LEARNING RECURRENT BELIEF REPRESENTATIONS FOR DIFFUSION POLICIES WITH PREDICTIVE WORLD MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion policies are commonly conditioned on a fixed observation window, restricting their effectiveness under partial observability, occlusion, and tasks requiring long-horizon memory. In a partially observable Markov decision process (POMDP), an effective policy should maintain a belief state that integrates historical information into a compact, structured world representation. We propose Belief-Inferring Recurrent Diffusion Policy (BIRD), which integrates recurrence with attention mechanisms to maintain a latent belief representation that conditions the diffusion policy. In addition to the imitation objective, the belief encoder is trained with an auxiliary world-modeling loss that predicts latent representations of future observations from the current belief and a future action sequence, encouraging the latent state to capture the underlying world state. Across six simulated 2D manipulation tasks, BIRD achieves a mean peak success rate of 74%, a $1.7\times$ improvement over the strongest baseline, and outperforms diffusion policies that lack a belief state or are trained solely with the imitation objective.

1 INTRODUCTION

Learning visuomotor policies from expert demonstrations has driven much of the progress in robot learning in recent years, especially in manipulation. Most of the methods in this domain rely on large generative control policies (GCPs) (Chi et al., 2023; Ze et al., 2024; Black et al., 2024) combined with action chunking (Zhao et al., 2023). GCPs can capture complex expert behavior and learn expressive action classes, which is especially relevant when combined with action chunking. However, GCPs struggle with partial observability. Most benchmarks sidestep it by considering fully observable systems, and action chunking only absorbs the short-range non-Markovianity of human demonstrations. Yet most real-world applications for autonomous robots deal with significant partial observability. Current GCPs address this problem using frame stacking (Chi et al., 2023) or transformers with limited context (Torne et al., 2025). Both approaches are limited by the number of frames that can be stacked, or kept in the transformer's context, making it challenging for GCPs to handle tasks with significant partial observability.

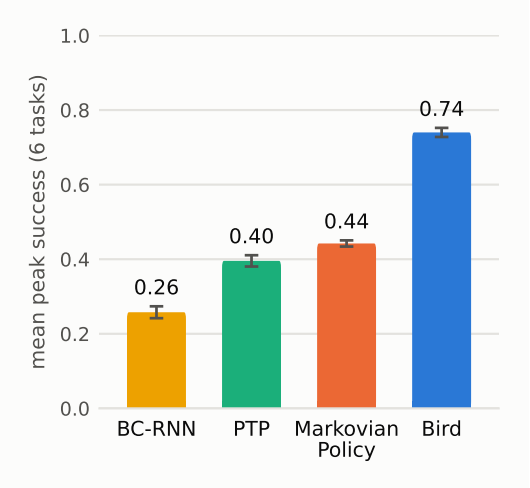


Figure 1: **Mean peak success over the six tasks.** Each bar averages the per-task results of Figure 4 over three seeds and six tasks, including the 0.00 that BC-RNN scores on both Push-Box variants. BIRD reaches 0.74, against 0.44 for the strongest baseline.

A partially observable task is modeled as a partially observable Markov decision process (POMDP), and a standard approach is to learn a belief representation of the system state (Kaelbling et al.,

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.

1998). Visual policies approximate this belief with either a transformer over a window of recent observations or a recurrent state, and each has a different limitation. A transformer is limited by what fits in its context. A task relevant observation that leaves the window of the last W observations cannot be recovered, since its observation is no longer in the transformer's context. Extending the context window increases attention cost quadratically, and the cloning loss must then fit a larger input from a fixed set of demonstrations, which leads to severe overfitting (De Haan et al., 2019; Lazzati et al., 2026). A recurrent state (Kapturowski et al., 2018) has no such bound, since it summarizes the entire history at $O(1)$ cost per step, but it must compress that history into a fixed-size state. This state must hold everything the task will need later, however, the cloning loss is applied only to the next action chunk, which does not guarantee that salient information is retained in the state for later in the episode.

We present BIRD, a belief-conditioned GCP for manipulation under partial observability that addresses both limits. A recurrent model learns a hidden state that integrates observations across the episode, so the belief's reach is not bounded by context. We accumulate these states into a latent memory of recent beliefs, which covers a longer span of the episode than a window of raw observations of the same size. The current belief cross-attends to this memory to incorporate salient cues from the past. To supervise the latent state representation to approximate the belief, we use a world-modeling objective to predict future vision tokens (Seo et al., 2023) of a frozen masked autoencoder (He et al., 2022). We also introduce a suite of six manipulation tasks in which the amount of memory a task requires is controlled, and find that BIRD matches or exceeds a memoryless diffusion policy where a reactive policy suffices and outperforms every baseline where the demonstrations are not Markovian.

2 Preliminaries

2.1 Partially Observable Markov Decision Processes

We model partially observable robot learning tasks as a POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, O, \rho_0 \rangle$, with state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, stochastic transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, emission model $O : \mathcal{S} \to \Delta(\mathcal{O})$, and initial state distribution ρ_0. At each step the agent receives $o_t \sim O(\cdot \mid s_t)$. The difficulty in acting in a POMDP comes from the fact that the agent cannot observe the state s_t, but only the observation o_t, thus an optimal policy must be conditioned on the belief, the posterior over states given the history, $b_t = p\big(s_t \mid o_{1:t},\, a_{1:t-1}\big)$. The belief can be updated recursively from b_{t-1}, a_{t-1} and o_t by Bayesian filtering, and it is a sufficient statistic for optimal control (Kaelbling et al., 1998), so that a POMDP is an MDP over beliefs.

