CLIC-CHUNKS: CONTRASTIVE POLICY LEARNING FROM INTERACTIVE CORRECTIONS WITH ACTION CHUNKING

Mukil Saravanan **Zhaoting Li** **Jens Kober**
Delft University of Technology
{mukilsaravanan,z.li-23,j.kober}@tudelft.nl

ABSTRACT

Interactive Imitation Learning (IIL) methods often struggle with noisy human feedback and compounding errors over extended execution horizons. To address this, we introduce CLIC-Chunks, a framework that extends the geometric policy shaping of Contrastive Learning from Interactive Corrections (CLIC) Li et al. [2025a] into a receding horizon control setting. By parameterizing the policy as an Energy-Based Model (EBM) optimized via Langevin dynamics, CLIC-Chunks predicts temporally extended action sequences (chunks) whose probability mass is concentrated within robust optimal regions rather than exact point targets. This formulation prevents the policy from overfitting to flawed human corrections while ensuring smooth, temporally coherent trajectory executions. We evaluate CLIC-Chunks across five continuous control tasks, including contact-rich and multi-arm manipulation. Empirical results demonstrate that our approach is able to learn shorter action chunks while maintaining high success rates. However, our analysis also identifies a critical scalability bottleneck, revealing that Langevin MCMC sampling efficacy degrades when the total action chunk dimensionality exceeds 20.

Our implementation is publicly available at `https://github.com/MukilSaravanan/CLIC-Chunks`.

Keywords Interactive Imitation Learning · Contrastive Learning · Learning from Demonstration · Action Chunking · Energy-based Models

1 Preliminaries

1.1 Interactive Imitation Learning Formulation

We consider a typical Interactive Imitation Learning problem where a human teacher aims to improve the behavior of the learning agent, referred to as the learner, by providing feedback on the learner's actions Celemin et al. [2022].

We formulate this environment as a continuous Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{H})$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the continuous action space, $\mathcal{T}(s'|s, a)$ represents the environment's transition dynamics, and $\mathcal{H}$ defines the teacher's feedback mechanism.

During the learning process, the robot executes an action $a^r \sim \pi_\theta(\cdot|s)$ given its current state $s \in \mathcal{S}$. The human teacher acts as an evaluator who observes the robot's performance. If the executed action a^r is deemed suboptimal, the teacher intervenes by providing a corrective feedback signal $a^h \in \mathcal{A}$. This continuous interaction yields a dataset of state-conditioned action pairs $\mathcal{D} = \{(s_i, a_i^r, a_i^h)\}$.

The fundamental objective of the learning agent is to find an optimal policy π^* that minimizes a surrogate loss function $\ell_\pi(s)$:

$$\pi^* = \arg\min_\pi \mathbb{E}_{s\sim d_\pi(s)}[\ell_\pi(s)] \tag{1}$$

where $d_\pi(s)$ is the state distribution induced by the robot's current policy, and $\ell_\pi(s)$ measures the divergence between the learner's predicted actions and the teacher's corrections. By iteratively querying feedback on the exact states

visited by the active policy, the IIL framework naturally mitigates the compounding errors (covariate shift) commonly encountered in traditional offline behavior cloning.

1.2 CLIC Overview

To optimize the policy within this IIL framework, traditional methods typically apply a behavior cloning loss that directly imitates the human's corrective action a^h. However, this strict formulation often leads to overfitting, particularly when the human feedback is sub-optimal.

To mitigate this overfitting, CLIC relaxes the assumption that the human correction is perfectly optimal. Instead of treating a^h as an exact point target, CLIC utilizes the observed action pair (a^r, a^h) to construct a single-step *desired action space*, denoted as $\hat{\mathcal{A}}_{ss}(a^r, a^h)$. Geometrically, this space defines a continuous boundary that encapsulates the human correction a^h and the presumed optimal action a^*, while strictly isolating and excluding the robot's suboptimal execution a^r. For intervention data, this space is typically defined as a hypersphere centered at the human action a^h:

$$\hat{\mathcal{A}}_{ss}(a^r, a^h) = \left\{ a \in \mathcal{A} \mid r \cdot \mathbb{D}(a^h, a^r) \geq \mathbb{D}(a, a^h) \right\} \tag{2}$$

where $r \in (0, 1]$ is a hyperparameter controlling the radius (and thus the volume) of the desired space, and $\mathbb{D}(a_1, a_2) = \|a_1 - a_2\|$ is the distance metric. By defining the target as a continuous geometric region rather than a rigid point estimate, the algorithm achieves inherent robustness to the noise and suboptimality frequently observed in human demonstrations.

During training, CLIC shapes the robot's policy $\pi_\theta(a|s)$ parameterized as an EBM to concentrate its probability within these desired regions. This is achieved by defining a probabilistic observation model over the desired action space and minimizing the Kullback-Leibler (KL) divergence between the current policy and a target distribution that favors actions inside $\hat{\mathcal{A}}_{ss}(a^r, a^h)$:

$$\ell_{\text{KL}}(\theta) = \mathbb{E}_{(s,a^r,a^h)\sim\mathcal{D}} \left[\text{KL} \left(\pi^{\text{target}}(\cdot|s) \| \pi_\theta(\cdot|s) \right) \right] \tag{3}$$

As the policy iteratively receives and aggregates multiple corrections for similar states, the intersection of these individual desired action spaces progressively refines the overall optimal region. This mechanism ensures the policy converges toward the true optimal actions without overfitting to any isolated, potentially flawed human demonstration.

