Shaping Dynamics Models via Future State Sampling for Improving Model-Based Reinforcement Learning

Benteng Ma
King's College London
London, United Kingdom
benteng.ma@kcl.ac.uk

David Watson
King's College London
London, United Kingdom
david.watson@kcl.ac.uk

Matteo Leonetti
King's College London
London, United Kingdom
matteo.leonetti@kcl.ac.uk

ABSTRACT

In model-based reinforcement learning, low prediction error from learned dynamics models does not necessarily lead to effective planning or policy improvement. Prior work shows that dynamics models trained with purposefully shaped prediction targets can substantially improve policy learning, even when their predicted transition distribution differs from the true environment dynamics. Additionally, multi-step models demonstrate that access to longer-horizon outcomes can enhance planning beyond purely one-step predictions. Motivated by these findings, our approach augments dynamics-model training with additional transition samples constructed from historical trajectory segments, enabling the model to assign probability to future reachable states without introducing additional inputs or model variables. We introduce two mechanisms for generating such augmented supervision targets and evaluate the approach in tabular domains. Results show that the approach can speed up policy learning, while ablations indicate that the benefit depends on structured future-state selection and moderate mixing of folded supervision during training.

KEYWORDS

Model-Based Reinforcement Learning, Dynamics Model Learning, Transition Shaping

1 INTRODUCTION

Model-based reinforcement learning (MBRL) relies on a learned dynamics model to support planning or policy updates [8–10, 17]. A common objective is to minimise the prediction error with respect to the true environment dynamics. However, predictive precision alone does not guarantee effective policy improvement. Previous work shows that dynamics models trained under purposefully shaped or biased supervision can produce better policies than models trained to match the exact true transition distribution [6, 19]. Moreover, complementary evidence from multi-step prediction suggests that exposing longer-horizon outcomes can improve planning beyond purely one-step models [3, 15]. Taken together, these results suggest that the usefulness of a dynamics model depends not only on predictive fidelity, but also on how its predictions shape the distribution of model-generated transitions encountered during rollouts; in particular, predictions that increase the likelihood of encountering transitions that provide informative learning signals or maintain strong long-horizon predictive accuracy by reducing compounding error.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25 – 26, 2026, Paphos, Cyprus, https://alaworkshop2026.github.io/. 2026.

Motivated by this perspective, we augment dynamics-model training using replay data. We sample short trajectory segments (i.e., temporally ordered transitions) from the replay buffer and, for each state-action pair, select a future state within the segment as a jump target, namely a later observed state treated as if it were the immediate one-step successor of the current state–action pair, according to a sampling distribution over candidate future states. This produces folded transition tuples, where multi-step transitions are collapsed into single-step targets by skipping intermediate states and aggregating rewards. The resulting tuples are stored in an auxiliary buffer and mixed with real one-step transitions during training. The model thus retains the standard one-step interface, conditioning only on the current state and action, while assigning probability mass to later reachable outcomes (Fig. 1).

This change in supervision alters the transition distribution learned by the model. Under stochastic dynamics or suboptimal early policies, high-value states may be difficult to reach reliably through standard one-step rollouts, and thus receive insufficient training signal. Our approach increases the probability of such future-reachable outcomes by constructing folded transitions, allowing them to be treated as plausible successors of the same state-action input while preserving the standard one-step model interface. As a result, model rollouts can reach these states more readily without requiring any modification to the underlying model-based RL algorithm (Fig. 2).

We define sampling distributions over candidate future states within each trajectory segment to generate these additional supervision targets (Sec. 4.2.1). This framework is instantiated with two mechanisms that control which future outcomes are incorporated while preserving behavioural consistency. We evaluate the resulting shaped dynamics models in tabular domains (Sec. 5), testing whether the approach improves policy learning and how performance depends on the folding design. The contributions are listed as follows:

- We introduce a data-distribution shaping method that augments dynamics-model training with transition targets from collected rollouts, enabling probability mass on later reachable states while preserving the one-step prediction form.
- We propose two mechanisms to control future state selection and shape the learned transition distribution.
- We show improved policy learning in tabular domains and provide ablations on sampling design and data mixing.

2 RELATED WORK

Our work mainly relates to two directions: (i) dynamics-model training aimed at policy improvement, and (ii) methods that expose longer-horizon outcomes.