In visual control, the belief is approximated by a learned recurrent network that consumes one observation at a time and carries a fixed-size hidden state h_t

$$b_t \;\approx\; h_t \;=\; f_\theta\big(h_{t-1},\, a_{t-1},\, o_t\big) \;\in\; \mathbb{R}^d. \tag{1}$$

The hidden state h_t is a function of the whole history $(o_{1:t}, a_{1:t-1})$ and compresses it into a fixed-size representation that is independent of episode length. This gives a tractable approximation of the belief and is standard in recurrent policy learning (Hafner et al., 2020; Kapturowski et al., 2018). Our goal is to learn f_θ jointly with a generative action head $\pi_\theta(\mathbf{A}_t \mid h_t)$ over action chunks $\mathbf{A}_t = (a_t, \ldots, a_{t+H-1})$ and a latent decoder that predicts future observation embeddings, which provides a world-modeling supervision for f_θ beyond imitation.

2.2 Recurrence and attention for belief representation

Recurrence and attention are two ways to represent the belief in a compact form. A recurrent approximation carries a fixed-size state and folds each new observation into it, as in Equation (1). An attention-based approximation summarizes the last W observations at every step,

$$\hat{b}_t^{\mathrm{att}} \;=\; g_\theta\big(o_{t-W+1:t},\, a_{t-W:t-1}\big), \tag{2}$$

where g_θ is typically a transformer (Vaswani et al., 2017). Frame stacking (Chi et al., 2023) is the degenerate case in which g_θ simply concatenates the raw observations in a short window, typically $W = 2$ or 3. The limits of both forms are discussed in Section 1. BIRD uses both, a recurrent state as in Equation (1) and attention over a window of those states rather than of raw observations.

Combining the complementary strengths of recurrence and attention lets the belief representation reach back to the start of the episode while still recovering information that the recurrent state has compressed away.

2.3 GENERATIVE CONTROL POLICIES

Recent work in imitation learning uses expert demonstrations $\mathcal{D} = \{(o_{1:T}, a_{1:T})\}$ to learn a generative policy π_θ that maps observations to actions. The policy is trained to maximize the likelihood of the expert actions given the observations

$$\mathcal{L}_{\mathrm{NLL}} = \mathbb{E}_{(o_{1:t},\, \mathbf{A}_t)\sim\mathcal{D}}\big[-\log \pi_\theta(\mathbf{A}_t \mid \hat{b}_t)\big]. \tag{3}$$

Predicting a chunk of H actions rather than a single action reduces the compounding error caused by test-time distribution shift (Ross et al., 2011; Zhao et al., 2023; Foster et al., 2024). Predicting an action chunk is a harder generative modeling problem than predicting a single action, which requires a more expressive policy head to model the conditional generative model over the whole chunk. We call this family of policies generative control policies (GCPs), and they share a single interface. Writing $\mathbf{A}^0$ for a demonstrated chunk, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ for noise of the same shape, and τ for a scalar time, a fixed path $\mathbf{A}^\tau$ interpolates between the two, and a network is trained to regress the local direction of that path from $\big(\mathbf{A}^\tau, \tau, c\big)$, where c is a conditioning vector computed from the observations. Actions are produced at test time by starting from noise and integrating the learned direction field. Members of the family differ in the path and in the integrator, and not in how c enters, which is what this paper changes.

Diffusion policies Chi et al. (2023) instantiate the GCP with a variance-preserving diffusion path and denoising score matching (Ho et al., 2020). With a schedule $\{\bar{\alpha}_\tau\}$ decreasing in τ, the path and the objective are

$$\mathbf{A}^\tau = \sqrt{\bar{\alpha}_\tau}\,\mathbf{A}^0 + \sqrt{1-\bar{\alpha}_\tau}\,\boldsymbol{\epsilon}, \qquad \mathcal{L}_{\mathrm{diff}} = \mathbb{E}_{\tau,\boldsymbol{\epsilon},\mathbf{A}^0}\big\|\boldsymbol{\epsilon} - \epsilon_\theta(\mathbf{A}^\tau, \tau, c)\big\|^2, \tag{4}$$

and the reverse chain is run at test time with denoising diffusion implicit models (DDIM) (Song et al., 2021). This is the formulation used by most visuomotor policies built on the recipe (Ze et al., 2024; Octo Model Team et al., 2024; Torne et al., 2025)

3 APPROACH

Overview BIRD combines recurrence and attention to learn a belief representation under partial observability. Recurrence compresses the full history into a fixed-size belief representation, and attention summarizes a window of these recurrent beliefs into a single vector that conditions the GCP. Concretely, the model shown in Figure 2 builds this conditioning in three stages. A masked autoencoder (MAE), pretrained on the demonstration frames and then frozen, maps each visual observation o_t to an embedding e_t. Then a gated recurrent unit (GRU) consumes a pooled vector $\tilde{e}_t$ of that embedding and the previous action a_{t-1} to produce a deterministic state h_t, which is paired with a discrete stochastic latent z_t to form the belief latent $\hat{b}_t = (h_t, z_t)$. Next, a lightweight decoder predicts future MAE embedding patches from $\hat{b}_t$ and the intervening actions with a world-modeling objective. This objective is trained jointly with behavior cloning, and is the only training signal on the recurrence beyond imitation. Finally, a cross-attention module summarizes a window of recent beliefs into one vector, which conditions the GCP's denoiser.