1.3 Action Chunking

Action chunking is a technique popularized in imitation learning to combat compounding errors and improve policy robustness Zhao et al. [2023]. Inspired by the neuroscience concept of grouping individual movements into cohesive motor units, action chunking shifts the learning paradigm from single-step decisions to temporally extended sequences. Formally, instead of learning a single-step policy $\pi_\theta(a_t|s_t)$, the policy is trained to predict a sequence of H future actions, $\pi_\theta(\mathbf{a}_{t:t+H-1}|s_t)$, conditioned solely on the current state observation s_t.

By predicting multiple steps simultaneously, action chunking effectively reduces the decision-making horizon of the task. This is particularly advantageous for maintaining temporal coherence, ensuring smooth robot motions, and modeling non-Markovian behaviors in human demonstration data (such as temporary pauses or temporally correlated confounders). During inference, the agent can execute the predicted chunk in an open-loop manner for a fixed number of steps T_a (where $T_a \leq H$) before querying the policy again with a new observation. This receding horizon execution plays a crucial balance between the stability of long-term planning and the reactivity of closed-loop control.

2 Extending Desired Action Space to Chunk Space

To reason over extended temporal horizons, we model the robot's policy as an EBM. Parameterized by a Multi-Layer Perceptron (MLP) θ, the EBM evaluates the compatibility of an H-step predicted action sequence $\mathbf{a}_{i:i+H-1}$ (flattened to dimension $D \times H$) conditioned on the state $\mathbf{s}_i$. The policy follows a Boltzmann distribution:

$$\pi_\theta(\mathbf{a}_{i:i+H-1}|\mathbf{s}_i) = \frac{\exp(-E_\theta(\mathbf{s}_i, \mathbf{a}_{i:i+H-1}))}{Z} \tag{4}$$

where $Z = \int \exp(-E_\theta(\mathbf{s}_i, \mathbf{a}))d\mathbf{a}$ is the intractable partition function.

When a human provides corrective feedback $\mathbf{a}^h_{i:i+H-1}$ for a suboptimal robot execution $\mathbf{a}^r_{i:i+H-1}$, we assume the true optimal sequence lies near the correction. Using the sequence-level L_2 norm distance $\mathbb{D}$, we construct a continuous *Circular Desired Action Chunk Space* $\hat{\mathcal{A}}^C$ defined as a hypersphere centered at the human's correction:

$$\hat{\mathcal{A}}^C(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) = \left\{ \mathbf{a}_{i:i+H-1} \in \mathcal{A}^H \mid r(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) \geq \mathbb{D}(\mathbf{a}_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) \right\} \tag{5}$$

Here, the radius $r(\mathbf{a}^r, \mathbf{a}^h) = \gamma \cdot \mathbb{D}(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1})$ is scaled by a hyperparameter $\gamma \in (0, 1]$. This explicitly encapsulates a robust region of optimal sequence behaviors while excluding the suboptimal robot chunk.

To prevent instability from a hard decision boundary during training, we apply a probabilistic observation model. This softens the boundary using a sigmoid function σ_T with a temperature parameter T:

$$p(\mathbf{a}_{i:i+H-1} \in \hat{\mathcal{A}}_i \mid \mathbf{a}_{i:i+H-1}, \mathbf{s}_i) = \sigma_T \left(r(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) - \mathbb{D}(\mathbf{a}_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) \right) \tag{6}$$

2.1 Energy-Based Model Sampler via Langevin Dynamics

To learn robust decision boundaries, the EBM requires negative action chunk samples $\mathcal{A}_i$ drawn directly from its current probability landscape. We generate these samples using Stochastic Gradient Langevin Dynamics (SGLD). The update rule at Markov Chain Monte Carlo (MCMC) step $k \in \{1, \ldots, N_{\text{MCMC}}\}$ is given by:

$$\tilde{\mathbf{a}}^{(k)}_{i:i+H-1} = \tilde{\mathbf{a}}^{(k-1)}_{i:i+H-1} - \lambda \nabla_{\mathbf{a}} E_\theta(\mathbf{s}_i, \tilde{\mathbf{a}}^{(k-1)}_{i:i+H-1}) + \sqrt{2\lambda}\omega^{(k)} \tag{7}$$

where the initial sequence $\tilde{\mathbf{a}}^{(0)}_{i:i+H-1} \sim \mathcal{U}(-1, 1)^H$, the step size is $\lambda > 0$, and standard Gaussian noise $\omega^{(k)} \sim \mathcal{N}(0, \mathbf{I})$ matches the dimension of the flattened chunk.

