-PTP)	0.59 \pm 0.01	0.17 \pm 0.01	0.00 \pm 0.00	0.00 \pm 0.00	0.20 \pm 0.19	0.03 \pm 0.02
Diffusion (no-hist)	0.67 \pm 0.03	0.79 \pm 0.06	0.51 \pm 0.14	0.60 \pm 0.08	0.28 \pm 0.04	0.12 \pm 0.05
Regression (PTP)	0.30 \pm 0.00	0.74 \pm 0.02	0.41 \pm 0.01	0.63 \pm 0.03	1.00 \pm 0.00	0.90 \pm 0.05
Regression (no-PTP)	0.27 \pm 0.00	0.78 \pm 0.04	0.40 \pm 0.07	0.51 \pm 0.02	1.00 \pm 0.00	0.90 \pm 0.00
Regression (no-hist)	0.64 \pm 0.07	0.19 \pm 0.02	0.18 \pm 0.00	0.43 \pm 0.04	0.45 \pm 0.00	0.00 \pm 0.00

Table 4: Success rate (%) on simulation tasks under closed-loop execution (chunk size = 1) .

Observations	Push-T	Square	Tool Hang	Transport	Long Square	Mean
2	0.53	0.13	0.62	0.053	0.02	0.37
4	0.53	0.68	0.84	0.13	0.02	0.51
8	0.59	0.83	0.86	0.48	0.11	0.63
16	0.64	0.85	0.82	0.51	0.81	0.77