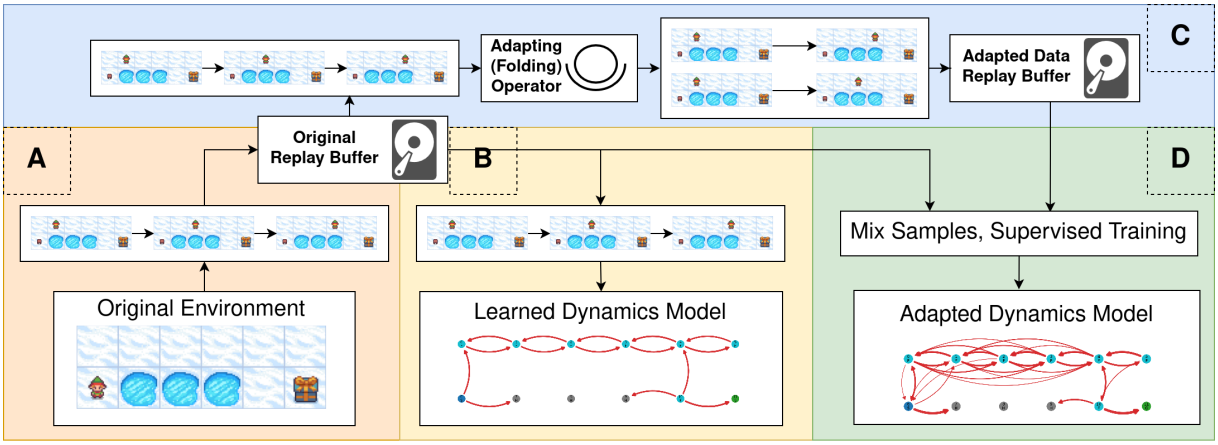


Figure 1: The procedure of learning such a dynamics model includes – Section A: Sampling trajectories from the real environment and saving them in the raw replay buffer; Section B: to train a standard dynamics model, data will only be sampled from the raw buffer; Section C and D: to train the model predicting the shaped dynamics, first sample real transitions from the raw buffer, then generate the folded data following the distribution from folding operator. The new data will be saved in an alternative replay buffer and will be mixed with the real data to train the shaped model.

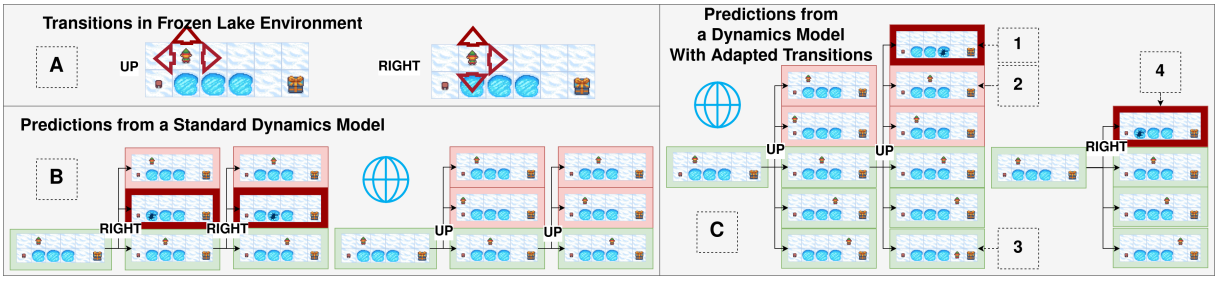


Figure 2: Illustration of the true and desired dynamics-model behaviour in the FrozenLake environment. The cells correspond to example outcomes shown in the figure. (A) True environment dynamics: each action moves in the intended direction with probability $1/3$, or slips to one of the two perpendicular directions with probability $1/3$ each; hitting a wall leaves the agent in place, and entering a hole terminates the episode. (B) A standard dynamics model reproduces these immediate stochastic outcomes, so rollouts tend to remain near the current state and may repeatedly encounter unfavourable outcomes such as 1, 2, and 4. (C) Our objective is to train a model whose learned transition distribution also assigns probability to later reachable states observed along trajectories (3), allowing rollouts to reach goal-relevant regions more efficiently while still representing possible failures (1 and 4).

Dynamics models trained for policy improvement. Prior work has modified how dynamics models are trained in order to improve downstream policy learning. Joint model–policy optimisation links model updates directly to policy-improvement objectives [6], and other approaches similarly incorporate value or uncertainty signals by changing training losses or planning procedures [11, 18, 20]. A complementary line reshapes the effective training distribution via policy-dependent data weighting while keeping the model form unchanged [19]. In contrast to objective- or loss-based methods, our approach does not change the model architecture or its input–output interface. Unlike weighting-based approaches that only reweight existing transitions, we augment supervision data by constructing additional transition targets from trajectory segments and including these constructed tuples during training.

Exposing longer-horizon outcomes. Several approaches expose longer-term consequences by predicting states at coarser temporal resolutions. Adaptive skip-interval models train predictors to output states a fixed or adaptively chosen number of steps ahead, effectively redefining the prediction horizon of the dynamics model [15]. Hierarchical predictors similarly operate over subsampled trajectories or multi-scale state sequences rather than immediate successors [3]. A complementary line introduces temporal abstraction on the action side–for example, options, skills, or action chunking–so that a single decision executes multiple environment steps [2, 5, 13, 14, 16]. Unlike these approaches, we do not redefine the dynamics model as a fixed-k-step or multi-scale predictor, nor do we introduce macro-actions or an extended action interface. Instead, we keep the model conditioned only on the current state and

action, while augmenting supervision so that later reachable states observed in trajectories can also be learned as possible successor outcomes.

3 PRELIMINARIES

3.1 Markov decision processes

We consider a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $P(s' \mid s, a)$ is the transition kernel, $r(s, a, s')$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. We denote the one-step outcome tuple as (s', r', done'), where $\text{done}' \in \{0, 1\}$ indicates termination.

3.2 Learned dynamics models

We use a learned dynamics model that represents a conditional distribution over outcome tuples

$$p_\theta(y \mid s, a), \tag{1}$$

where $y = (s', \bar{r}, \text{done})$ contains the successor state, a reward label stored in supervision tuples, and a termination flag. For real transitions, $\bar{r} = r'$; while under folded supervision (Sec. 4.2), $\bar{r}$ replaces multi-step rewards with a single-step label, typically computed as a discounted sum along the corresponding trajectory segment.

4 METHOD

4.1 Overview: shaping transition predictions via folded supervision

We build on the learned dynamics model defined in Eq. (1), which represents a conditional distribution over outcome tuples $y = (s', \bar{r}, \text{done})$ given (s, a).

