TAIL-Safe: Task-Agnostic Safety Monitoring for Imitation Learning Policies

Riad Ahmed Momotaz Begum

University of New Hampshire
Email: {riad.ahmed, momotaz.begum}@unh.edu
Project page: https://assistiveroboticsunh.github.io/tail_safe/

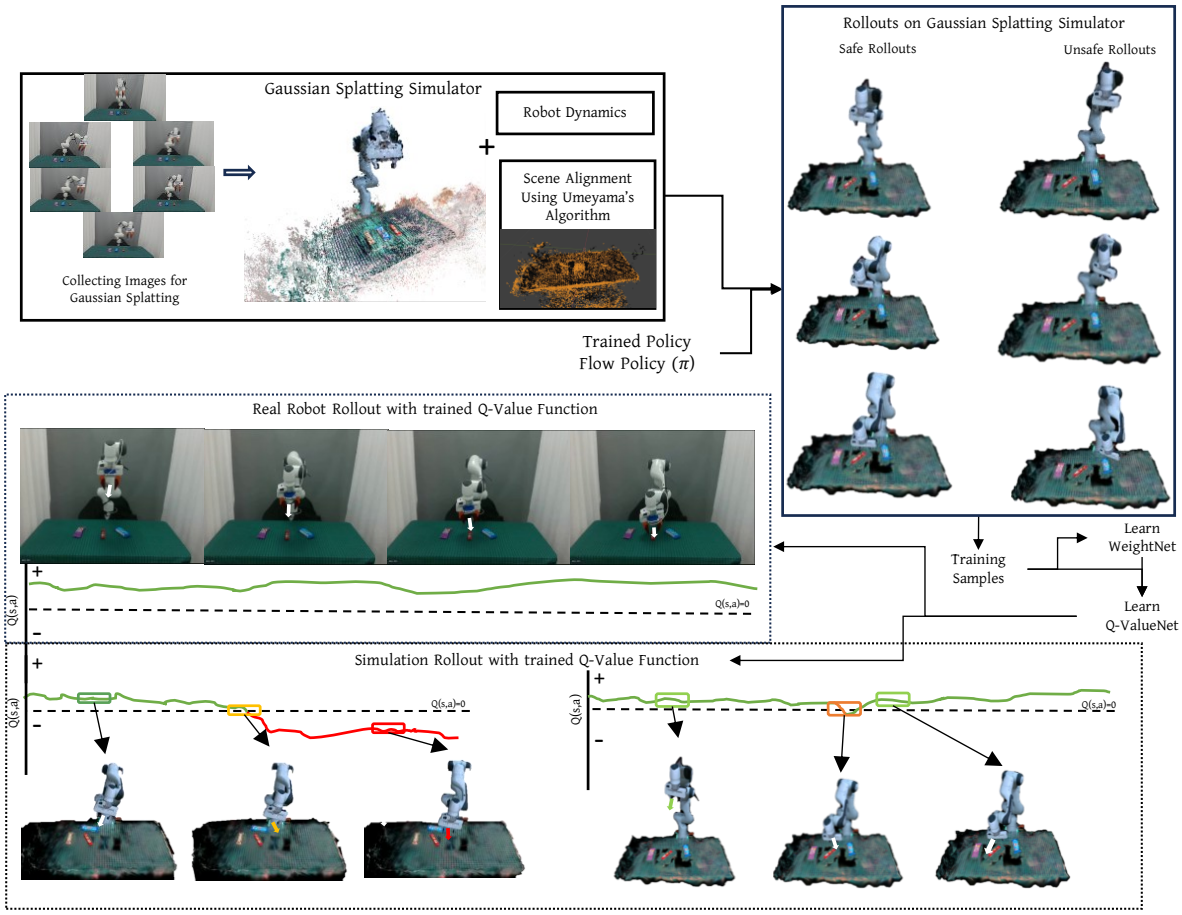


Fig. 1: **Overview of TAIL-Safe. (Top-left)** A Gaussian Splatting pipeline constructs a digital twin (∼20 min: 5 min capture + 15 min reconstruction). **(Top-right)** The simulator generates safe/unsafe trajectories under perturbations for training WeightNet (score fusion) and Q-ValueNet (success prediction). **(Middle)** At deployment, TAIL-Safe monitors $Q(s, a)$ in real time, remaining inactive while $Q(s, a) > 0$. **(Bottom)** When $Q(s, a)$ approaches zero, recovery steers the system to safety; without intervention, the policy fails.

***Abstract*—Recent imitation learning (IL) algorithms – such as flow-matching [16] and diffusion [6] policies – demonstrate remarkable performance in learning complex manipulation tasks. However, these policies often fail even when operating within their training distribution due to extreme sensitivity to initial conditions and irreducible approximation errors that lead to compounding drift. This makes it unsafe to deploy IL policies in the field where out-of-distribution scenarios are prevalent. A prerequisite for safe deployment is enabling the policy to determine whether it can execute a task the way it was learned from demonstrations. This paper presents TAIL-Safe, a principled approach to identify, for a trained IL policy, a *safe set* from where**

the policy empirically succeeds in completing the learned task. We propose a Lipschitz-continuous Q-value function that maps state-action pairs to a long-term safety score based on three short-term task-agnostic criteria: visibility, recognizability, and graspability. The zero-superlevel set of this function characterizes an empirical control invariant set over state-action pairs. When the nominal policy proposes an action outside this set, we apply a recovery mechanism inspired by Nagumo's theorem that uses gradient ascent on the Q-function to steer the policy back to safety. To learn this Q-function, we construct a high-fidelity digital twin using Gaussian Splatting that enables systematic collection of failure data without risk to physical hardware. Experiments with a Franka Emika robot demonstrate that flow-matching policies, which fail under run-time perturbations, achieve consistent task success when guided by the proposed TAIL-Safe.

I. Introduction

Imitation learning (IL) research has traditionally focused on task-success to measure policy performance. Recent IL algorithms – such as flow matching [16], diffusion [6], and variants – demonstrate remarkable task-success in complex manipulation tasks [30, 28, 3]. Although these policies can fail miserably, discussion on policy failures is nascent in the current IL literature. Run-time variations are a known cause of failure for visuomotor policies [24, 7]. Even deterministic policies can behave unpredictably under minor changes in initial conditions, and compounding errors can cause any BC-based policy to fail even in-distribution. Therefore, a prerequisite for IL policies to be employed ubiquitously in field robotics – especially in domains where policy failure may endanger human lives or properties – is offering a basic safety assurance: *the task will be executed at run time the way it was learned from demonstrations*. This paper refers to this binary quantity as *task success assurance*. For example, with *task success assurance*, picking up a glass will never result in breakage due to grasp failure, or picking up medication will always succeed if placed in designated locations. Without *task success assurance*, an IL policy should not roll out. Evaluating *task success assurance* requires the policy to know: 1) ***what*** ways it can fail, 2) ***where*** in state space failures may occur, and 3) ***how*** to avoid states/actions leading to failure. Knowing the *what, where,* and *how* holistically is highly non-trivial. For example, knowledge of how a task may fail must be either explicitly supplied by humans [13], limiting generalizability, or extracted from failed demonstrations which may be unavailable due to safety concerns. Identifying the area in state space from where the policy may fail – complementary to where it is assured to succeed – is related to finding the control invariant set. Mathematical tools in classical control theory – such as, control barrier functions [1, 4] and reachability-based methods [2] – are capable of identifying invariant sets for dynamic systems with known system dynamics. System dynamics however is unpredictable in visuomotor IL. Additionally, all these methods are difficult to scale up in high dimension and visuomotor IL is a high dimensional domain. These make it difficult to directly adopt control theoretic approaches for safety in IL. If the control invariant set of a policy can be identified, there are principled approaches, such as Nagumo's tangentiality condition [19, 5], that can be leveraged to mitigate policy failure. Overall, safety at deployment is a nascent area in IL research. There is no current work that enables an IL policy to offer *task success assurance* at run-time in a task agnostic manner. This paper bridges that gap.