Encoder Pretraining Learning vision encoders in recurrent backbones is a challenging problem due to the computational cost of backpropagating through time, in addition to the memory challenge of training over a window of visual observations. We address this issue by pretraining the vision encoder using a masked autoencoding objective, following the approach of Seo et al. (2023). Each frame is encoded by a vision transformer ϕ (Dosovitskiy et al., 2021) into L patches.

$$e_t \;=\; \phi(I_t) \;\in\; \mathbb{R}^{(L+1)\times d_{\mathrm{tok}}}. \tag{5}$$

The recurrent core does not consume this grid directly. A parameter-free readout pools it into the mean over all tokens and a spatial softmax over the patch positions, which returns the expected image coordinate of every token channel, and the result is concatenated with an MLP embedding

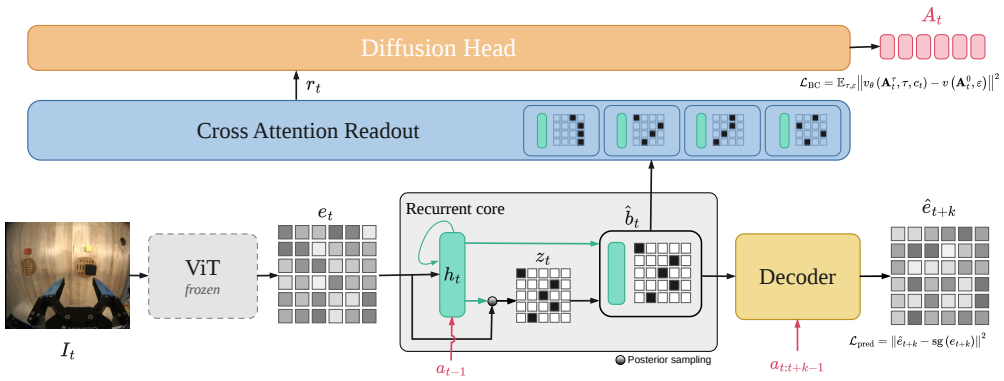


Figure 2: **Overview of BIRD.** BIRD conditions a diffusion policy on a learned belief representation in three stages: (1) a frozen MAE encoder embeds each frame I_t, (2) a recurrent core folds the embedding into the belief $\hat{b}_t$, and (3) a cross-attention readout summarizes a window of recent beliefs into a single vector r_t that conditions the diffusion head. During training, a decoder predicts future embeddings from $\hat{b}_t$ to supervise the recurrence.

of the proprioception into the vector $\tilde{e}_t$ that enters the recurrent core. ϕ is trained by masked autoencoding (He et al., 2022). This allows us a fixed target for our world modeling objective that is lighter to predict than full images. In addition the visual representation is more capable of capturing visual cues and structure. We discuss the architecture of the masked autoencoder in more detail in the appendix.

Belief Representation The belief representation is learned similar to the latent states learned in the recurrent state space models (RSSMs) used in the Dreamer family of models (Hafner et al., 2020; 2021). The recurrent core is a gated recurrent unit (Cho et al., 2014) whose linear maps are block diagonal. At each step, the recurrent unit updates the deterministic state h_t from the previous state h_{t-1}, the pooled observation embedding $\tilde{e}_t$, and the previous action a_{t-1}. A discrete stochastic state z_t is then sampled from a distribution conditioned on h_t and $\tilde{e}_t$. Unlike the RSSM, BIRD has no prior and no KL term, so z_t acts as a stochastic discrete bottleneck rather than as a posterior. The belief is represented by concatenating z_t and h_t along the feature dimension,

$$h_t = f_{\text{rec}}\big(h_{t-1},\, \tilde{e}_t,\, a_{t-1}\big), \qquad \text{recurrent unit} \tag{6}$$

$$z_t \sim q\big(\cdot \mid h_t,\, \tilde{e}_t\big), \qquad \text{stochastic state} \tag{7}$$

$$\hat{b}_t = \big[\,\text{flatten}(z_t)\,;\, h_t\,\big]. \qquad \text{belief} \tag{8}$$

On an episode boundary h_{t-1} and a_{t-1} are zeroed, so every post-reset state is a pure function of the observations from the reset onward.

We train the recurrence with truncated backpropagation through time, which runs the network over windows of limited length without resetting the state between them, so the belief can still carry information across the episode. Each window starts from the belief stored in the replay buffer for its first frame, which every update writes back and a periodic full-episode pass refreshes. The stochastic state is a set of categorical variables over K classes, and sampling them yields a discrete sparse representation of the kind used in world models and reinforcement learning (Hafner et al., 2020; 2021; Hansen et al., 2024; Lavoie et al., 2023).

Attention over Belief Representations In principle, the recurrent unit has already compressed the history into $\hat{b}_t$, and the policy could condition on it alone. In practice, however, $\hat{b}_t$ might lose some information required by the policy to solve the POMDP. We therefore treat the sequence of recent beliefs as a memory and read it out with cross-attention. The memory is a window of W beliefs, separated in time by a stride S,

$$M_t = \big(\hat{b}_{t-(W-1)S},\, \ldots,\, \hat{b}_{t-S},\, \hat{b}_t\big). \tag{9}$$

Since each $\hat{b}$ is already a running summary of everything before it, a skipped frame is not an unobserved frame but one that has been folded into the slots that remain. We use $W = 20$ and $S = 3$, so the memory spans the last 58 steps. The past beliefs in M_t are detached, so the readout's gradient reaches the recurrent core only through $\hat{b}_t$, and information that must survive longer than the window has to be carried by the recurrent state.

The readout is a single-query cross-attention between the current belief and the memory,

$$r_t = \hat{b}_t + \gamma\,\mathrm{MHA}\big(W_q\,\hat{b}_t,\, M_t,\, M_t\big), \tag{10}$$

where γ is a single learned scalar that gates how much of the retrieved history gets added to the current belief.

This design combines the strengths of recurrence and attention. The recurrent state can carry long-horizon information, while the attention readout recovers details that the current belief may have discarded.

Action head The action head is the one-dimensional convolutional U-Net of Diffusion Policy (Chi et al., 2023), which denoises the H action tokens of the chunk. Conditioning enters by feature-wise linear modulation (Perez et al., 2018). The readout of Equation (10) is concatenated with the current proprioceptive vector, rescaled to $\bar{x}_t \in [-1, 1]^{d_x}$, and with an embedding of the diffusion time τ,

$$c_t = \big[\, r_t \,;\, \bar{x}_t \,\big], \qquad \omega_t = \big[\, c_t \,;\, \mathrm{emb}(\tau) \,\big], \tag{11}$$

and every residual block of the U-Net shifts its intermediate activation by a per-channel offset read off ω_t. The same ω_t modulates every block, which is why the readout of Equation (10) has to collapse the memory into a single vector rather than hand it over as a sequence. The belief therefore conditions all H actions of the chunk identically, and the convolutions of the trunk alone resolve their ordering. Only the current step's proprioception enters Equation (11), so the head is never handed a raw W-step trajectory of proprioception and any benefit of the memory is attributable to the beliefs.