The transition behaviour learned by the model depends on how supervision tuples are constructed and sampled during training. We instantiate this principle by constructing additional supervision tuples from short trajectory segments via a stochastic folding operator and mixing them with standard one-step transitions.

4.2 Segment-folding supervision from sequential data

We assume access to sequential data (either from online interaction or a replay buffer), so that trajectory segments can be sampled. As illustrated in Fig. 1, our training procedure augments the replay buffer with additional supervision tuples constructed from these segments. For each state-action pair (s_t, a_t) in a segment, we sample a future state from the same trajectory and assign a folded reward label (i.e., weighted accumulated reward from the starting state to this later state), forming an additional supervision tuple with the same input structure as standard one-step transitions. The dynamics model is then trained on the mixture of real and folded tuples (algorithm details discussed in Sec. 4.3).

Formally, this construction is defined through a stochastic folding operator $\mathcal{F}(\tau_{t:t+K})$ that maps a sampled trajectory segment $\tau_{t:t+K}$ to a supervision tuple. We present two concrete instantiations of this operator. The first restricts the *support* of admissible future states using state abstraction (Sec. 4.2.3), while the second reweights the *density* over candidate future states using value information (Sec. 4.2.4).

4.2.1 *Folded target construction via segment folding.*

Let a sampled trajectory segment of length K starting at time t be

$$\tau_{t:t+K} \triangleq \Big(\big(s_{t+i},\, a_{t+i},\, r_{t+i},\, \text{done}_{t+i+1}\big)_{i=0}^{K-1},\, s_{t+K} \Big), \tag{2}$$

The folding operator $\mathcal{F}(\tau_{t:t+K})$ constructs a folded target by selecting a future-step offset $\kappa \in \{1, \ldots, K\}$ from the sampled segment. We write the selection distribution as

$$q_{\text{fold}}(\kappa \mid \tau_{t:t+K}), \qquad \kappa \in \{1, \ldots, K\}, \tag{3}$$

where q_{fold} specifies how likely each candidate future position is to be selected.

The sampled offset determines the folded target

$$y_t \triangleq \big(s'_t,\, \bar{r}_t,\, \text{done}'_t\big), \tag{4}$$

with

$$s'_t \triangleq s_{t+\kappa}, \tag{5}$$

$$\text{done}'_t \triangleq \text{done}_{t+\kappa}, \tag{6}$$

$$\bar{r}_t \triangleq \text{FoldR}(\tau_{t:t+\kappa}), \tag{7}$$

where $\bar{r}_t$ summarises rewards along the selected sub-segment. In this work, we use discounted accumulation with a folding discount factor $\bar{\gamma} \in [0, 1]$:

$$\text{FoldR}(\tau_{t:t+\kappa}) \triangleq \sum_{i=0}^{\kappa-1} \bar{\gamma}^{\,i} r_{t+i}. \tag{8}$$

Because κ may exceed 1, the successor-state component s'_t may correspond either to the immediate next state or to a later reachable future state from the same trajectory. The resulting supervision therefore preserves the standard prediction format of Eq. (1) while allowing the learned model to represent both one-step and multi-step reachable outcomes.

4.2.2 *Model fitting under mixed supervision.*

The folded supervision tuples generated by the folding operator are collected into a replay buffer, denoted by $\mathcal{D}_{\text{fold}}$. Let $\mathcal{D}_{\text{raw}}$ denote the real one-step transition replay buffer. During training, mini-batches are formed by mixing samples from $\mathcal{D}_{\text{raw}}$ and $\mathcal{D}_{\text{fold}}$, using a folded-sample ratio $\rho(\ell) \in [0, 1]$, where ℓ denotes the training iteration and $\rho(\ell)$ may be fixed or scheduled over training.

The dynamics model is trained using the same predictive objective as in standard dynamics modelling, without modifying the model architecture or optimisation procedure; our method acts purely by augmenting the supervision data seen during training.

4.2.3 *Segment folding with state abstraction.*

Our first proposed folding mechanism guides folded-target selection by exploiting local behavioural similarity within a trajectory segment. Inspired by previous work applying state abstraction as a trade-off between compression and the preservation of decision-relevant structure [1], the intuition is that if consecutive timesteps exhibit similar policy behaviour, they likely correspond to the same local behavioural phase, so folding across such regions may simplify the effective transition structure the dynamics model needs to capture. However, compressing too aggressively may remove useful distinctions between states and discard control information needed for later optimisation. We therefore aim to fold only within locally consistent regions whilst avoiding merges between behaviourally different timesteps.

To operationalise this intuition, we partition each trajectory segment into temporally contiguous groups and use the resulting grouping to restrict admissible folded future targets. We score candidate groupings to favour within-group behavioural consistency and clear separation between groups, whilst penalising partitions that unduly distort the agent's current policy structure.

Given such a grouping, we restrict admissible folded targets by defining a set of allowed future offsets for each start timestep. For example, consider a segment with group labels 0, 0, 1, 1, 1, 2, 3, 3, 3: from $t = 3$ (label 1), folding is allowed to $t = 4$ (same label) or $t = 5$ (the first timestep with the next label 2), but not to later labels (e.g., 3). Figure 3 illustrates the same restriction within this example.