During inference, the policy executes the single optimal sequence that minimizes the energy:

$$\hat{\mathbf{a}}^*_{t:t+H-1} = \arg\min_{\mathbf{a}} E_\theta(\mathbf{s}_t, \mathbf{a}) \tag{8}$$

To approximate this global minimum within the high-dimensional chunk space, we apply a two-stage MCMC refinement. An initial Langevin exploration chain is followed by a fine-tuning stage where λ decays via a polynomially annealed schedule. This annealed noise schedule facilitates the escape from shallow local minima during early iterations, promoting stable convergence to the deepest available energy basin.

3 Policy Shaping via Desired Action Chunk Spaces

The objective of CLIC-Chunks is to concentrate the probability mass of E_θ within the constructed desired action spaces. Because the partition function Z in Eq. 4 is intractable over the high-dimensional chunk space, we approximate the distribution using MCMC. Specifically, we draw N_a counter-example action sequences via Langevin dynamics to form a discrete evaluation set $\mathcal{A}_i$:

$$\mathcal{A}_i = \{\mathbf{a}^h_{i:i+H-1}, \mathbf{a}^r_{i:i+H-1}\} \cup \left\{ \mathbf{a}^{(j)}_{i:i+H-1} \mid j = 1, \ldots, N_a \right\} \tag{9}$$

Using this set, we estimate a target policy distribution via policy-weighted Bayes. We re-weight the current policy's probabilities based on the observation model to upweight actions inside the desired space without deviating excessively from the current policy:

$$\pi^{\text{target}}(\mathbf{a}_{i:i+H-1}|\mathbf{s}_i) = \frac{p_{\hat{\mathcal{A}}_i}(\mathbf{a}_{i:i+H-1}|\mathbf{s}_i) \exp(-E_\theta(\mathbf{s}_i, \mathbf{a}_{i:i+H-1}))}{\sum_{\mathbf{a}'_{i:i+H-1} \in \mathcal{A}_i} p_{\hat{\mathcal{A}}_i}(\mathbf{a}'_{i:i+H-1}|\mathbf{s}_i) \exp(-E_\theta(\mathbf{s}_i, \mathbf{a}'_{i:i+H-1}))} \tag{10}$$

where $p_{\hat{\mathcal{A}}_i}(\ldots)$ denotes the observation model probability $p(\mathbf{a}_{i:i+H-1} \in \hat{\mathcal{A}}_i \mid \mathbf{a}_{i:i+H-1}, \mathbf{s}_i)$.

Finally, the policy is optimized by minimizing the Kullback-Leibler (KL) divergence between this target distribution and the current policy across a sampled batch $\mathcal{B}$. To maintain a smooth energy landscape and prevent overfitting to isolated human demonstrations, we augment the loss with an infinity-norm gradient penalty (with margin m):

$$\ell(\theta) = \mathbb{E}_{\mathcal{B}} \left[\text{KL} \left(\pi^{\text{target}}(\cdot|\mathbf{s}_i) \| \pi_\theta(\cdot|\mathbf{s}_i) \right) \right] + \lambda \mathbb{E}_{\mathcal{A}_i} \left[\max(0, \|\nabla_{\mathbf{a}} E_\theta\|_\infty - m)^2 \right] \tag{11}$$

4 Algorithm

CLIC-Chunks extends the interactive refinement of desired action spaces into a receding horizon control framework. At each decision step, the policy predicts a full trajectory chunk of length H, but only executes a shorter sub-sequence of T_a steps. When the executed behavior is suboptimal, the teacher provides corrective feedback specifically for those T_a steps. This partial feedback is then padded to match the full prediction horizon and aggregated in a data buffer. By periodically sampling from this buffer, the method constructs desired action chunk spaces and computes a surrogate loss to shape the energy-based policy. This continuous evaluation and update process concentrates the policy's probability mass within the bounds of optimal trajectory sequences, maintaining temporal consistency while preventing the model from overfitting to isolated sequence labels. The detailed descriptions are mentioned in Algorithm 4

Algorithm 1 CLIC - Chunks: Contrastive policy Learning from Interactive Corrections (with Action Chunking)