Writing $\mathbf{A}_t^0$ for the demonstrated chunk, the head is trained with the GCP objective of Section 2.3, applied at every timestep of every sampled window,

$$\mathcal{L}_{\mathrm{BC}} = \mathbb{E}_{\tau,\boldsymbol{\epsilon}}\Big[\big\|v_\theta(\mathbf{A}_t^\tau, \tau, c_t) - v(\mathbf{A}_t^0, \boldsymbol{\epsilon})\big\|^2\Big], \tag{12}$$

where v is the schedule's regression target, the noise for DDIM and the velocity for flow matching.

World modeling objective The behavior cloning loss encourages the belief to retain information that helps predict the action chunk over the next H steps, and is indifferent to everything else. This includes information whose payoff arrives much later, which is exactly what a partially observable task requires. A belief that retains a sufficient statistic for the next H actions already minimizes the imitation loss (Gangwani et al., 2020), so the imitation loss leaves unconstrained what the belief keeps beyond that horizon. We therefore add an auxiliary objective that supervises the belief directly. Let e_{t+k} be the frozen MAE embedding grid k steps ahead (Equation (5)), and let $a_{t:t+k-1}$ be the ground-truth actions over that interval. A light transformer decoder ψ predicts these future embeddings from the belief and the actions,

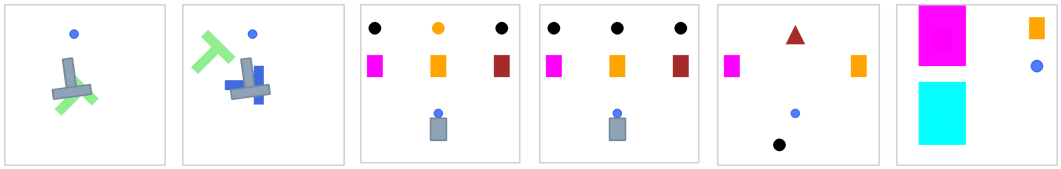
$$\hat{e}_{t+k} = \psi\big(\,[\,\hat{b}_t \,;\, a_{t:t+k-1}\,]\,\big) \in \mathbb{R}^{(L+1)\times d_{\mathrm{tok}}}, \tag{13}$$

and the loss is a masked, per-token squared error against the detached target,

$$\mathcal{L}_{\mathrm{pred}} = \big\|\hat{e}_{t+k} - \mathrm{sg}\big(e_{t+k}\big)\big\|_2^2, \tag{14}$$

where $\mathrm{sg}(\cdot)$ is the stop-gradient operator. Equation (14) maximizes a lower bound on what the belief knows about the future state. Following Gangwani et al. (2020, Eq. 9), with s_{t+k} the unobserved state k steps ahead and q a variational density,

$$\begin{aligned} I(\hat{b}_t;\, s_{t+k} \mid a_{t:t+k-1}) &\geq I(\hat{b}_t;\, e_{t+k} \mid a_{t:t+k-1}) \\ &= H(e_{t+k} \mid a_{t:t+k-1}) - H(e_{t+k} \mid \hat{b}_t, a_{t:t+k-1}) \\ &\geq H(e_{t+k} \mid a_{t:t+k-1}) + \mathbb{E}\big[\log q(e_{t+k} \mid \hat{b}_t, a_{t:t+k-1})\big], \end{aligned} \tag{15}$$



(a) Push-T (fully obs.) (b) Push-T (two goals) (c) Push-Box (visible cue) (d) Push-Box (transient) (e) Counting (f) Drawer

Figure 3: **The six evaluation tasks.** Left three fully observed, right three requiring memory. Panels (c) and (d) are the same scene at the same reset state: the colored signal circle indicates which of the three goals the box must be pushed to, and it stays lit for the whole episode in (c) but goes dark after 80 frames in (d). The policy observes 96×96.

where the first inequality is data processing along $\hat{b}_t \to s_{t+k} \to o_{t+k} \to e_{t+k}$, which holds because o_{t+k} depends on the history only through s_{t+k} and the frozen tokenizer is a fixed function of o_{t+k}, and the second is the variational bound of Agakov & Barber (2004). Taking q Gaussian with fixed variance and mean ψ recovers Equation (14) up to constants. We take the forward target from NE-Dreamer (Bredis et al., 2026), where predicting next-step encoder embeddings from the latent state gave substantial gains on DMLab tasks that involve memory and spatial reasoning.

Conditioning the head on $a_{t:t+k-1}$ is designed to make the objective a statement about memory rather than dynamics, and Section 4.3 tests this at $k = 2$. Without those actions, the bound concerns $I(\hat{b}_t; e_{t+k})$, which a head can raise by learning whatever the environment makes predictable on average, and much of that is already visible in the current frame. Self-predictive objectives in control regress a pooled target embedding from a pooled context embedding (Grill et al., 2020; Schwarzer et al., 2021), and so do not aim to maintain the environment's spatial structure. Image joint-embedding architectures predict patch-level targets (Assran et al., 2023), but from the current image and not from a recurrent state. Equation (14) instead scores every patch position separately, and so asks where rather than only whether. Because the tokenizer is frozen, that target is a fixed function of pixels, which also means Equation (14) admits no representational collapse while being lighter than a pixel decoder.