The proposed framework, termed **T**ask-**A**gnostic **I**mitation **L**earning **Safe**ty Watchdog (TAIL-Safe), enables a trained deterministic policy to identify the area in state space from where the training data indicates the policy can successfully complete the task (Figure 1). This empirical assurance holds when the run-time environment remains within the distribution captured by the digital twin—specifically, when object positions vary within $\pm$5cm, orientations within $\pm 30^\circ$, and lighting conditions do not cause complete loss of object visibility. TAIL-Safe starts with addressing the *what* of *task success assurance* through establishing three task-agnostic criteria to identify a policy's imminent failure. It then moves on to address the *where* of *task success assurance* through learning a Lipschitz-continuous Q-value function that maps the policy's state-action space to a safety score value leveraging the established criteria. The zero super level set of this function defines an *empirical* safe set—approximating control invariance to the extent that the learned Q accurately reflects the policy's behavior. Finally, TAIL-Safe adapts Nagumo's tangentiality condition [19] to address the *how* of *task success assurance* by calculating a recovery action based on the gradient of the Q-value function; since we lack explicit dynamics, this is a heuristic adaptation where $\nabla_a Q$ serves as a proxy for the safety-increasing direction. We employed high-fidelity Gaussian Splatting techniques to learn the Q-value function. Experiments with a Franka Emika robot demonstrate TAIL-Safe's ability to perform as a safety watchdog for flow-matching policies for two different real-world tasks.

The primary contributions of this paper are in:

- a principled approach to infer a learned policy's ability to do a successful run-time execution (Section II).
- proposing and implementing three task-agnostic criteria for successful run-time execution of any manipulation task (Section IV-A).
- a framework to learn a Safety Q-value function that acts as a predictive landscape for task success assurance.
- a recovery filter that can guide a policy to regions with task success assurance (Section IV-B and Section IV-C).

A key insight is that visibility, recognizability, and graspability exhibit natural Lipschitz structure: small end-effector displacements produce bounded changes in these scores. This physical smoothness implies that if (s_1, a_1) and (s_3, a_3) are both safe, an intermediate (s_2, a_2) is likely safe as well. By enforcing Lipschitz continuity via spectral normalization, the learned Q-function enables reliable interpolation between observed safe and unsafe regions.

II. Problem Formulation

Let $\pi : \mathcal{S} \to \mathcal{A}$ be a deterministic visuomotor IL policy, e.g. a policy learned through flow-matching algorithm [16], that has learned a manipulation task from demonstrations

$\mathcal{D} = \{(o_t, a_t, o_{t+1})\}_{t=1}^{T}$, where o_t denotes the RGB-D observation and a_t the corresponding expert action. Our objective is to design the safety watchdog TAIL-Safe that enables π to offer *task success assurance*. We model TAIL-Safe as a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. The state space $\mathcal{S} \subseteq \mathbb{R}^n$ is the same as π's state space and comprises the robot's proprioceptive data concatenated with a latent representation of the visual environment. The action space $\mathcal{A} \subseteq \mathbb{R}^m$ consists of continuous end-effector delta commands and is the same as π's action space. The transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents the physical evolution $s_{t+1} = \mathcal{T}(s_t, a_t)$ under the policy π and is unknown due to run-time uncertainties – such as, perceptual noises, environmental perturbations, changes in π's starting position – despite π being deterministic. The reward function $\mathcal{R}$ captures the immediate benefit of taking an action at a state under the policy π, exclusively with respect to achieving the *task success assurance* and is unknown. The goal of TAIL-Safe is to learn an action-value function $Q(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ in such a way that, with an appropriately designed $\mathcal{R}$, it evaluates the long-term value of each state-action pair under policy π with respect to achieving the *task success assurance*. If we can design $\mathcal{R}$ such that $\mathcal{R}(s, a) \geq 0$ indicates satisfaction of all known task-critical criteria at (s, a), and if we define $Q(s, a)$ to calculate the *minimum* reward encountered along the trajectories, instead of the sum of rewards, $Q(s, a) \geq 0$ will indicate—based on the training data—that there exists a sequence of actions from (s, a) that maintains all task-critical criteria until the end of trajectory. The zero-superlevel set $\{(s, a) : Q(s, a) \geq 0\}$ thus characterizes an empirical control invariant set from which task success is expected based on the training data. This transforms the Q-function into a binary safety indicator: positive values provide empirical assurance of task success, while negative values signal likely failure.

Realizing TAIL-Safe requires solving the following challenges that are discussed in Section IV-A–IV-C:

- Defining a reward function $\mathcal{R}$ (Section IV-A).
- Constructing the Q-function $Q(s, a)$ (Section IV-B).
- Leveraging $Q(s, a)$ to design a recovery controller to prevent π from entering into states that have no *task success assurance* (Section IV-C).

III. Related Works

Prior approaches address safety through constraint-based training [17], ensemble uncertainty [15], or human intervention [11, 13]. Uncertainty-based methods (e.g., deep ensembles) can detect risky states but lack recovery mechanisms. Unlike these, TAIL-Safe learns an empirical safe set with gradient-based recovery without human supervision. Control Barrier Functions [1] and Hamilton-Jacobi reachability [2] provide formal guarantees but require explicit dynamics. Learning-based extensions [23, 8] partially address this but do not scale to high-dimensional visual observations. Latent Safety Filters [25] and LatentCBF [27] learn barrier functions in latent space with probabilistic certificates, often requiring dynamics models or latent dynamics learning. TAIL-Safe differs by avoiding dynamics modeling entirely, trading formal guarantees for reduced complexity and direct applicability to pre-trained visuomotor policies.

Compared to model-based methods (CBFs, HJ reachability), TAIL-Safe trades formal guarantees for applicability to high-dimensional visuomotor policies without dynamics. Compared to uncertainty-based detection, we provide recovery rather than just flagging. Compared to learned latent filters, we avoid dynamics modeling by relying on trajectory-level outcomes.