Notations
H : Prediction horizon (predicted action chunk size)
T_a : Execution horizon (number of steps executed per chunk, $T_a \leq H$)
$\mathcal{D}$: Data buffer of observed action chunk pairs $(\mathbf{s}_i, \mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1})$
$\mathcal{B}$: Batch sampled from data buffer $\mathcal{D}$
π_θ : Policy parameterized by θ, via an energy model E_θ
$\mathcal{A}_i$: Set of action chunk samples for state $\mathbf{s}_i$
$\hat{\mathcal{A}}_i$: Shorthand for $\hat{\mathcal{A}}(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1})$
$p_{\hat{\mathcal{A}}_i}$: Shorthand for observation model $p(\mathbf{a}_{i:i+H-1} \in \hat{\mathcal{A}}(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) | \mathbf{a}_{i:i+H-1}, \mathbf{s}_i)$
b : In-episode update frequency
N_{training} : End-of-episode training steps
η : Learning rate
▷ Interactive Imitation Learning Loop with Receding Horizon
for $episode = 1, 2, \ldots$ **do**
  $t \leftarrow 1$
  while $t \leq$ max_steps **do**
    Observe $\mathbf{s}_t$, predict action chunk $\mathbf{a}^r_{t:t+H-1} \sim \pi_\theta(\cdot|\mathbf{s}_t)$
    Execute first T_a steps of the chunk: $\mathbf{a}^r_{t:t+T_a-1}$
    Receive teacher feedback chunk $\mathbf{a}^h_{t:t+T_a-1}$, if robot execution is suboptimal
    Append $(\mathbf{s}_t, \mathbf{a}^r_{t:t+T_a-1}, \mathbf{a}^h_{t:t+T_a-1})$ (padded to H) to $\mathcal{D}$, if feedback is provided
    if $\mathbf{a}^h_{t:t+T_a-1}$ **or** $t \% b = 0$ **then**
      Sample batch $\mathcal{B}$ from $\mathcal{D}$
      $\{\hat{\mathcal{A}}_i, p_{\hat{\mathcal{A}},i}\}_{i \in \mathcal{B}} \leftarrow$ DESIREDACTIONCHUNKSPACE($\mathcal{B}$)
      $\theta \leftarrow$ POLICYSHAPING($\mathcal{B}, \{\hat{\mathcal{A}}_i\}, \{p_{\hat{\mathcal{A}},i}\}, \theta$)
    end if
    $t \leftarrow t + T_a$ ▷ Step forward by execution horizon
  end while
  Update policy π_θ using POLICYSHAPING for N_{training} steps
end for
▷ Constructing Desired Action Chunk Spaces
function DESIREDACTIONCHUNKSPACE($\mathcal{B}$)
  for each $(\mathbf{s}_i, \mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) \in \mathcal{B}$ **in parallel do**
    Create Circular $\hat{\mathcal{A}}(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1})$ via Eq. 5
    Define $p(\mathbf{a}_{i:i+H-1} \in \hat{\mathcal{A}}(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) | \mathbf{a}_{i:i+H-1}, \mathbf{s}_i)$ via Eq. 6
  end for
  return $\{\hat{\mathcal{A}}_i\}, \{p_{\hat{\mathcal{A}},i}\}$
end function
▷ Policy shaping via Desired Action Chunk Spaces
function POLICYSHAPING($\mathcal{B}, \{\hat{\mathcal{A}}_i\}, \{p_{\hat{\mathcal{A}},i}\}, \theta$)
  for each $(\mathbf{s}_i, \mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) \in \mathcal{B}$ **in parallel do**
    Draw action chunk samples $\mathcal{A}_i$ from $E_\theta(\mathbf{s}_i, \cdot)$, via Eq. 9
    Estimate $\pi_\theta(\mathbf{a}_{i:i+H-1}|\mathbf{s}_i) \propto \exp(-E_\theta(\mathbf{s}_i, \mathbf{a}_{i:i+H-1}))$, for $\mathbf{a}_{i:i+H-1} \in \mathcal{A}_i$, via Eq. 4
    Calculate $p(\mathbf{a}_{i:i+H-1} \in \hat{\mathcal{A}}(\mathbf{a}^r_{i:i+H-1}, \mathbf{a}^h_{i:i+H-1}) | \mathbf{a}_{i:i+H-1}, \mathbf{s}_i)$, for $\mathbf{a}_{i:i+H-1} \in \mathcal{A}_i$
    Estimate target $\pi^{\text{target}}(\mathbf{a}_{i:i+H-1}|\mathbf{s}_i)$, for $\mathbf{a}_{i:i+H-1} \in \mathcal{A}_i$, via Eq. 10
    Accumulate $\ell(\theta)$ += $\mathbb{E}_{\mathcal{B}}[\text{KL}(\pi^{\text{target}}(\cdot|\mathbf{s}_i) \| \pi_\theta(\cdot|\mathbf{s}_i))]$ via Eq. 11
  end for
  $\theta \leftarrow \theta - \eta \nabla_\theta \ell(\theta)$
  return θ
end function

5 Experiments

The primary objective of our experimental evaluation is to demonstrate the efficacy, robustness, and scalability of CLIC-Chunks in continuous control tasks. Specifically, we aim to answer the following key questions:

1. Does extending desired action spaces to temporally extended chunks improve overall policy performance (see evaluation metrics in Section 5.1) under optimal feedback?
2. How do the spatial hyperparameters (such as the radius ratio r) influence the final success rate of the algorithm?

5.1 Tasks and Evaluation Metrics

We evaluate our approach on six simulated continuous control tasks ranging from low-dimensional navigation to high-dimensional, contact-rich manipulation: Line Following, Push-T, MetaWorld-Hammer, Square, Pick-Can, and TwoArm-Lift (details in Appendix 7.1). The learned policies are assessed using four evaluation metrics:

1. Task Success Rate ($\mathcal{S}$) The percentage of randomized evaluation episodes where the agent achieves the environment's success condition within the allotted timesteps.