Formally, let c_t denote the group label of timestep t, and let c_t^+ be the next distinct label that appears after t in temporal order (if any). The admissible offsets are:

- **Intra-group:** any $t + \kappa$ with $\kappa \geq 1$ such that $c_{t+\kappa} = c_t$;
- **Next-group:** any $t + \kappa$ with $\kappa \geq 1$ such that $c_{t+\kappa} = c_t^+$.

All other offsets receive zero probability under $q_{\text{fold}}(\cdot \mid \tau_{t:t+K})$.

We now describe how candidate groupings are scored, as this score determines which grouping is ultimately retained for the current segment. Intuitively, a useful grouping should satisfy two requirements. First, different groups should correspond to different action patterns, so that the grouping preserves behaviourally meaningful structure. Secondly, timesteps placed in the same group should have similar policy distributions; otherwise the grouping would blur the current local decision structure.

Formally, let m be a contiguous partition of the timesteps $\{0, \ldots, K-1\}$, and let c_t denote the group label assigned to timestep t. For each group c, define

$$\mathcal{G}(c) \triangleq \{u : c_u = c\},$$

the set of timesteps assigned to that group.

To quantify action-pattern separation, we first define the empirical action distribution within each group:

$$p_m(a \mid c) \quad \triangleq \quad \frac{1}{|\mathcal{G}(c)|} \sum_{u \in \mathcal{G}(c)} \mathbf{1}[a_u = a]. \tag{9}$$

We also define the empirical weight of group c in the segment by

$$p_m(c) \quad \triangleq \quad \frac{|\mathcal{G}(c)|}{K}, \tag{10}$$

which accounts for how often that group appears in the segment, and the resulting marginal action distribution

$$p_m(a) \quad \triangleq \quad \sum_c p_m(c)\, p_m(a \mid c). \tag{11}$$

This marginal is simply the empirical action distribution over the whole segment. We then measure how informative the grouping is about action behaviour using mutual information, where C and A denote the empirical group-label and action variables:

$$I_m(C; A) \quad \triangleq \quad \sum_c p_m(c) \sum_a p_m(a \mid c) \log \frac{p_m(a \mid c)}{p_m(a)}. \tag{12}$$

This quantity is large when different groups induce noticeably different action distributions, meaning that the grouping separates distinct local action patterns.

We next penalise policy distortion within each group. The idea is simple: if the timesteps assigned to the same group already have similar policy distributions, then representing them by a group-average policy causes little distortion. Conversely, if their policies differ substantially, grouping them together blurs the original per-timestep decision structure.

For each timestep t and action a, define the group-averaged policy

$$\tilde{\pi}_m(a \mid s_t) \quad \triangleq \quad \frac{1}{|\mathcal{G}(c_t)|} \sum_{u \in \mathcal{G}(c_t)} \pi(a \mid s_u), \tag{13}$$

and the total policy-distortion penalty

$$\mathrm{KL}_\pi(m) \quad \triangleq \quad \sum_{t=0}^{K-1} \sum_a \pi(a \mid s_t) \log \frac{\pi(a \mid s_t)}{\tilde{\pi}_m(a \mid s_t)}. \tag{14}$$

This penalty is small when policies are consistent within groups and large when grouping merges timesteps with substantially different policy distributions.

Combining the two terms, we score a candidate grouping by

$$\mathrm{score}(m) \quad \triangleq \quad I_m(C; A) - \beta\, \mathrm{KL}_\pi(m), \tag{15}$$

where $\beta \geq 0$ controls the trade-off between separating different action patterns and preserving the original per-timestep policy structure.

In implementation, each candidate grouping is represented by a short discrete mask vector over the segment timesteps, which is then converted into contiguous temporal groups. We initialise a population of random candidates, evaluate each grouping using the score in Eq. (15), and evolve the population with tournament selection, one-point crossover, and simple local mutations that modify short temporal regions of the mask. The highest-scoring grouping is then retained for the current segment. Because the score depends on the current policy, the resulting grouping can adapt as training progresses rather than imposing a fixed abstraction.

Once a grouping has been selected, the admissible offsets defined above determine the support of folded transitions. On this admissible set, we optionally apply a geometric distance decay,

$$q_{\text{fold}}(\kappa \mid \tau_{t:t+K}) \quad \propto \quad \exp\big(-\alpha_{\text{abs}}(\kappa - 1)\big). \tag{16}$$

This weighting is applied only to admissible offsets. Here $\alpha_{\text{abs}} \geq 0$ is the distance-decay rate, so larger values favour nearer admissible targets. We then sample κ from $q_{\text{fold}}(\cdot \mid \tau_{t:t+K})$ and apply (7) to obtain the folded transition stored in $\mathcal{D}_{\text{fold}}$.

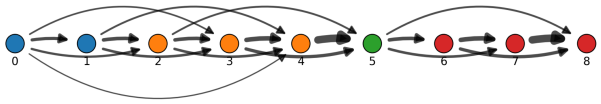


Figure 3: State-abstraction-guided folding: A trajectory segment is partitioned into abstract groups (colours) by a searched grouping m. Directed edges denote admissible folded endpoints restricted to the same group or the immediately subsequent group in temporal order, while edge thickness reflects a distance-biased sampling preference.

4.2.4 Segment folding with value-guided propagation. This second mechanism complements the abstraction-guided variant above. Rather than changing which future offsets are admissible, it keeps the admissible support $\kappa \in \{1, \ldots, K\}$ unchanged and instead reweights the *density* over candidate future states.