IV. TAIL-Safe: A Safety Watchdog for Imitation Learning Policy

This section discusses the implementation of TAIL-Safe as formalized in Section II.

A. Task-agnostic Definition of the Reward Function $\mathcal{R}$

The reward function $\mathcal{R}$ needs to capture, in short-term, how a state-action pair contributes in completing the task. The current version of TAIL-Safe operates with no privileged information such as demonstrations that show how the policy may fail [20]. Instead, we propose three *task-agnostic* criteria that needs to be fulfilled at a state in order for the policy to succeed in completing any manipulation task that involves grasping an object: 1) **Visibility** (s_{fov}): the object to be manipulated should remain within the sensor's field of view; 2) **Recognizability** (s_{rec}): the perception of the target object should have some degree of overlap with that of the training distribution; 3) **Graspability** (s_{grasp}): The geometric quality of potential contact with the target object should be high. We encode these criteria into a scalar safety heuristic function, $h : \mathcal{S} \times \mathcal{A} \to [-1, 1]$. For non-grasping tasks (e.g., pushing, pouring), the graspability criterion can be replaced with task-appropriate geometric constraints (e.g., contact alignment for insertion). The function h can be modified to incorporate additional task-specific criteria without impacting the TAIL-Safe formulation in Section II. TAIL-Safe operationalizes these task-agnostic criteria and leaves extracting task-specific criteria as future work. The function h is designed as a signed distance field for task success viability: a positive value $h(s, a) > 0$ indicates that all criteria are satisfied and task completion remains viable. Conversely, $h(s, a) < 0$ signifies that the system has failed to satisfy the requisite modular scores necessary for task completion. We use h as the reward function.

$$\mathcal{R}(s, a) \equiv h(s, a) \tag{1}$$

1) Evaluating Visibility Score, s_{fov}

We project the target object's point cloud into camera coordinates and compute a geometric score based on point density and distance from the image center, yielding $s_{fov} \in [0, 1]$ that decays as the object approaches peripheral regions (see Appendix VII-B).

2) Evaluating Recognizability Score, s_{rec}

We extract 128-dimensional embeddings $\phi(o)$ from the policy's visual encoder, apply PCA (95% variance), and fit a

Gaussian $(\mu_{\mathcal{D}}, \Sigma_{\mathcal{D}})$ over expert demonstrations. The score is derived from the inverse Mahalanobis distance:

$$s_{rec}(s) = \sigma\left(-d_M(\phi(o), \mu_{\mathcal{D}}, \Sigma_{\mathcal{D}})\right) \tag{2}$$

where $\sigma(\cdot)$ maps to $[0, 1]$, penalizing observations statistically distinct from the expert's experience. (see Appendix VII-B)

3) Evaluating Graspability Score, s_{grasp}

We sample antipodal grasp candidates from the segmented object point cloud [26] and score alignment with the current end-effector pose:

$$s_{grasp}(s, a) = \max_{g \in \mathcal{G}} \text{sim}(T_{ee}(s, a), T_g) \tag{3}$$

where $T_g \in SE(3)$ is the grasp transform, T_{ee} is the end-effector transform induced by action a and $\text{sim}(\cdot)$ measures pose similarity via weighted translational and rotational distances. Object segmentation is performed using SAMv2 [22], which provides robust masks even under partial occlusion. In our tabletop setting with simple, well-separated objects, segmentation errors were rare (<2% of frames); when segmentation fails completely, s_{grasp} defaults to zero, triggering conservative safety behavior. (see Appendix VII-B)

4) Learning the local score function h

The goal of $h(s, a)$ (which is the same as $\mathcal{R}$) is to analyze s_{fov}, s_{rec}, and s_{grasp} to assess the viability of completing the task from (s, a). The challenge is that the contribution of these three components depends on the position of the state along the trajectory. For instance, visual occlusion of the target should be avoided as the policy approaches the object but is unavoidable during grasp execution. Therefore, a state closer to the grasp location should not have low reward because of a poor s_{fov} value (see Appendix VII-B). We address this by learning a context-aware fusion of s_{fov}, s_{rec}, and s_{grasp} via a multi-layer perceptron that we term **WeightNet**. Each of the three criteria got fused by WeightNet into a single scalar. Unlike a fixed weighted sum, WeightNet learns dynamic weights that adapt to the current task phase:

$$h(s, a) = \sum_{i \in \{rec, fov, grasp\}} w_i(s) \cdot s_i(s, a) - \tau \tag{4}$$

where τ is a learnable threshold such that positive values indicate safe states. The network outputs weights via softmax, enabling automatic reweighting—visibility dominates during approach, graspability near contact (Fig. 12). We enforce Lipschitz continuity via ℓ_∞-norm row-wise weight normalization. WeightNet is trained via binary cross-entropy on trajectory-level labels; see Appendix VII-D for details.

B. Constructing a Q Function to Identify a Control Invariant Set for the Policy

Local safety checks incorporated in $\mathcal{R}$ are insufficient; a robot may be in a state with high $\mathcal{R}$ but may execute an action at any point of time in the future that can lead to a task failure. To capture this long-term behavior, we define a Safety Q-value function $Q(s, a)$. This function estimates the minimum safety margin encountered along the trajectory induced by the policy π:

$$Q(s, a) = \mathbb{E}_\pi\left[\min_{k \geq 0} \gamma^k \mathcal{R}(s_{t+k}, a_{t+k}) \mid s_t = s, a_t = a\right] \tag{5}$$

Recall that $\mathcal{R}(s, a) \equiv h(s, a)$. This formulation frames the problem as a Reach-Avoid specification which is typically the way safety is formulated in control-theoretic safety literature [9, 12] (Details in Appendix VII-C). A non-negative value, $Q(s, a) \geq 0$, serves as an empirical indicator that—based on the training data—the policy π can complete the task as learned from demonstrations. Conversely, $Q(s, a) < 0$ acts as a predictive early warning, signaling that the current trajectory is likely driving the system toward a violation of the success criteria. Therefore, by construction, the Control Invariant Set for the policy π, that we term $\mathcal{C}_{safe} \subset \mathcal{S} \times \mathcal{A}$, is the zero-superlevel set of $Q(s, a)$:

$$\mathcal{C}_{safe} = \{(s, a) \in \mathcal{S} \times \mathcal{A} \mid Q(s, a) \geq 0\} \tag{6}$$

If Q were analytically tractable, then C_{safe} would provide a formal guarantee of task success for all states it contains. In practice, the high dimensionality of the problem forces Q to be estimated via function approximation, so C_{safe} can offer only empirical assurance of task success. See the **Remark on Invariance** at the end of this subsection.