2. Normalized Action L2 Error ($\bar{\mathcal{E}}_{L2}$) To account for varying episode lengths, we compute the average Euclidean distance between the executed action $\mathbf{a}_t^r$ and the optimal oracle action $\mathbf{a}_t^*$ per timestep over an episode of length T:

$$\bar{\mathcal{E}}_{L2} = \frac{1}{T}\sum_{t=1}^{T} \|\mathbf{a}_t^* - \mathbf{a}_t^r\|_2 \tag{12}$$

3. Normalized Feedback Effort ($\bar{\mathcal{F}}$) To account for varying episode lengths, we quantify the human cognitive burden as the intervention rate: the fraction of timesteps requiring a corrective teacher intervention $f_t \in \{0, 1\}$ over the total episode length T:

$$\bar{\mathcal{F}} = \frac{1}{T}\sum_{t=1}^{T} f_t \tag{13}$$

5.2 Experiments with Accurate Feedback

We evaluate CLIC-Chunks under *accurate feedback*, where corrective signals are precise and noise-free. Across all tasks, this leads to high success rates, particularly in simpler settings like Line Following, which achieves near-perfect performance across horizons. For more complex manipulation tasks (PickCan, Push-T, MetaWorld Hammer), accurate feedback improves learning stability at small prediction horizons. However, L2 error grows with larger horizons, especially in high-dimensional action spaces, for example, TwoArmLift (action dim 14) fails at horizon $H = 2$. The detailed performance is tabulated in Table 1 and their corresponding training graphs are depicted in Appendix 7.3.

5.2.1 Impact of Action Chunk Dimension on Performance

From Table 2, we analyze the impact of total action size (chunk $\times$ action dimension) on task performance. Across environments, performance saturates around a total action size of 20, highlighting two issues: the energy landscape becomes complex, and the Langevin MCMC sampler struggles to sample high-dimensional action chunks. These observations indicate a practical upper bound for chunk-based prediction, with negligible improvement observed even when increasing MCMC iterations $\{25, 50, 75, 100\}$, consistent with known limitations of EBM in the literature.

5.2.2 Effects of Hyperparameters

We analyze the sensitivity of CLIC-Chunks to its algorithmic parameters. Specifically, we study the effect of the prediction horizon (chunk size) H and the radius ratio r, which determines the volume of the Circular Desired Action Chunk Space, balancing strict imitation with robust exploration. Table 3 reports task performance for different r values, averaged over H. Results indicate that low radius ratios (e.g., $r = 0.1$) achieve the best overall performance across tasks. Larger values of r can degrade performance by either limiting exploration around human actions in the action chunk space.

Table 1: Performance comparison across tasks for RadiusRatio = 0.1. Bold rows indicate the best overall performance for each task.

Task	H	Avg Timesteps	Success Rate	Feedback / Step	L2 Error / Step
Line Following					
LineFollowing	1	716.2	1.00	0.0039	0.0247
LineFollowing	3	725.0	1.00	0.0006	0.0157
LineFollowing	**5**	**725.4**	**1.00**	**0.0000**	**0.0119**
LineFollowing	8	720.5	1.00	0.0012	0.0235
Push-T					
PushT	1	232.1	0.82	0.1960	0.0032
PushT	2	243.8	0.90	0.2161	0.0033
PushT	**3**	**239.9**	**0.91**	**0.2229**	**0.0034**
PushT	4	247.6	0.54	0.2341	0.0026
PushT	5	246.6	0.64	0.2586	0.0030
PushT	8	268.2	0.66	0.3220	0.0028
PushT	12	327.4	0.14	0.3631	0.0011
PushT	15	323.9	0.12	0.3546	0.0008
MetaWorld Hammer					
Hammer	**1**	**91.2**	**1.00**	**0.1203**	**0.5346**
Hammer	2	91.7	1.00	0.1354	0.5418
Hammer	3	90.2	1.00	0.1687	0.5605
Hammer	4	93.3	0.99	0.2849	0.9911
Hammer	5	102.0	0.81	0.3579	2.3041
Hammer	8	161.7	0.00	0.3773	2.6357
Hammer	12	188.3	0.00	0.3660	4.6139
Hammer	15	211.7	0.00	0.3481	3.7313
PickCan					
PickCan	**1**	**396.2**	**0.94**	**0.0363**	**0.1566**
PickCan	2	442.4	0.94	0.0510	0.1851
PickCan	3	432.1	0.58	0.3391	0.7902
PickCan	5	419.5	0.00	0.3500	3.1282
PickCan	8	356.3	0.00	0.3480	4.2085
Two Arm Lift					
TwoArmLift	**1**	**133.5**	**0.46**	**0.3909**	**6.1425**
TwoArmLift	2	383.7	0.01	0.3769	4.4937
TwoArmLift	3	184.9	0.02	0.3714	8.5940
TwoArmLift	5	420.4	0.00	0.3582	3.7949
TwoArmLift	8	436.4	0.00	0.3511	3.7642
TwoArmLift	12	344.2	0.00	0.3606	5.2890
TwoArmLift	15	414.2	0.00	0.3622	4.1256

6 Conclusion

In this work, we introduced CLIC-Chunks, a framework for learning temporally extended action chunks in continuous control tasks using energy-based models. Our experiments demonstrate that extending desired action spaces into chunks can learn short action chunks while maintaining high performance under accurate feedback in simpler tasks such as Line Following, and scale to more complex manipulation tasks including PickCan, Push-T, and MetaWorld Hammer.

We identify key limitations arising from high-dimensional action chunks: the energy landscape becomes increasingly complex, and Langevin MCMC sampling struggles in these spaces, setting a practical upper bound on effective chunk size. Additionally, flattening predicted chunks across the horizon discards temporal correlations, which can lead to accumulated errors in high-dimensional tasks.