Training Once the MAE is pretrained and frozen, the rest of the model is trained jointly in a single phase on one backward pass,

$$\mathcal{L} = \lambda_{\text{BC}}\, \mathcal{L}_{\text{BC}} + \lambda_{\text{pred}}\, \mathcal{L}_{\text{pred}}. \tag{16}$$

so the belief model is shaped by both terms at once, with no further freezing schedule. The belief model, the readout, the prediction head and the action head share one optimizer, LaProp (Ziyin et al., 2020) with adaptive gradient clipping (Brock et al., 2021) and a linear warmup. The masked autoencoder has its own AdamW optimizer (Loshchilov & Hutter, 2019) and is active only during pretraining. Remaining hyperparameters are listed in Section B.

4 Results

Our experiments address three questions. First, whether a recurrent belief helps where memory is required, and whether we can avoid drop in performance in Markovian tasks associated with long context policies (De Haan et al., 2019; Lazzati et al., 2026). Second, whether a recurrent belief is better than simply using a transformer with long context, which is the other way to give a diffusion policy access to history. Third, we ablate the different components of BIRD in Section 4.3. We describe our experimental setup below and list all hyperparameters in Section B. Every method is trained offline for 10^5 gradient steps and evaluated every 5×10^3 steps by rolling out 50 episodes. We use the best evaluation checkpoint, as is standard in imitation learning (Mandlekar et al., 2021; Chi et al., 2023), and report the mean and the standard deviation across three seeds.

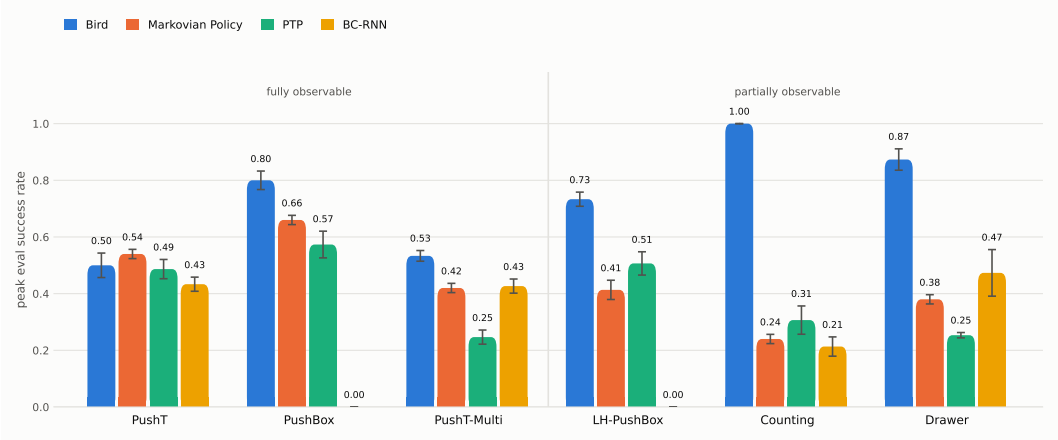


Figure 4: BIRD outperforms every baseline on the three partially observable tasks, and on the fully observable ones it leads or is within noise of the best baseline. Peak success rate on the six tasks, split into the three that are fully observable (left) and the three that are not (right). Bars are the mean over three seeds and error bars one standard deviation. The Markovian Policy is Chi et al. (2023), PTP is Torne et al. (2025), and BC-RNN is Mandlekar et al. (2021). BC-RNN scores 0.00 on every seed of both Push-Box variants, the fully observable one included. Push-Box actions are close to an extrapolation of the agent's own motion, which a one-step policy can fit and which stalls once the agent is at rest.

4.1 Experimental Setup

Tasks We evaluate on six simulated 2D manipulation tasks built on the Push-T environment (Florence et al., 2022; Chi et al., 2023), shown in Figure 3 and chosen so as to have partially observable tasks as well as fully observed ones as control. We describe the environments in more detail in Section A.

Baselines We compare against two diffusion policies that are trained on the same demonstrations with the same action chunking, sampler and replan rate as BIRD, and that differ from it in how history reaches the denoiser as well as in their encoder and auxiliary objective, plus a third baseline run as published. **Markovian Policy** is Diffusion Policy (Chi et al., 2023): a ResNet-18 observation encoder and a 1D convolutional U-Net denoiser conditioned on the current observation. We use a single frame, where the published configuration stacks two, so that this memoryless reference point has no access to history. **PTP** is the past-token-prediction policy of Torne et al. (2025): a transformer denoiser with conditioning tokens that are per-frame image encodings over a window of the last 16 observations, with the past-token prediction auxiliary loss enabled. This is the alternative of lengthening the window instead of using recurrence. **BC-RNN** is the recurrent behavioral cloning policy of Mandlekar et al. (2021), run with the image configuration published with that work and built from its reference implementation: a ResNet-18 encoder with a spatial-softmax readout, a two-layer LSTM with 1000 hidden units, and a five-mode Gaussian mixture head that predicts one action per step.

4.2 Results

Figure 4 shows the performance of each method on the six tasks. BIRD is the best method on every task except Push-T, where it is within noise of the Markovian Policy. We find that conditioning on a recurrent belief does not cost performance where a reactive policy already suffices, even though the belief replaces the policy's visual encoder entirely and the policy conditions on a longer history, the setting in which history-conditioned policies are prone to overfit (De Haan et al., 2019; Lazzati et al., 2026). BIRD also leads on Push-Box with a visible cue and on Push-T (two goals), where no memory is required, so part of its advantage does not come from memory. Its margin over the strongest baseline grows from at most 0.14 on the fully observable tasks to between 0.22 and 0.69 on the partially observable ones. The ablations in Section 4.3 point to the predictive loss, whose

	BIRD	No attention	No stoch. latent	No forward prediction	No predictive loss	Conv. encoder
Fully observable	0.65 ± 0.03	0.66 ± 0.04	0.61 ± 0.02	0.72 ± 0.02	0.49 ± 0.04	0.66 ± 0.02
Partially observable	0.80 ± 0.03	0.61 ± 0.05	0.82 ± 0.02	0.78 ± 0.04	0.58 ± 0.17	0.81 ± 0.04