The intuition is to prefer future targets whose predicted values differ substantially from the current state, since such targets are more likely to induce informative updates, whilst still discouraging excessively long jumps with a distance decay.

Let $V(s)$ denote the current value estimate maintained by the agent; in our tabular setting we use $V(s) = \max_a Q(s, a)$ from the current Q-table. For a fixed start timestep t, define the value contrast of candidate offset κ by

$$\Delta V_{t,\kappa} \triangleq |V(s_{t+\kappa}) - V(s_t)|, \qquad \kappa \in \{1, \ldots, K\}. \tag{17}$$

We then assign each admissible future offset an unnormalised weight with two components: a contrast term that upweights future targets whose predicted values differ more from the current state, and a distance-decay term that downweights offsets that are farther away in time:

$$w_t(\kappa) \triangleq \left(\Delta V_{t,\kappa} + \varepsilon_v\right)^p \cdot \exp(-\alpha\,(\kappa - 1)), \tag{18}$$

where $\varepsilon_v > 0$ is a small bias to avoid zero weight when $\Delta V_{t,\kappa} = 0$, $p \geq 1$ controls how strongly large contrasts are amplified, and $\alpha \geq 0$ controls the strength of the distance decay. Thus, larger value contrast increases the weight, whereas longer jump distance decreases it; a distant target receives high probability only if its value contrast is large enough to offset the temporal penalty. The resulting folding distribution is obtained by normalising these weights over the candidate offsets for the same start timestep:

$$q_{\text{fold}}(\kappa \mid \tau_{t:t+K}) \triangleq \frac{w_t(\kappa)}{\sum_{j=1}^{K} w_t(j)}. \tag{19}$$

A folded sample is then emitted by (7) using the sampled κ.

For example, we use $s_{t:t+K} \triangleq (s_t, s_{t+1}, \ldots, s_{t+K})$, consider the value sequence along a segment (as visualised in Figure 4):

$$V(s_{t:t+6}) = [0.2,\ 0.25,\ 0.05,\ 0.12,\ 0.05,\ 0.3,\ 0.4]. \tag{20}$$

For the start state s_t with $V(s_t) = 0.2$, the largest contrast occurs at a far future state ($\kappa = 6$, contrast 0.2), while $\kappa = 2$ also has substantial contrast (0.15). When $\alpha > 0$, the decay term penalises long jumps, so the distribution in (19) can place its highest mass on a *near* but *high-contrast* future state (such as $\kappa = 2$), while still assigning non-trivial probability to more distant, high-contrast targets (such as $\kappa = 6$). Figure 4 visualises the same trade-off: thicker edges correspond to larger value contrast and shorter jump distance. Since the distribution in (19) is normalised separately for each start timestep, later positions have fewer candidate future states due to the finite segment length. This boundary effect does not materially change the intended reweighting behaviour.

4.3 Training framework

This section describes how the proposed transition-shaping method is applied in the training procedure used in our experiments. The folding operator $\mathcal{F}$ generates additional supervision data from sequential segments and is instantiated by the two mechanisms described in Sects. 4.2.3 and 4.2.4. Model updates draw from both real transitions and folded transitions, with a synthetic fraction $\rho(\ell) \in [0, 1]$ controlling how much folded data is mixed into each update at training iteration ℓ.

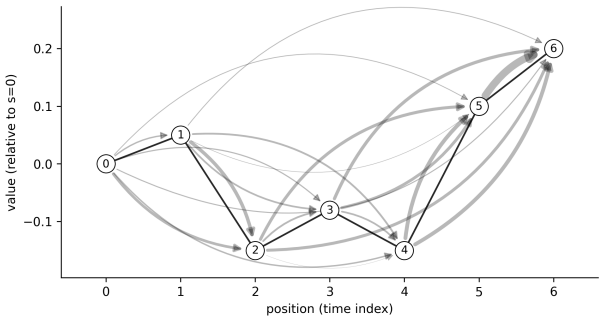


Figure 4: Value-guided segment folding. Nodes are positioned by time (horizontal axis) and predicted value (vertical axis). Edges correspond to candidate future offsets $\kappa \in \{1, \ldots, K\}$, and edge thickness reflects the sampling probability in (19), increasing with value contrast and decreasing with jump distance. This mechanism preserves the admissible support while reshaping the *density* over folded targets.

Algorithm 1 Online training with folded supervision

Require: env $\mathcal{E}$, agent π, model M, operator $\mathcal{F}$, schedule $\rho(\cdot)$

while interaction budget not reached **do**
  Collect real transitions with π; append to $\mathcal{D}_{\text{raw}}$.
  Generate folded samples from $\mathcal{D}_{\text{raw}}$ via $\mathcal{F}$; append to $\mathcal{D}_{\text{fold}}$.
  Update synthetic fraction $\rho(\ell)$ according to schedule.
  Update dynamics model M using mixed batches from $\mathcal{D}_{\text{raw}}$ and $\mathcal{D}_{\text{fold}}$ with fraction $\rho(\ell)$.
  Improve π by rollouts from M.
  Periodically evaluate π on $\mathcal{E}$.
end while
return π

We follow the standard model-based RL loop, where the agent collects real transitions from the environment to train a dynamics model, and the learned model is used to support policy improvement. Real transitions are stored in a replay buffer $\mathcal{D}_{\text{raw}}$, and the folding operator $\mathcal{F}$ samples segments from $\mathcal{D}_{\text{raw}}$ to generate additional folded transitions, which are stored in a separate buffer $\mathcal{D}_{\text{fold}}$. The dynamics model is then trained on batches drawn from both buffers according to the current synthetic fraction $\rho(\ell)$. As training proceeds, newly collected real transitions continue to expand $\mathcal{D}_{\text{raw}}$ and provide data for further folding. The resulting procedure is summarised in Algorithm 1.