Table 2: Impact of Action Chunk Dimension on Performance

Environment	Obs Dim (dim_o)	Action Dim (dim_a)	Highest Action Chunk	Total Action Size	EBM Sampler Output Dim
MetaWorld Hammer	39	4	5	20	24 (20 + 4)
PushT	24	2	8	16	18 (16 + 2)
TwoArmLift	48	14	1	14	28 (14 + 14)

Table 3: Effect of Radius Ratio (r) on performance (averaged over H). Bold rows indicate the best overall performance for each task.

Task	r	Avg Timesteps	Success Rate	Feedback / Step	L2 Error / Step
PushT	0.10	266.9	0.59	0.271	0.0025
PushT	**0.30**	**267.3**	**0.61**	**0.274**	**0.0026**
MetaWorld-Hammer	**0.10**	**128.6**	**0.60**	**0.270**	**2.114**
MetaWorld-Hammer	0.30	126.0	0.60	0.273	2.585
PickCan	0.01	409.3	0.49	0.225	1.694
PickCan	0.05	419.5	0.47	0.225	1.334
PickCan	**0.10**	**410.1**	**0.49**	**0.226**	**1.419**
PickCan	0.30	429.2	0.36	0.229	1.423
TwoArmLift	0.10	331.6	0.07	0.367	5.458
TwoArmLift	**0.30**	**343.7**	**0.09**	**0.366**	**4.818**

Future work may explore temporal coherence analysis to evaluate trajectory smoothness and identify inconsistencies at chunk boundaries. Increasing the model capacity of the EBM neural network or adopting sequence-based architectures (e.g., LSTM or Transformer encoders) could better capture temporal evolution of actions with the action chunks. Alternatively, regularizing the energy landscape may reduce the sampling difficulty of Langevin MCMC, and flow-matching representations for Q-functions, as in Li et al. [2025b], offer another promising direction.

7 Appendix

7.1 The Setup of Experiments

In all the experiments, we used state-based observations as input for the policy across all methods. The summary of the simulated tasks is reported in Table 4.

7.1.1 Simulated Tasks

The task descriptions are as follows:

- **Push-T:** This task involves the robot pushing a T-shape object to a fixed target location using a circular end-effector.
- **Pick-Can:** The objective is to pick up a can object from the workspace and transport it to a fixed target bin.
- **TwoArm-Lift:** This task involves two robotic arms working cooperatively to lift a shared object.
- **Line Following:** The objective is for the agent to navigate and closely follow a predefined continuous trajectory or line.
- **MetaWorld-Hammer:** The robot must grasp a hammer and use it to drive a nail into a wooden board, requiring precise contact-rich manipulation.

For each task, the object's initial position is randomly initialized at the beginning of each episode to ensure robust policy evaluation.

Table 4: Tasks Summary

Tasks	State dim	Action dim	Multi-modal	Contact Rich
Line Following	5	2	×	×
Push-T	24	2	✓	✓
MetaWorld-Hammer	35	4	×	✓
Pick-Can	40	7	×	×
TwoArm-Lift	48	14	×	×

7.2 Implementation Details

7.2.1 Network Architecture

To accommodate the prediction of temporally extended action sequences, the Energy-Based Model (EBM) $E_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+H-1})$ is parameterized as a Multi-Layer Perceptron (MLP). The input to the network is the concatenation of the state observation vector and the flattened action chunk sequence of dimension $D \times H$. The architecture consists of four hidden layers with dimensions $[512, 512, 512, 256]$. Each hidden layer applies a ReLU activation function immediately followed by Layer Normalization to stabilize training. The final layer outputs a single scalar energy value, and its weights and biases are initialized to zero to prevent large initial energy spikes.

7.2.2 Langevin Dynamics and Action Sampling

To sample from the implicit policy distribution during training and evaluation, we utilize Markov Chain Monte Carlo (MCMC) via Langevin dynamics. During training, counter-example action chunks are generated using 50 Langevin steps initialized from random uniform noise $\mathcal{U}(-1, 1)$. The total number of action samples N_a drawn per state to estimate the partition function and the target policy probabilities is set to 512. During evaluation, the sampling process employs a two-stage refinement: an initial Langevin chain is followed by a fine-tuning stage where the step size decays from 10^{-1} to 10^{-5} for more precise energy minimization.

7.2.3 Optimization and Gradient Penalty

The model is optimized using the Adam optimizer with a base learning rate of 3×10^{-4}. We apply a Cosine Annealing learning rate scheduler with warmup restarts, employing a 10-step warmup period and a minimum learning rate of 10^{-5}. The batch size is set to 32, sampled uniformly from a replay buffer with a maximum capacity of $60,000$ transitions. During the KL divergence loss calculation, the softmax temperature τ is set to 2.0. To prevent exploding gradients, the norm of the network parameters is clipped to 1.0.

To further regularize the highly expressive EBM and ensure a smooth energy landscape, we apply an additional gradient penalty term to the total loss. This penalty computes the gradient of the predicted energy with respect to the input actions, $\nabla_{\mathbf{a}} E_\theta(\mathbf{s}, \mathbf{a})$, and applies a squared hinge loss to restrict the gradient norm to a maximum margin of 1.0.