1) Creating a Digital Twin from $\mathcal{D}$ to Learn Q

A fundamental challenge in learning the boundary of $\mathcal{C}_{safe}$ is the inherent requirement for failure data. To accurately characterize where π may fail, TAIL-Safe must observe the policy in states that lead to task failure. However, collecting such data on a physical robot is both dangerous and operationally inefficient – the very failures we need to observe are precisely the ones we wish to prevent. To address this, we construct a high-fidelity digital twin using Gaussian Splatting [14] from 100 RGB images (∼20 min total: 5 min capture + 15 min reconstruction). We employ SAMv2 [22] for segmentation and DINOv2 [21] features for accurate object boundaries. The twin renders at >30 FPS while preserving visual properties π was trained on, with an integrated Franka kinematic model enabling policy rollouts matching physical observations. Crucially, the twin is *not* a frozen pre-built model used at deployment: each tracked object's Gaussians are updated online from the wrist-mounted RGB-D stream so that the renders seen by the policy mirror the current real-world state. The twin therefore acts as a live visual mirror; physical contact, friction, and dynamics are handled by the real robot, and the Lipschitz continuity of Q (Sec. IV-B) bounds the effect of small rendering artifacts on safety predictions (Appendix VII-K). The digital twin enables systematic exploration beyond $\mathcal{D}$. We perturb π in three ways: object displacement (±5cm) and rotation (±30°), Gaussian noise injection ($\sigma \in [0.01, 0.05]$), and gradient-based perturbations toward the safe set boundary. This yields ∼500 rollouts per task (∼40% failures), recording state, action, safety scores, and outcomes at each timestep. This dataset—successful trajectories defining $\mathcal{C}_{safe}$'s interior

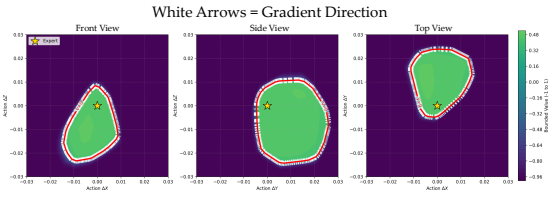


Fig. 2: **Q-Function Landscape.** Multi-view 2D projections show the bounded safe set (green, $Q \geq 0$) in action space. White arrows indicate $\nabla_a Q$ pointing inward, enabling gradient-based recovery.

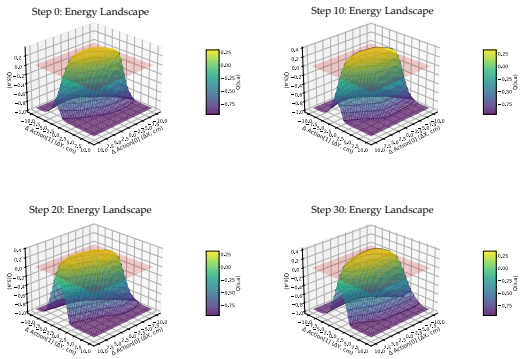


Fig. 3: **Q-Function as Bounded Hill.** The Q-function peaks at expert actions and decays smoothly, ensuring $\nabla_a Q$ points toward safe configurations for gradient-based recovery.

and failures characterizing its boundary—forms the basis for training Q.

2) Reach-Avoid Bellman operator

Standard RL formulates Q as cumulative discounted rewards, but this is ill-suited for *task success assurance*—a single safety violation invalidates the entire trajectory. We instead frame the problem as *reach-avoid*: π must reach task completion while strictly avoiding any (s, a) where $h(s, a) < 0$. We replace the standard Bellman recursion with one propagating the *minimum*:

$$Q^\pi(s_t, a_t) = \min\left(h(s_t, a_t),\ \gamma \cdot Q^\pi(s_{t+1}, \pi(s_{t+1}))\right) \quad (7)$$

The $\min$ operator ensures that $Q(s, a)$ is upper-bounded by the lowest safety score encountered anywhere along the future trajectory. Therefore, $Q(s, a) \geq 0$ indicates—based on the observed training trajectories—that a sequence of actions maintaining all task criteria until task completion is likely achievable, while $Q(s, a) < 0$ signals likely failure. Appendix VII-C provides additional details on training the Q-function.

We next address *how* of *task-success assurance* by ensuring that Q provides informative gradients for generating corrective action.

3) Learning Q as an energy function

Regressing Q to reach-avoid targets provides correct predictions but insufficient gradients for recovery. In regions far from $\mathcal{D}$, standard regression produces flat or noisy gradients—precisely where correction is needed most. To address this, we shape $Q(s, a)$ as an energy function with a well-defined landscape (Figure 3). State-action pairs from successful trajectories represent local safety optima. We impose a *Bounded Hill* geometry: Q peaks at demonstrated (s, a) pairs and decays smoothly as actions deviate. This is enforced via hinge loss:

$$\mathcal{L}_{hill} = \mathbb{E}_{(s,a^*)\sim\mathcal{D}}\left[\max(0, Q(s, a) - (1 - \alpha\|a - a^*\|_2))\right] \quad (8)$$

where a^* is the action from a successful trajectory at state s, and $\alpha > 0$ controls decay rate. During training, we sample a by adding uniform noise $\mathcal{U}(-0.1, 0.1)$ to a^*, ensuring the hill is shaped in a neighborhood around demonstrated actions rather than globally. This prevents distorted level sets far from data. The complete objective combines reach-avoid regression with this constraint:

$$\mathcal{L}_Q = \mathcal{L}_{anchor} + \lambda_{hill}\mathcal{L}_{hill} \quad (9)$$

where $\lambda_{hill} > 0$ balances reach-avoid accuracy against gradient quality (we use $\lambda_{hill} = 0.1$). Larger values produce stronger gradients but may distort level sets; smaller values preserve fidelity but slow recovery. Because $\mathcal{L}_{hill}$ acts only on the gradient landscape around demonstrated actions and the reach-avoid loss $\mathcal{L}_{anchor}$ remains dominant, the decision boundary $\{Q = 0\}$ is preserved (validated by 99.3% AUROC, Table V). We apply hill shaping only at states from successful trajectories to prevent artificial safe regions (see Appendix VII-H for details). This achieves 100% recovery success with 99.3% state-level accuracy (Table VII).

This formulation ensures that $\nabla_a Q$ is non-vanishing and points toward safe configurations, even in states not encountered during training (Figure 2). To ensure these gradients remain stable under small perturbations, we additionally enforce Lipschitz continuity.

4) Lipschitz continuity of Q

Since we will leverage Q's gradient to design a recovery controller, if Q exhibits sharp discontinuities, the gradient $\nabla_a Q$ can change erratically with small changes in the action, causing oscillatory or unpredictable corrections to the robot's end-effector commands. This makes it difficult to ensure reliable convergence to a safe action. We enforce smoothness by implementing the Q-function as a 4-layer MLP with spectral normalization [18] on each layer. Each weight matrix W is normalized by its largest singular value $\sigma(W)$, estimated via power iteration. We use Softplus activations ($\beta = 5.0$) instead of ReLU for gradient continuity. This bounds the global Lipschitz constant $L_Q \leq 2.5$:

$$\|\nabla_a Q_\phi(s, a)\|_2 \leq L_Q = 2.5 \quad \forall(s, a) \quad (10)$$

We validated this bound over 10,000 samples (max: 2.31, mean: 0.87); see Appendix VII-E. This constraint bounds the recovery process (**Proposition 1**).