Table 1: **Ablations.** Peak success rate, averaged over the two tasks of each group (Section 4.3). Errors are one standard deviation across seeds. Each column is BIRD with a single component removed. No attention sets $W = 1$, so the denoiser sees the current belief alone. No stochastic latent keeps the deterministic state only, at matched belief width. No forward prediction scores the tokenizer grid at the current step instead of k steps ahead and drops the action-conditioned head. No predictive loss removes the auxiliary objective entirely. Conv. encoder replaces the frozen tokenizer with a convolutional encoder trained end to end.

removal lowers the fully observable group from 0.65 to 0.49. PTP trails the Markovian Policy on all three fully observable tasks. It differs from the other diffusion policies in its denoiser, a transformer rather than the U-Net used by BIRD and the Markovian Policy, and Chi et al. (2023) report the transformer denoiser to be more sensitive to hyperparameters.

The two Push-Box variants differ only in whether the cue persists, so the drop between them isolates the cost of having to remember. Every diffusion policy loses performance once the cue disappears, and BIRD is the only method whose transient-cue score stays above every baseline's visible-cue score. The two diffusion baselines also swap places across the pair, so a benchmark containing only one variant would support the opposite conclusion about which baseline is stronger. BC-RNN fails on both variants, including the fully observable one. The separation is largest where the demonstrations are not Markovian at all. On Counting and Drawer, BIRD leads every baseline by a wide margin, and no baseline is consistently the runner-up. BC-RNN also carries a recurrent state and trails BIRD on both, but it differs from BIRD in more than its memory, with a single-step Gaussian mixture head and no auxiliary loss. The No attention ablation in Section 4.3 isolates the recurrent belief within BIRD.

4.3 Ablations

We run the ablations on four tasks, Push-T and Push-Box with a visible cue as the fully observable pair and Push-Box with a transient cue and Drawer as the partially observable pair, so each arm is measured both where memory is required and where it is not. We omit Counting, where BIRD is saturated at 1.00.

No attention conditions the denoiser on the single current belief ($W = 1$ in Equation (9), no readout of Equation (10)), so everything the policy sees must have survived in the recurrent state. It matches BIRD on the fully observable tasks, 0.66 against 0.65, and drops from 0.80 to 0.61 on the partially observable ones, a loss comparable to removing the predictive loss. We find that the recurrent state is not sufficient on its own at this training budget where the task requires memory. It still outperforms every baseline on the partially observable pair, where the strongest baseline averages 0.40 (Figure 4). **No stochastic latent** keeps only the deterministic state, widened so the belief keeps its original dimension. It trails BIRD on the fully observable tasks, 0.61 against 0.65, and matches it on the partially observable ones, 0.82 against 0.80. We find that at matched width the stochastic latent contributes little. **No forward prediction** targets the tokenizer grid at the current step and not at future time steps, as in Seo et al. (2023). On the fully observable tasks it outperforms BIRD, 0.72 against 0.65, and on the partially observable tasks it is within noise, 0.78 against 0.80. We find that at the horizon we use, $k = 2$, the benefit of the predictive loss comes from predicting the token grid from the belief rather than from predicting it ahead in time. Whether longer horizons reward retention where memory is required, as next-embedding prediction does in NE-Dreamer (Bredis et al., 2026), is left to future work. **No predictive loss** trains the belief from the behavioral cloning loss alone, and degrades both groups, from 0.65 to 0.49 on the fully observable tasks and from 0.80 to 0.58 on the partially observable ones. We find that the predictive objective is necessary even where no memory is required, which suggests that it also shapes what the belief encodes about the current scene and not only what it retains. **Conv. encoder** replaces the frozen masked autoencoder with a convolutional encoder trained end to end under a latent prediction loss. It matches BIRD on

both groups, 0.66 against 0.65 on the fully observable tasks and 0.81 against 0.80 on the partially observable ones. We find that the frozen tokenizer does not drive performance, and its benefit is the one that motivates it in Section 3, removing the vision encoder from backpropagation through time.

5 Related Work

Imitation learning for manipulation Behavioral cloning remains the dominant paradigm for learning manipulation from demonstrations (Chi et al., 2023; Simchowitz et al., 2025; Barreiros et al., 2026), with a long line of work on its susceptibility to compounding error. DAgger (Ross et al., 2011) queries the expert online, adversarial imitation (Ho & Ermon, 2016) matches occupancy measures, and energy-based formulations (Florence et al., 2022) address the multimodality of the expert conditional. ACT (Zhao et al., 2023) showed that predicting temporally extended action chunks substantially dampens error accumulation, and Diffusion Policy (Chi et al., 2023) replaced regression with a conditional denoising model, an approach that now scales to generalist policies (Black et al., 2024). We adopt chunked diffusion and change how the policy is conditioned, so that it can solve tasks that require long-horizon memory.

Partial observability and history-conditioned policies Partial observability is the norm in manipulation, as objects are occluded, cues are transient, and the robot's own past encodes state that no frame can recover. The POMDP literature resolves this with belief states (Kaelbling et al., 1998; Ding et al., 2014), which recurrent policies approximate with a learned hidden state. R2D2 (Kapturowski et al., 2018) established recipes for training recurrent agents from replay, and Dreamer (Hafner et al., 2020; 2021; 2023) turned the recurrent state estimator into a world model used for imagination. In imitation learning, recurrent encoders have been applied to memory-dependent and multi-stage manipulation (Jia et al., 2024; Zhou et al., 2025). MaIL is evaluated on fully observable benchmarks, and MTIL encodes the full history to resolve temporal ambiguity but trains it by imitation alone with a regression head. A recent alternative avoids recurrence altogether and simply lengthens the observation window, adding a past-token prediction loss to stop the extra tokens from being ignored (Torne et al., 2025). We extend previous work by combining recurrence and attention to learn a belief representation that can condition a GCP to handle partial observability.