7.3 Training & Evaluation Results

7.3.1 Line Following

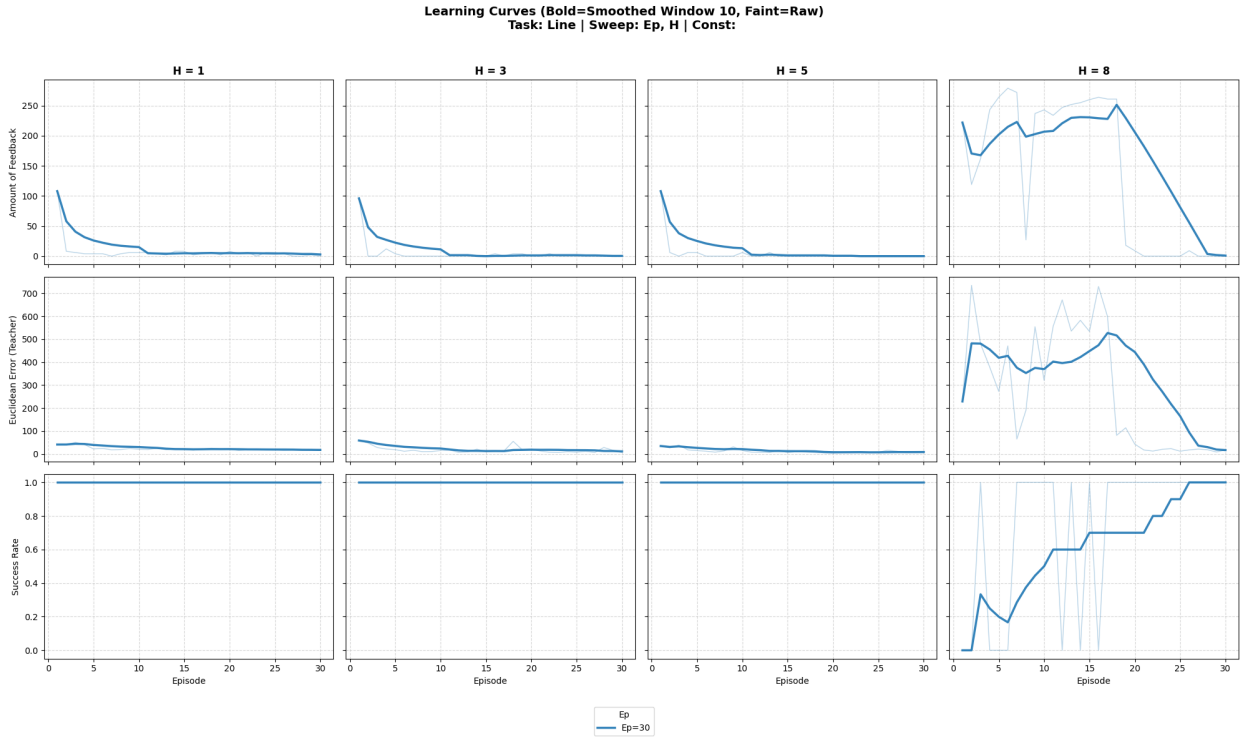

Figure 1: Training & Evaluation Performance of Line Following

7.3.2 PushT

Figure 2: Training & Evaluation Performance of PushT

7.3.3 MetaWorld-Hammer

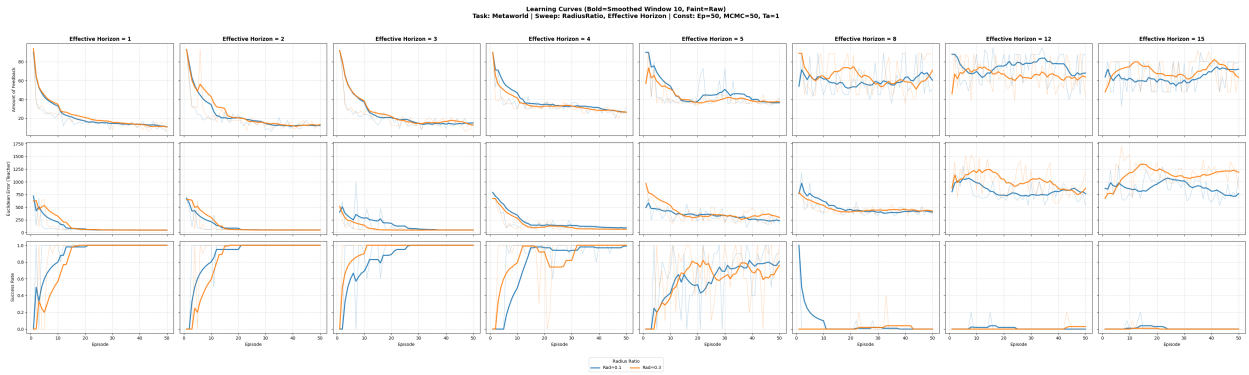

Figure 3: Training & Evaluation Performance of MetaWorld-Hammer

7.3.4 Pick-Can

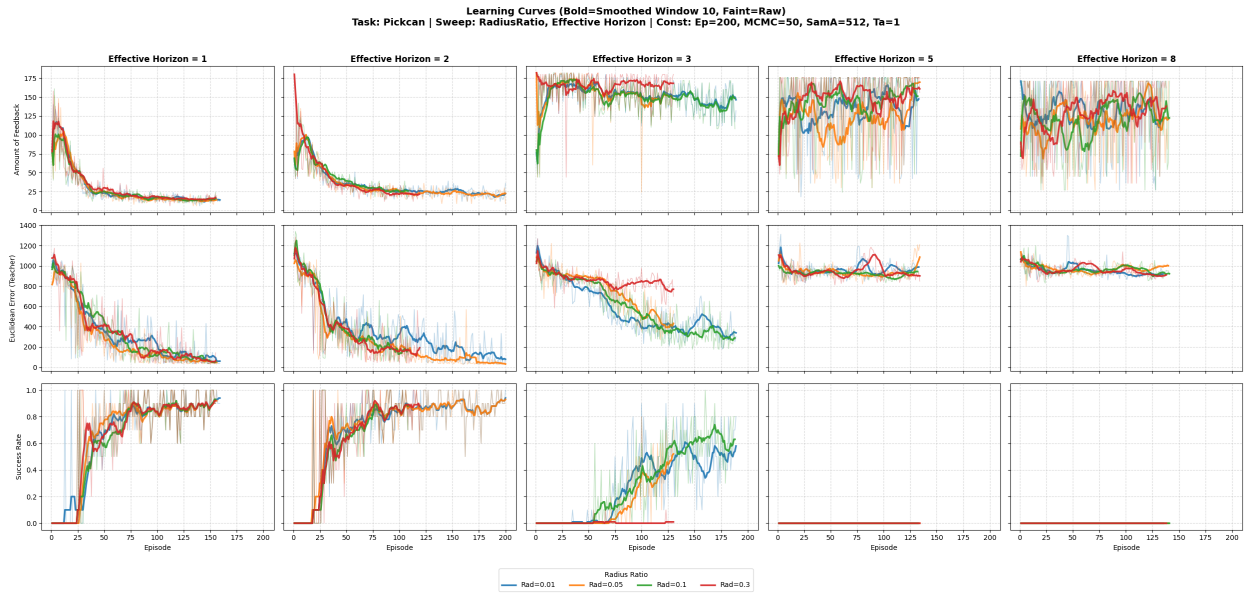

Figure 4: Training & Evaluation Performance of Pick-Can

7.3.5 twoArmLift

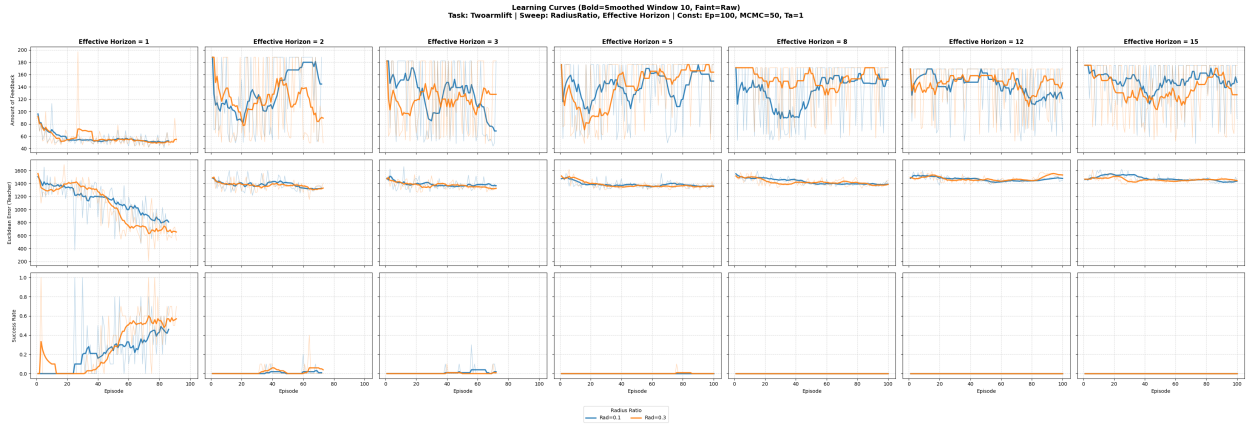

Figure 5: Training & Evaluation Performance of twoArmLift

References

Zhaoting Li, Rodrigo Pérez-Dattari, Robert Babuska, Cosimo Della Santina, and Jens Kober. Beyond behavior cloning: Robustness through interactive imitation and contrastive learning, 2025a. URL https://arxiv.org/abs/2502.07645.

Carlos Celemin, Rodrigo Pérez-Dattari, Eugenio Chisari, Giovanni Franzese, Leandro de Souza Rosa, Ravi Prakash, Zlatan Ajanović, Marta Ferraz, Abhinav Valada, and Jens Kober. Interactive imitation learning in robotics: A survey, 2022. URL https://arxiv.org/abs/2211.00600.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking, 2025b. URL https://arxiv.org/abs/2507.07969.