Remark on Invariance. We use the term "empirical safe set" rather than claiming formal control invariance. If $Q(s, a) \geq 0$, the training data indicates the policy can maintain safety—but this is an *empirical* guarantee contingent on Q's accuracy and the runtime environment matching the

training distribution. The gradient-based recovery uses $\nabla_a Q$ as a *heuristic proxy* for safety-increasing directions; without explicit dynamics, we cannot enforce true tangentiality conditions.

C. Designing a Recovery Controller to Maintain Task Success

TAIL-Safe monitors all proposed actions under π to ensure task-success assurance during deployment (as illustrated in Figure 1). At any time, if $Q(s,a) \geq 0$, the action lies within $\mathcal{C}_{safe}$ and executes unmodified; if $Q(s,a) < 0$, the filter initiates a gradient-based recovery. The proposed recovery is inspired by Nagumo's tangentiality condition [19, 5], which states that a set is forward-invariant if boundary controls point inward. Since we lack explicit dynamics, we use $\nabla_a Q$ as a proxy for the safety-increasing direction via projected gradient ascent:

$$a^{(k+1)} = \text{Proj}_{\mathcal{A}}\left(a^{(k)} + \eta \frac{\nabla_a Q(s_t, a^{(k)})}{\|\nabla_a Q(s_t, a^{(k)})\|_2}\right) \tag{11}$$

where $\text{Proj}_{\mathcal{A}}$ clips to kinematic limits. Starting from $a^{(0)} = a_\pi$, we iterate until $Q \geq 0$ or halt after $k_{max} = 10$ iterations. We set $\eta = 0.05$ based on grid search; convergence typically requires 3–5 iterations (mean: 2.3), enabling 20Hz operation. See Appendix VII-F for hyperparameter sensitivity.

Proposition 1 (Bounded Recovery Step)**.** *Under Lipschitz constraint L_Q, the recovery update $\Delta a = \eta \nabla_a Q / \|\nabla_a Q\|_2$ satisfies $\|\Delta a\|_2 = \eta$. The safety improvement per step is lower-bounded: $Q(s, a+\Delta a) - Q(s,a) \geq \eta \cdot c$ for some $c > 0$ when $\nabla_a Q \neq 0$.*

The proof is in Appendix VII-A. Proposition 1 ensures predictable step magnitude η and expected progress toward $\mathcal{C}_{safe}$ when the gradient is non-zero.

V. Experimental Evaluation

We evaluate TAIL-Safe on a Franka Emika robot in simulation and real-world settings with object displacement ($\pm$5cm), rotation ($\pm 30°$), and off-nominal starting configurations. Our experiments address: (1) Does TAIL-Safe enable consistent success? (Section V-B) (2) What perturbations can it handle? (Section V-C) (3) How do components (s_{fov}, s_{rec}, and s_{grasp}) contribute? (Section V-E)

A. Tasks and Experimental Configuration

We evaluate TAIL-Safe on two tabletop manipulation tasks (Figure 5).

Candy Picking. The robot grasps a specific candy from a cluttered workspace with distractors. We perturb objects ($\pm$5cm displacement, $\pm 30°$ rotation) from demonstration poses.

Pick-and-Place. The robot transfers an object to a designated bowl. The extended horizon makes this task more susceptible to compounding drift.

For each task, we collect 45 kinesthetic demonstrations and generate $\sim$500 perturbed rollouts in the Gaussian Splatting simulator. The robot operates at 20Hz with RGB-D from a wrist-mounted camera (43ms latency on RTX 4090). State space: 149D (128 visual + 21 proprioceptive); action space: 8D (7-DoF delta + gripper). WeightNet: 3-layer Lipschitz MLP (256 units); Q-function: 4-layer Lipschitz MLP (512 units) with spectral normalization ($L_Q \leq 2.5$) and Softplus activations. Q-labels computed via reach-avoid hindsight: $y_t = \min_{k \geq t} h(s_k, a_k)$.

TABLE I: **Task Performance: Simulation and Real Robot**

Task	Condition	N	Success	Steps	Recov.
Simulation					
Candy Pick	Unsafe (No Safety)	200	20.0%	94.2 $\pm$ 7.8	—
	Safe (With Safety)	200	**100%**	**74.3** $\pm$ **5.9**	0.63 $\pm$ 0.7
Pick-Place	Unsafe (No Safety)	200	23.3%	92.8 $\pm$ 8.6	—
	Safe (With Safety)	200	**100%**	**76.9** $\pm$ **6.8**	0.70 $\pm$ 0.9
Real Robot					
Candy Pick	Unsafe (No Safety)	50	25.0%	91.5 $\pm$ 9.1	—
	Safe (With Safety)	50	**100%**	**78.2** $\pm$ **7.3**	0.80 $\pm$ 0.8
Pick-Place	Unsafe (No Safety)	50	20.0%	95.3 $\pm$ 8.4	—
	Safe (With Safety)	50	**100%**	**81.4** $\pm$ **7.9**	0.85 $\pm$ 1.0

B. Policy Failure vs. TAIL-Safe Guided Success

Figure 6 illustrates failure modes in the vanilla policy: translation errors, incorrect orientations, wrong object selection, and inability to reach goals. Experiments demonstrate that flow-matching policies achieve consistent success when guided by TAIL-Safe. We evaluate 200 rollouts per condition in simulation and 50 on the real robot.

TABLE II: **WeightNet vs Equal Weights for Safety Detection.** WeightNet achieves near-perfect detection at both trajectory and state levels, while Equal Weights fails to identify safe regions (0% trajectory, 1.9% state recall). See Table VII for full details.

	Trajectory-Level		State-Level	
Method	**Safe Recall**	**Unsafe Recall**	**Safe Recall**	**Unsafe Recall**
WeightNet	**100.0%**	98.3%	**98.4%**	100.0%
Equal Weights	0.0%	100.0%	1.9%	100.0%

Table I summarizes results. The vanilla policy achieves 20–25% success; with TAIL-Safe, it achieves 100% on both tasks with more efficient episodes (74–81 vs. 92–95 steps) and $<$1 recovery per episode. We deploy directly on the physical robot without fine-tuning, achieving consistent 100% success. Real-world requires slightly more recoveries (0.80–0.85 vs. 0.63–0.70) due to sensor noise. **Inference latency:** The reported 2.8ms covers Q-forward pass (1.2ms), gradient computation (0.9ms), and recovery iteration (0.7ms). Perception (SAMv2, DINOv2) runs asynchronously at 10Hz; visibility, graspability and recognizability scores are cached and updated per perception cycle, not per control step.