5 EXPERIMENTS

This section evaluates whether shaping the transition distribution for dynamics-model training improves downstream policy learning. The experiments follow the framework in Sec. 4.3: the folding operator $\mathcal{F}$ constructs additional transition samples from replayed trajectory segments using the mechanisms in Secs. 4.2.3 and 4.2.4. The dynamics model is trained on a mixture of real and folded

transitions, with the mixing fraction $\rho(\ell)$ controlling how much folded data is used in each update (Sec. 4.3), and the updated model is used throughout policy learning.

We address the following questions:

(1) **Effectiveness:** Does transition-distribution shaping speed up policy learning while maintaining stable training?
(2) **Performance of the sampling methods:** Do different future-state sampling designs (state-abstraction vs. value-guided) lead to different performance outcomes?
(3) **Sensitivity to mixing strength:** How sensitive is performance to the mixing fraction $\rho(\ell)$ between real and constructed transitions? What schedules work well in practice?
(4) **Shaping methods vs. "just adding jumps":** Are performance gains explained by the proposed sampling distributions, or can similar improvements be obtained by adding long-range constructed transitions without these designs?

To answer these questions, we report (i) results across three domains, (ii) mixing-fraction ablations over fixed and scheduled $\rho(\ell)$, and (iii) ablations comparing the proposed two folding mechanisms against a random-jump baseline, i.e., an unguided segment-folding baseline (baseline-jump). Implementation details and domain choices are described next, followed by the main results and ablations.

5.1 Experimental implementation

All experiments use tabular dynamics models and policies, which provide stable training behaviour and keep the implementation transparent. For each state–action pair (s, a), the dynamics model stores the set of observed successor states and an explicit transition probability $p_\theta(s' \mid s, a)$, together with reward and termination statistics. Rather than estimating transition probabilities from cumulative counts over the full replay history, we update the tabular model by batched exponential moving average (EMA). Let B_ℓ denote the mini-batch sampled at update ℓ, and let $\hat{p}_\ell(y \mid s, a)$ be the empirical outcome distribution in that batch for outcome tuple $y = (s', \bar{r}, \text{done})$:

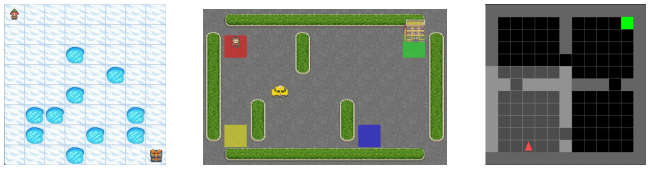
$$\hat{p}_\ell(y \mid s, a) \quad \triangleq \quad \frac{N_\ell(s, a, y)}{\sum_{\tilde{y}} N_\ell(s, a, \tilde{y})}, \tag{21}$$

where $N_\ell(s, a, y)$ counts how often (s, a, y) appears in B_ℓ. The model is then updated in the simplified EMA form

$$p_{\ell+1}(y \mid s, a) \quad \triangleq \quad (1 - \lambda_M)\, p_\ell(y \mid s, a) \; + \; \lambda_M\, \hat{p}_\ell(y \mid s, a), \tag{22}$$

with step size $\lambda_M \in (0, 1]$. New outcomes are added as new edges, after which the transition probabilities for that state–action pair are renormalised. Low-probability outcomes are pruned during separate maintenance steps using a threshold τ_p.

The policy is represented by a Q-table updated from replay using the same EMA-style rule. Action selection during training uses a stochastic mixture of greedy, softmax, and uniform exploration, yielding an explicit policy distribution. This soft policy distribution is also used when computing the KL-based quantities in Sec. 4.2.3, ensuring consistency between the policy used for control and the distributions used by the folding operators.



(a) FrozenLake **(b) Taxi** **(c) Four Rooms**

Figure 5: Environments used in experiments.

5.1.1 Environments. We evaluate the proposed approach on three discrete environments: FrozenLake, Taxi, and Four Rooms (MiniGrid), shown in Fig. 5. These tasks differ in reward sparsity, transition stochasticity, and state-space size while remaining compatible with tabular dynamics models. FrozenLake has sparse rewards and stochastic transitions, Taxi provides structured rewards with moderate stochasticity, and Four Rooms has a larger state space with deterministic transitions. In FrozenLake, the agent uses a small positive initial Q-value (0.001) to encourage exploration; other environments use zero for this initialisation.