Predictive and self-supervised auxiliary objectives Predicting the future has long been used as a learning signal for control (Finn & Levine, 2017; Ebert et al., 2018), but pixel-space prediction is expensive and models detail that the policy does not need. Latent self-prediction, as in SPR (Schwarzer et al., 2021), TD-MPC (Hansen et al., 2022; 2024), and the joint-embedding objectives they inherit from (Grill et al., 2020), avoids this but must manage representational collapse via target networks or stop-gradients. Masked world models (Seo et al., 2023) decouple representation learning from dynamics by training a masked autoencoder with its own optimizer and having the recurrent state reconstruct its token grid. Our predictive head is a direct descendant, differing in that the target is k steps ahead and conditioned on the intervening actions.

6 Conclusions and Limitations

We presented BIRD, a diffusion policy for manipulation under partial observability that conditions action generation on a recurrent belief over the observation history using a cross-attention mechanism. On tasks where partial observability is introduced in a controlled way, belief conditioning outperforms every baseline on the tasks that require memory. We also showed that a predictive auxiliary objective in latent space is necessary to shape the belief representation. Furthermore we found that the recurrent belief can be used without loss on fully observable tasks, so the same architecture can be applied to both.

Limitations The main limitation of our method is the high computational cost of training a recurrent belief model, which requires backpropagation through time. This makes it difficult to scale to very long sequences or very large models. Further, the replay buffer and sampling strategy could be improved to better handle long-horizon tasks, for example with memoroids (Morad et al., 2024).

ETHICS STATEMENT

This work involves no human subjects and no personal data: every result is on simulated manipulation benchmark, built on public, freely available datasets and simulators.

AI USE STATEMENT

We used generative AI tools to polish writing, improve readability, and suggest structure for this paper, to create and edit software code, and to create or modify figures; every such use was reviewed by the authors. We did not use generative AI to generate any dataset or synthetic data, or to formulate or prove any mathematical claim in this paper; the remaining disclosure categories in the call for papers do not apply to this work. All AI-assisted code was verified by running the experiments reported in this paper, and all AI-assisted text was reviewed and edited by the authors. The authors take full responsibility for the content of this paper, including all text, claims, and artifacts produced with AI assistance.

REPRODUCIBILITY STATEMENT

We will release our implementation, configurations, and launch commands publicly upon publication. Every figure and table in the paper regenerates directly from logged training metrics via plotting scripts in that repository. Each training run's log records its complete command line, making any reported run's exact configuration recoverable.

REFERENCES

Felix Agakov and David Barber. Variational information maximization for neural coding. In *International Conference on Neural Information Processing*, pp. 543–548. Springer, 2004.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11 (113):eaea6201, 2026.

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

George Bredis, Nikita Balagansky, Daniil Gavrilov, and Ruslan Rakhimov. Next embedding prediction makes world models stronger. *arXiv preprint arXiv:2603.02765*, 2026.

Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning (ICML)*, 2021.

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.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1724–1734, 2014.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. volume 32, 2019.

Xuchu Ding, Stephen L Smith, Calin Belta, and Daniela Rus. Optimal control of markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 59(5): 1244–1257, 2014.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Frederik Ebert, Sudeep Dasari, Alex X. Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. In *Conference on Robot Learning (CoRL)*, 2018.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

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.

Dylan J Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. volume 37, pp. 120602–120666, 2024.

Tanmay Gangwani, Joel Lehman, Qiang Liu, and Jian Peng. Learning belief representations for imitation learning in pomdps. In *uncertainty in artificial intelligence*, pp. 1061–1071. PMLR, 2020.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. volume 29, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Xiaogang Jia, Qian Wang, Atalay Donat, Bowen Xing, Ge Li, Hongyi Zhou, Onur Celik, Denis Blessing, Rudolf Lioutikov, and Gerhard Neumann. Mail: Improving imitation learning with mamba. 2024.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Samuel Lavoie, Christos Tsirigotis, Max Schwarzer, Ankit Vani, Michael Noukhovitch, Kenji Kawaguchi, and Aaron Courville. Simplicial embeddings in self-supervised learning and downstream classification. In *International Conference on Learning Representations (ICLR)*, 2023.

Filippo Lazzati, Kyle Stachowicz, William Chen, Alberto Maria Metelli, Andrew Wagenmaker, and Sergey Levine. Why does action chunking improve behavioral cloning performance in robotic control? *arXiv preprint arXiv:2608.02547*, 2026.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

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. 2021.

Steven Morad, Chris Lu, Ryan Kortvelesy, Stephan Liwicki, Jakob Foerster, and Amanda Prorok. Recurrent reinforcement learning with memoroids. *Advances in Neural Information Processing Systems*, 37:14386–14416, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations (ICLR)*, 2021.

Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning (CoRL)*, 2023.

Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous. *arXiv preprint arXiv:2503.09722*, 2025.

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

Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. *arXiv preprint arXiv:2505.09561*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *Robotics: Science and Systems (RSS)*, 2024.

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

Yulin Zhou, Yuankai Lin, Fanzhe Peng, Jiahui Chen, Kaiji Huang, Hua Yang, and Zhouping Yin. Mtil: Encoding full history with mamba for temporal imitation learning. *IEEE Robotics and Automation Letters*, 2025.

Liu Ziyin, Zhikang T. Wang, and Masahito Ueda. LaProp: Separating momentum and adaptivity in Adam. *arXiv preprint arXiv:2002.04839*, 2020.