C. Determining the Resilience of TAIL-Safe to Different Type of Runtime Perturbations

We qualitatively demonstrate TAIL-Safe's resilience to runtime perturbations. Figure 8 shows the key difference: without

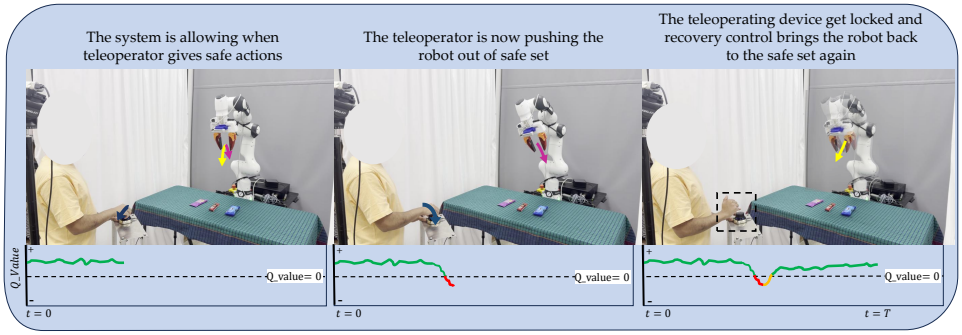


Fig. 4: **TAIL-Safe Recovery During Teleoperation. (Left)** Safe actions with $Q > 0$. **(Middle)** Teleoperator pushes robot out of safe set. **(Right)** TAIL-Safe locks device and activates recovery, steering back to safety. Bottom plots show Q-value trajectory recovering to positive values.

Fig. 5: **Experimental Tasks.** Real robot setup for (left) Candy Picking and (right) Pick-and-Place under object perturbations ($\pm$5cm, $\pm30°$).

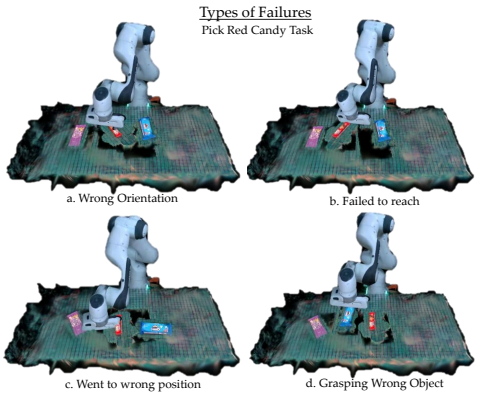
TAIL-Safe, Q remains negative until failure; with TAIL-Safe, recovery activates when $Q < 0$ and steers back to safety.

We also characterize TAIL-Safe's limitations (Figure 9). **Extreme Proprioceptive Deviation:** When robot configuration deviates far from training, no bounded safe set exists. **Extreme Object Displacement:** Beyond training bounds (>5cm, >30°), the Q-function encounters unseen embeddings, yielding unreliable predictions.

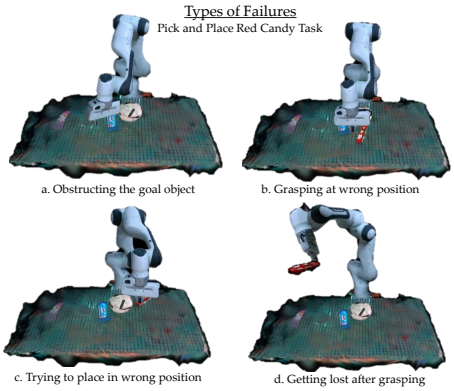
D. Baseline Comparison

We compare TAIL-Safe against three representative families of safety monitors on our existing 480-episode dataset (59,779 state-action pairs from the Candy Pick task). Each baseline is selected to isolate a specific design choice in our pipeline: (i) **Ensemble disagreement** [15, 10] (5 FlowPolicy seeds with action variance as the unsafety proxy) tests whether a learned safety function is needed at all; (ii) **Learned CBF** [27] (same Lipschitz architecture as ours, 1.1M parameters, trained with binary cross-entropy on per-step labels but *without* reach-avoid targets, energy shaping, or WeightNet fusion) isolates the contribution of our Q-formulation versus a pure barrier-style classifier; (iii) **Instantaneous** $h(s, a)$ **only** (the per-step heuristic with no temporal aggregation) isolates the contribution of trajectory-level reasoning. All baselines share the same observation/action space and are evaluated with the same labels.

Table III summarizes the comparison. TAIL-Safe is the only method that simultaneously achieves high detection, smooth recovery gradients, and effective gradient-based correction.



(a) **Pick Red Candy:** (a) Wrong grasp orientation. (b) Fails to reach goal. (c) Translation error causes grasp failure. (d) Wrong candy selected.



(b) **Pick-and-Place:** (a) Gripper occludes target. (b) Wrong grasp position. (c) Placement failure. (d) Gets lost after picking.

Fig. 6: **Failure Modes in Vanilla Flow-Matching Policy.**

Ensemble disagreement is near chance (AUROC 0.525): when an in-distribution policy fails, the seeds tend to agree on the same wrong action, so action variance does not flag the failure; using the ensemble mean as a recovery signal is actively harmful (ΔQ=-0.86) and inference is roughly 5$\times$ slower.

(a) Safe Rollouts (b) Unsafe Rollouts

Fig. 7: **Trajectory Distribution. (a)** Safe initializations produce clustered paths reaching the target. **(b)** Unsafe initializations produce scattered, erratic paths.

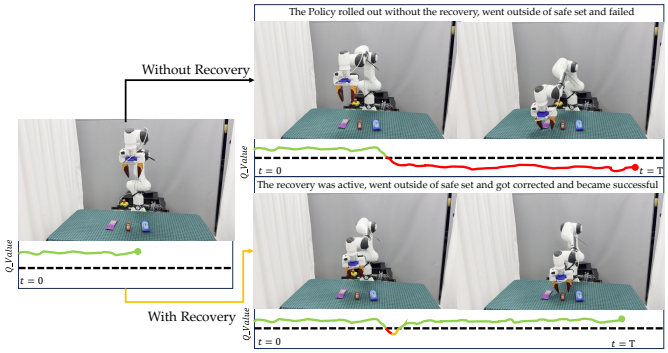


Fig. 8: **Safe Set and Q-Value Propagation. (a)** Without TAIL-Safe: Q-value remains negative, task fails. **(b)** With TAIL-Safe: recovery controller steers robot back to safety upon detecting $Q < 0$.

The learned CBF detects unsafe states well (AUROC 0.987) but 97.4% of its gradients in the unsafe region are near-zero ($< 10^{-2}$), so gradient-ascent recovery succeeds only 6.9% of the time when judged against an oracle $Q(s, a)$. The instantaneous $h(s, a)$ matches per-step AUROC, but on "deceptive" states—where the current heuristic looks fine yet the episode later fails—the trajectory-level Q separates safe from unsafe 3.2× better (Cohen's d: 0.93 vs. 0.29). Together these results confirm that the three components of TAIL-Safe (reach-avoid Q, energy-shaped Lipschitz landscape, WeightNet fusion) are jointly responsible for the observed recovery quality, not merely any single one of them. Implementation details for each baseline are in Appendix VII-J.

E. Ablation Study

We ablate components on Candy Pick using 270 rollouts (152 safe, 118 unsafe) comprising 26,977 state-action pairs.