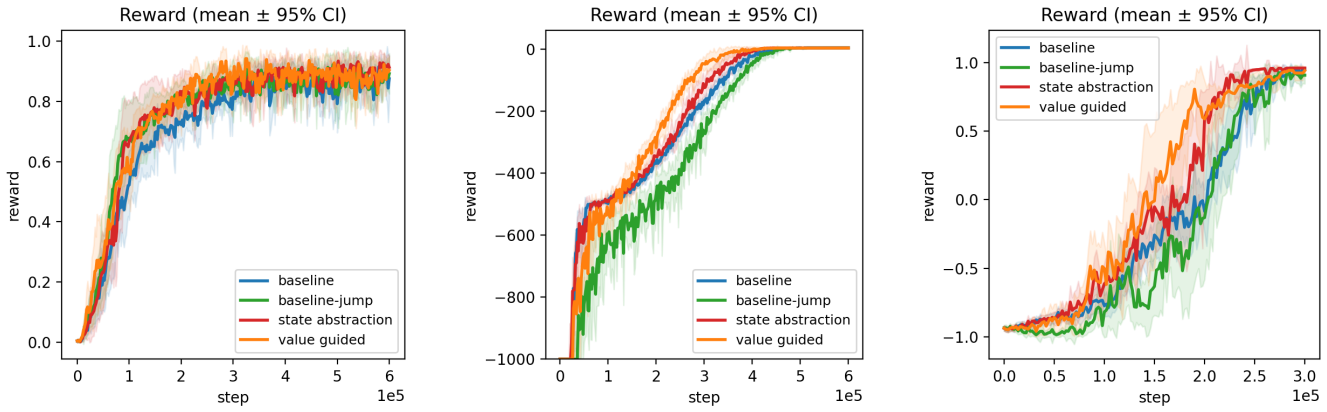
5.1.2 Evaluation protocol and metrics. For each experiment task, all methods use identical interaction budgets, optimisation schedules, refitting frequencies, and evaluation intervals. Each configuration is trained over 20 independent base random seeds and evaluated on 100 fixed test environment seeds (and the test environments are reset with the seeds every time) to ensure both diversity and comparability (in Four Rooms, the agent will be randomly initialised inside the bottom-left room with arbitrary direction). For comparison, a baseline uses the same training procedure throughout but trains the dynamics model exclusively from the raw replay buffer without any folded transitions.

5.1.3 Hyperparameters. All folding-related hyperparameters are fixed across all environments and experiments. We use a segment length of $K = 10$. For abstraction-guided folding, we use a distance-decay rate $\alpha_{\text{abs}} = -\log 0.9 \approx 0.105$, bottleneck strength $\beta = 0.5$, and folding discount $\bar{\gamma} = 0.8$. For value-guided folding, we use distance decay $\alpha = 0.5$, contrast exponent $p = 1.5$, and folding discount $\bar{\gamma} = 0.8$. These parameters are chosen empirically and remain unchanged across domains. We ablate the mixing fraction $\rho(\ell)$ between real and folded transitions, as it directly controls the strength of transition-distribution shaping. In our experiments, $\rho(\ell)$ is linearly reduced from 0.5 to 0.25.

5.2 Results and Discussion

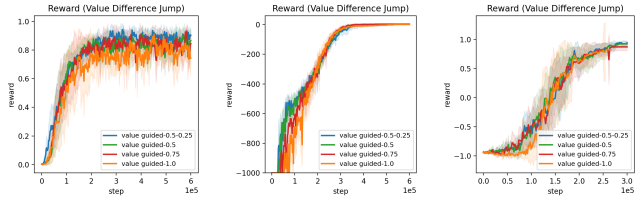
Across FrozenLake, Taxi, and Four Rooms (Fig. 6), both shaping variants outperform the real-only baseline. Value-guided shaping yields the strongest and most consistent gains in Taxi, while state-abstraction can match or slightly exceed it in the later stage of training (notably in Four Rooms); in FrozenLake the two variants reach similar performance, both slightly outperform the baseline.

5.2.1 Ablations and diagnostics. The difference in mixing strength slightly affects performance (Fig. 7). Value-guided shaping degrades when constructed transitions dominate (ρ=1.0) and performs best with a moderate schedule ($\rho : 0.5 \rightarrow 0.25$), whereas state-abstraction



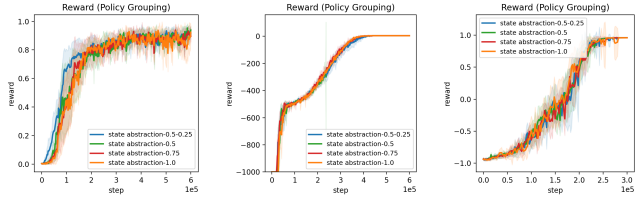
(a) FrozenLake (b) Taxi (c) Four Rooms

Figure 6: Reward curves of experiments for FrozenLake, Taxi and Four Rooms environments, across 20 seeds, x-axis is real environment rollout steps, y-axis is the average evaluation reward, shadings are 95% confidence intervals.



FrozenLake Taxi Four Rooms

(a) Value-guided shaping under different mixing fractions $\rho(\ell)$.



FrozenLake Taxi Four Rooms

(b) State-abstraction shaping under different mixing fractions $\rho(\ell)$.

Figure 7: Effect of the mixing fraction $\rho(\ell)$ between real and constructed transitions. Columns correspond to FrozenLake, Taxi, and Four Rooms. Across 20 seeds, x-axis is real environment rollout steps, y-axis is the average evaluation reward, shadings are 95% confidence intervals.

is less sensitive to ρ. A random-jump baseline helps only in FrozenLake and otherwise does not match the proposed variants. This suggests that the gains cannot be explained simply by adding long-range constructed transitions: unguided long-range jumps do not reproduce the same effect consistently across domains, indicating that the future-target selection strategy itself is important.

5.2.2 Summary. Across domains, both target-selection mechanisms improve policy learning over real-only training. Ablations show that structured target selection and moderate data mixing are important: excessive constructed data can hurt performance, and unstructured long-range jumps are not reliably beneficial.