A Task Descriptions

Partial observability is introduced in a controlled way rather than as an uncontrolled side effect of the scene. All tasks use 96×96 RGB observations plus the 2D end-effector position, and a 2D positional action. Push-T is the standard fully observed benchmark, which we include as a control because nothing in it requires history. Push-T (two goals) asks for two target poses in sequence under a gated reward, with both drawn on screen for the whole episode, so it stresses horizon rather than memory and which stage is active stays readable from the block's own position. Push-Box places three candidate goals on screen, each paired with a signal circle, and colors the one circle that indicates where the box must be pushed. Its transient-cue variant is pixel-for-pixel identical except that the active circle goes dark after the first 80 rendered frames, so the goal is afterwards recoverable only from memory and the gap between the two variants measures what a policy's memory is worth. In Counting the agent must visit three waypoints in the repeating order $[0, 2, 0, 2, 1]$, and because two of them are visited twice with nothing on screen distinguishing the first pass from the second, the progress count has to be carried internally; missing the next waypoint within 32 steps returns the agent to the start circle and resets the count. In Drawer two covered drawers each hide a block of unknown color, a target patch shows which block color must be pushed onto it, and the agent has to approach a drawer to reveal what it hides, remember that, and only then commit, since peeking without committing sends it back to the marker it started on.

B Implementation Details

All hyperparameters are listed below. Values are shared across every task unless stated otherwise.

B.1 Visual tokenizer

Component	Hyperparameter	Value
Input	image resolution	$96 \times 96 \times 3$
	proprioception	2-D end-effector position
MAE encoder	patch size	8 (via 3 stride-2 convolutions)
	token grid $g \times g$	12×12 ($L = 144$, $+1$ class token)
	embedding width d_{tok}	256
	depth / heads	4 / 4
MAE decoder	embedding width	256
	depth / heads	3 / 4
	mask ratio ρ	0.75
	extra prediction token	action
	normalized-pixel loss	off
	loss over	all patches
Pretraining	optimizer	AdamW
	learning rate / ϵ	3×10^{-4} / 10^{-5}
	weight decay / grad. clip	10^{-6} / 100
	warmup steps	2500
	pretraining steps N_{pre}	10^4 (then frozen)
Readout	token mean	enabled (256 dims)
	spatial softmax	enabled (512 dims, 0 parameters)
	temperature β	0.5
	proprioception MLP	3 layers, 256 units, SiLU, RMSNorm
	embedding width $\tilde{e}_t$	1024

B.2 RECURRENT BELIEF MODEL

Component	Hyperparameter	Value
Recurrent core	type	block-diagonal GRU
	deterministic state h_t	2048
	blocks	8
	hidden width	256
	transition layers	1
	activation / normalization	SiLU / RMSNorm
Stochastic state	inputs	h_t and $\tilde{e}_t$
	categorical variables	32
	classes per variable K	16
	layers	1
	uniform mixing ratio	0.01
	gradient estimator	straight-through
Belief	width $\hat{b}_t$	2560 ($2048 + 32 \times 16$)
	prior / KL term	none
Prediction head	type	flat vision-transformer decoder
	embedding width	128
	depth / heads	2 / 4
	prediction horizon k	2
	same-timestep head	disabled
	target	detached MAE token grid at $t + k$
	conditioning	$\hat{b}_t$ and $a_{t:t+k-1}$

B.3 ACTION HEAD

Component	Hyperparameter	Value
Conditioning	belief window W	20
	window stride S	3
	history slots	detached
	readout heads / width	4 / 512
	conditioning width	2562 (2560 readout + 2 proprioception)
Denoiser	trunk	1D convolutional U-Net (Chi et al., 2023)
	channel widths	512 / 1024 / 2048
	kernel size / groups	5 / 8
	conditioning	FiLM (Perez et al., 2018)
	diffusion-time embedding	128
Generative	formulation	flow matching (Lipman et al., 2023)
	action chunk H	16
	executed actions n_{act}	8
	train timesteps	100
	inference steps	16

B.4 OPTIMIZATION

The belief model, the attention readout, the prediction head and the denoiser share one optimizer on a single backward pass; the masked autoencoder has a second, active only during pretraining (above).

Component	Hyperparameter	Value
Shared optimizer	optimizer	LaProp
	learning rate	4×10^{-5}
	$\beta_1,\ \beta_2,\ \epsilon$	0.9, 0.999, 10^{-20}
	adaptive gradient clipping	0.3 (min. param. norm 10^{-3})
	warmup steps	1000 (linear, then constant)
Loss scales	λ_{BC}	1.0
	λ_{pred}	1.0
Batching	windows per batch	16
	window length T	64
	action-head samples per update	256
	end-of-episode padding	16 frames ($= H$)
Schedule	gradient steps	10^5
	evaluation interval	5×10^3
	evaluation episodes	50
	precision	bfloat16 autocast

B.5 Baseline configurations

The Markovian Policy and PTP share BIRD's diffusion recipe (flow matching, $H = 16$, $n_{\mathrm{act}} = 8$, 16 inference steps), demonstrations, training budget and evaluation protocol, so those comparisons hold the diffusion recipe fixed. BC-RNN shares the demonstrations, training budget and evaluation protocol but not the diffusion recipe. It is run with the image configuration of Mandlekar et al. (2021), including its unclipped gradients, so its comparison with BIRD is between whole methods. It differs from that configuration only where our tasks force it to: a single 96×96 camera in place of two, and actions scaled to $[-1, 1]$ by the dataset range rather than standardized.

Method	Hyperparameter	Value
Markovian Policy	observation encoder	ResNet-18, $96 \to 76$ random crop
	denoiser	1D convolutional U-Net
	observation window	1 (current frame)
PTP	observation encoder	ResNet-18, $96 \to 76$ random crop, per frame
	denoiser	transformer, 8 layers / 4 heads / 256 width
	observation window	16
	past-action slots	15
	past-token prediction	enabled (all past slots)
	weight EMA	enabled
BC-RNN	observation encoder	ResNet-18, $96 \to 76$ random crop
	visual readout	spatial softmax, 32 keypoints
	recurrent core	LSTM, 2 layers / 1000 hidden
	action head	5-mode Gaussian mixture, 1 action per step
	sequence length	10
	optimizer	Adam, 10^{-4}, batch 100, unclipped