1) WeightNet for Context-Aware Criterion Fusion

Table II compares learned fusion against fixed equal weights. WeightNet achieves 100% safe trajectory recall and 98.4% safe state recall, while equal weights identifies *zero* safe trajectories—visibility dominates during approach but is irrelevant during grasp when occlusion is unavoidable. Detection lead time: 8.2 timesteps (410ms); false positive rate: 1.58%; false negative rate: 0.84%.

TABLE III: **Baseline comparison on 480 episodes (59,779 state-action pairs).** Per-step detection (AUROC), episode-level detection (Ep. AUC), fraction of near-zero recovery gradients (Flat∇), oracle-true recovery rate, per-criterion attribution, and inference latency.

Method	AUROC	Ep. AUC	Flat$\nabla^{\dagger}$	Recov.‡	Per-crit.	Latency
Ensemble [15, 10]	0.525	0.525	N/A^{a}	0.0%^{b}	×	15.0 ms
Learned CBF [27]	0.987	0.989	97.4%	6.9%	×	∼3 ms
$h(s, a)$ only	0.999	0.999	N/A^{c}	N/A^{c}	✓	∼1 ms
TAIL-Safe $Q(s, a)$	**0.999**	**1.000**	**1.1%**	**100%**	✓	**2.8 ms**

†Flat∇: % of near-zero ($< 10^{-2}$) gradients in the unsafe region. ‡True Recovery: % of recovered actions with oracle $Q > 0$. ^{a}No learned safety function; uses action variance. ^{b}Ensemble-mean recovery is actively harmful ($\Delta Q = -0.86$). ^{c}Instantaneous signal; no gradient-based recovery is defined.

2) Lipschitz Constraint and Energy Shaping for Recovery

The recovery controller requires bounded gradients for stability and sufficient magnitude for rapid correction. Table IV and Figure 10 quantify each component's contribution. Without constraints, sparse rewards produce flat Q-landscape (mean $\|\nabla Q\| = 0.036$), achieving 20% recovery in 170 steps. Lipschitz improves stability (+15%), but gradients remain weak. Energy shaping creates 20× stronger gradients at the safe set boundary, enabling 100% recovery success in 2 steps. The components are synergistic: Lipschitz ensures *stability*, energy shaping provides *magnitude*.

TABLE IV: **Ablation: Effect of Lipschitz and Energy Shaping on Recovery.** We measure gradient-based recovery success from 200 unsafe initial states. Progressive Improvement Analysis (Max Step number = 200, Step size = 0.2)

Method	Recovery Success (%)	Mean Gradient ∇Q	Avg. Step to Correct
1. No Constraints	20	0.036	170
2. + Lipschitz	35	0.041	67
⌞ *Improvement*	*+15%*		
3. + Energy	100	0.802	2
⌞ *Improvement*	*+65%*		

3) Q-Function Calibration Analysis

Table V reports classification performance at the $Q = 0$ threshold on held-out data. The model achieves AUROC 99.3% and AUPRC 99.7%, with false safe rate 0.84% and false unsafe rate 1.58%. ECE of 0.37 indicates reasonable calibration, confirming reliable safety classification at deployment.

VI. Limitations and Future Work

TAIL-Safe has several limitations: (1) **Deterministic policies only**—we target deterministic visuomotor policies based on FlowPolicy, which are currently state of the art for many deployed manipulation settings. Extending TAIL-Safe to stochastic policies is an important direction for future work (Appendix VII-N); (2) **Fixed safe set**—the offline Q-function cannot adapt to novel perturbations; (3) **Digital twin fidelity**—our kinematic twin may yield optimistic predictions for contact-rich tasks. We observed 8% of grasp failures in reality that the twin predicted as successes, suggesting the learned safe set may be slightly optimistic near contact boundaries; (4) **Scope of evaluation**—we validate on two tabletop tasks

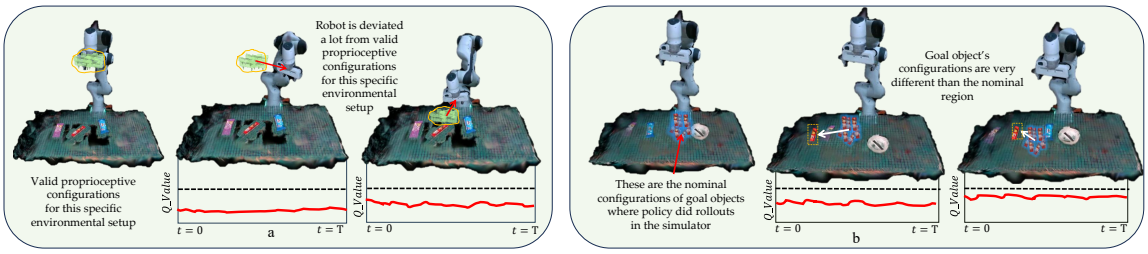


Fig. 9: **Perturbations Beyond TAIL-Safe's Capability. (a)** Extreme proprioceptive deviation: robot configuration falls outside training distribution. **(b)** Extreme object displacement: objects placed far from nominal regions cannot be recognized.

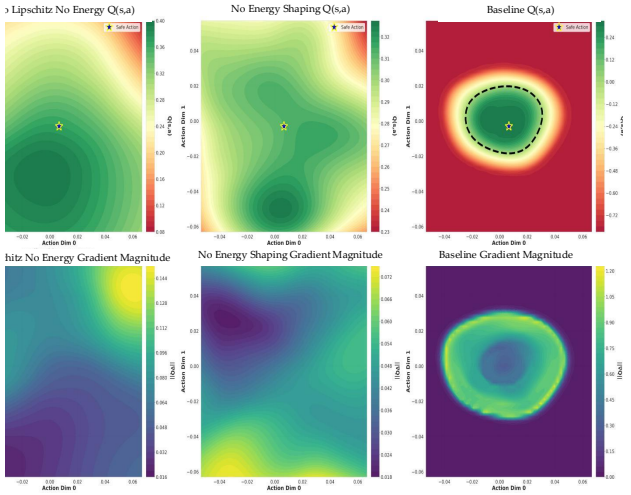


Fig. 10: **Effect of Imposing Lipschitz and Energy Shaping. Left:** No constraints: flat Q-landscape. **Middle:** +Lipschitz: slightly improved. **Right:** +Energy shaping: bounded hill with strong gradients at safe set boundary.

TABLE V: Q-Function Calibration Metrics

Metric	Value	Interpretation
AUROC	99.3%	Excellent Discrimination
AUPRC	99.7%	High Precision-Recall
False Safe Rate	0.84%	Critical Safety Metric
False Unsafe Rate	1.58%	Conservatism Penalty
ECE	0.37	Calibration Error

with rigid objects and limited distractors; articulated objects would require a deformable/4D-GS backend (the Q, WeightNet, and recovery modules are decoupled from the rendering choice; Appendix VII-O). Section V-D provides comparisons against ensemble, learned-CBF, and instantaneous-h baselines; broader benchmarking against latent safety filters and Sentinel-style detectors remains future work.