6 CONCLUSIONS AND FUTURE WORK

We introduced folded supervision to augment one-step dynamics-model training with later reachable states, without changing the model's input/output interface. Across tabular experiments, this accelerates policy learning; ablation studies show that the benefit depends on the folding mechanism used and appropriate data mixing, rather than simply adding random jumps to future states.

One natural next step would be to extend folded supervision to function approximation. Such settings may in fact be more suitable for this approach, since stronger generalisation across states and tasks might allow the transition-shaping mechanism to be used more effectively than in the present tabular setting.

A natural question is why value-guided folding is effective. The present results suggest that the observed benefit involves more than simply introducing unguided long-range jumps, since the random-jump baseline does not show the same effect consistently across domains. A more focused set of control experiments would help clarify the relative roles of stronger induced learning signals and the structured selection of future targets.

It is also of interest to study whether support restriction and density reweighting can be combined into a unified folding rule, and to compare or integrate the approach with model–policy coupling and multi-step prediction methods [3, 6, 15, 19].

Finally, sequential multi-task, curriculum learning, and transfer learning settings may provide a broader testbed for the idea, especially in conjunction with shared world models under function approximation [4, 7], where stronger generalisation across related states and tasks may be possible. In contrast, in the present tabular setting, transition, reward, and termination statistics are stored in a state-specific manner, which may limit transfer even when the folding mechanism itself is useful.

ACKNOWLEDGMENTS

The authors thank Matthew Barker of King's College London for his generous help and support during the writing of this paper. The authors acknowledge the use of the King's Computational Research, Engineering and Technology Environment (CREATE) at King's College London in support of this research [12].

REFERENCES

[1] David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L Littman, and Lawson LS Wong. 2019. State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3134–3142.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[3] Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. 2024. Simple Hierarchical Planning with Diffusion. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*. Poster.

[4] Jie Cheng, Ruixi Qiao, Yingwei Ma, Binhua Li, Gang Xiong, Qinghai Miao, Yongbin Li, and Yisheng Lv. 2025. Scaling Offline Model-Based RL via Jointly-Optimized World-Action Model Pretraining. In *International Conference on Learning Representations*, Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (Eds.), Vol. 2025. 42396–42418. https://proceedings.iclr.cc/paper_files/paper/2025/file/689cffc97600f9deb8374fc8fa918b8e-Paper-Conference.pdf

[5] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2019. Diversity is All You Need: Learning Skills without a Reward Function. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. Poster.

[6] Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Russ R Salakhutdinov. 2022. Mismatched no more: Joint model-policy optimization for model-based RL. *Advances in Neural Information Processing Systems* 35 (2022), 23230–23243.

[7] Ignat Georgiev, Varun Giridhar, Nicklas Hansen, and Animesh Garg. 2025. PWM: Policy Learning with Multi-Task World Models. In *Proceedings of the 13th International Conference on Learning Representations (ICLR 2025)*. Poster.

[8] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2020. Dream to Control: Learning Behaviors by Latent Imagination. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*. Oral.

[9] Nicklas Hansen, Xiaolong Wang, and Hao Su. 2022. Temporal Difference Learning for Model Predictive Control. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022) (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, Baltimore, Maryland, USA.

[10] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. 2019. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems* 32 (2019).

[11] Chengxing Jia, Fuxiang Zhang, Tian Xu, Jing-Cheng Pang, Zongzhang Zhang, and Yang Yu. 2024. Model gradient: unified model and policy learning in model-based reinforcement learning. *Frontiers of Computer Science* 18, 4 (2024), 184339.

[12] King's College London. 2022. King's Computational Research, Engineering and Technology Environment (CREATE). https://doi.org/10.18742/rnvf-m076 Retrieved April 7, 2026.

[13] Qiyang Li, Zhiyuan Zhou, and Sergey Levine. 2025. Reinforcement Learning with Action Chunking. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*. Poster.

[14] Yu-An Lin, Chen-Tao Lee, Chih-Han Yang, Guan-Ting Liu, and Shao-Hua Sun. 2024. Hierarchical programmatic option framework. *Advances in Neural Information Processing Systems* 37 (2024), 126677–126724.

[15] Alexander Neitz, Giambattista Parascandolo, Stefan Bauer, and Bernhard Schölkopf. 2018. Adaptive skip intervals: Temporal abstraction for recurrent dynamical models. *Advances in Neural Information Processing Systems* 31 (2018).

[16] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. 2020. Dynamics-Aware Unsupervised Discovery of Skills. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*. Talk.

[17] Richard S Sutton. 1991. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin* 2, 4 (1991), 160–163.

[18] Anirudh Vemula, Yuda Song, Aarti Singh, J. Andrew Bagnell, and Sanjiban Choudhury. 2023. The Virtues of Laziness in Model-based RL: A Unified Objective and Algorithms. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023) (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, Honolulu, Hawaii, USA.

[19] Xiyao Wang, Wichayaporn Wongkamjan, Ruonan Jia, and Furong Huang. 2023. Live in the moment: Learning dynamics model adapted to evolving policy. In *International Conference on Machine Learning*. PMLR, 36470–36493.

[20] Xiyao Wang, Ruijie Zheng, Yanchao Sun, Ruonan Jia, Wichayaporn Wongkamjan, Huazhe Xu, and Furong Huang. 2024. COPlanner: Plan to Roll Out Conservatively but to Explore Optimistically for Model-Based RL. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*. Poster.