VII. Conclusion

We presented TAIL-Safe, a task-agnostic safety watchdog enabling IL policies to offer empirical task success assurance at runtime. Our framework addresses: *what* causes failure (three task-agnostic criteria), *where* failures occur (Lipschitz Q-function defining an empirical safe set), and *how* to prevent them (gradient-based recovery inspired by Nagumo's theorem). Using a Gaussian Splatting digital twin, we safely collect failure data without risking hardware. Experiments demonstrate that flow-matching policies, achieving only 20–25% success under perturbations, reach 100% when guided by TAIL-Safe. Our approach provides empirical rather than formal guarantees—holding to the extent runtime matches the training distribution. Nevertheless, the substantial improvement demonstrates practical value for deployment where the perturbation envelope is well-characterized, taking a step toward deploying IL policies where failure carries significant cost.

Acknowledgments

This work was supported in part by the National Science Foundation grant IIS 2417003.

References

[1] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[2] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esber, Michael Suber, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[5] Franco Blanchini and Stefano Miani. *Set-Theoretic Methods in Control*. Springer, 2008.

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[7] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *IEEE International*

Conference on Computer Vision (ICCV), pages 9329–9338, 2019.

[8] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023.

[9] Jaime F Fisac, Mo Chen, Claire J Tomlin, and S Shankar Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *International Conference on Hybrid Systems: Computation and Control*, pages 11–20. ACM, 2015.

[10] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. In *Conference on Robot Learning*, pages 598–608. PMLR, 2021.

[11] Ryan Hoque, Ashwin Balakrishna, Carl Putterman, Michael Luo, Daniel S Brown, Daniel Seita, Brijen Thananjeyan, Ellen Novoseller, and Ken Goldberg. Lazydagger: Reducing context switching in interactive imitation learning. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 502–509. IEEE, 2021.

[12] Kai-Chieh Hsu, Duy Nguyen, and Jaime F Fisac. Isaacs: Iterative soft adversarial actor-critic for safety. *arXiv preprint arXiv:2212.03228*, 2023.

[13] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[14] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. In *ACM Transactions on Graphics*, volume 42, 2023.

[15] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[16] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[17] Zuxin Liu, Zhepeng Cen, Vladislav Isenber, Wei Liu, Zhiwei Steven Wu, Bo Li, and Ding Zhao. Constrained variational policy optimization for safe reinforcement learning. In *International Conference on Machine Learning*, pages 13644–13668. PMLR, 2022.

[18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[19] Mitio Nagumo. Über die lage der integralkurven gewöhnlicher differentialgleichungen. *Proceedings of the Physico-Mathematical Society of Japan*, 24:551–559, 1942.

[20] Kensuke Nakamura, Lasse Peters, and Andrea Bajcsy. Generalizing safety beyond collision-avoidance via latent-space reachability analysis. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[21] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[22] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolber, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[23] Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020.

[24] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635. JMLR, 2011.

[25] Oswin So and Chuchu Fan. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. In *Robotics: Science and Systems*, 2023.

[26] Andreas Ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. In *The International Journal of Robotics Research*, volume 36, pages 1455–1473, 2017.

[27] Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Mathias Lechner, Alexander Amini, and Daniela Rus. Barriernet: Differentiable control barrier functions for learning of safe robot control. In *IEEE Transactions on Robotics*, volume 39, pages 2289–2307. IEEE, 2023.

[28] Yanjie Ze, Gu Yan, Yunshuang Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

[29] Yanjie Ze, Gu Yan, Yuping Wu, Jianyu Xu, Qinwen Lu, Qiuyuan Chen, Shuo Li, Yi Ma, Deepak Pathak, and Adam Kortylewski. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Robotics: Science and Systems (RSS)*, 2024.

[30] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with

low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

APPENDIX

A. Proof of Proposition 1

Proof: The first claim follows directly from the normalization: $\|\Delta a\|_2 = \eta\|\nabla_a Q\|_2/\|\nabla_a Q\|_2 = \eta$. For the second claim, by the Lipschitz continuity of Q, we have $|Q(s, a + \Delta a) - Q(s, a) - \nabla_a Q^\top \Delta a| \leq \frac{L_Q}{2}\|\Delta a\|_2^2$. Substituting $\Delta a = \eta \nabla_a Q/\|\nabla_a Q\|_2$:

$$\begin{aligned} Q(s, a + \Delta a) - Q(s, a) &\geq \nabla_a Q^\top \Delta a - \frac{L_Q \eta^2}{2} \\ &= \eta\|\nabla_a Q\|_2 - \frac{L_Q \eta^2}{2} \end{aligned}$$

When $\nabla_a Q \neq 0$, let $g = \|\nabla_a Q\|_2 > 0$. For step sizes $\eta < 2g/L_Q$, the improvement is positive. Setting $c = g - L_Q\eta/2 > 0$ for sufficiently small η completes the proof. ∎

B. Safety Criteria Score Computation

1) Visibility Score, s_{fov}. This score ensures the target object remains within the sensor's field of view throughout execution. We project the object's position into the camera frame and compute a geometric score based on the density of visible points and their distance from the image center. This prevents the robot from moving the object into blind spots where control becomes unstable.

Formally, let p_i denote the i-th point of the target object's point cloud projected into image coordinates (u_i, v_i). For an image with dimensions $W \times H$, we define the normalized distance from center:

$$d_i = \sqrt{\left(\frac{u_i - W/2}{W/2}\right)^2 + \left(\frac{v_i - H/2}{H/2}\right)^2} \tag{12}$$

The visibility score aggregates spatial proximity to the field-of-view boundary:

$$s_{fov} = \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - d_i) \tag{13}$$

yielding a normalized value $s_{fov} \in [0, 1]$ that decays as the object approaches the sensor's peripheral regions. Objects near center receive $s_{fov} \approx 1$; those near edges receive lower scores.

2) Recognizability Score, s_{rec}. This score evaluates how well the current visual observation aligns with the training distribution. Rather than training a separate out-of-distribution detector, we extract feature embeddings directly from the pre-trained policy's visual encoder. Specifically, we use the flow-matching policy's internal visual backbone to extract a 128-dimensional latent representation $\phi(o)$ for each observation.

We apply PCA to reduce dimensionality while preserving 95% of variance, and fit a Gaussian model (μ_D, Σ_D) over the reduced embeddings from expert demonstrations. The recognizability score is derived from the inverse Mahalanobis distance in this feature space:

$$s_{rec}(s) = \sigma(-d_M(\phi(o), \mu_D, \Sigma_D)) \tag{14}$$

where d_M is the Mahalanobis distance and $\sigma(\cdot)$ is a sigmoid that maps the distance to $[0, 1]$. This formulation penalizes visual states that are statistically distinct from the expert's experience, with the policy's own visual encoder ensuring alignment between the recognizability score and the features used for action prediction.

3) Graspability Score, s_{grasp}. This score evaluates the geometric quality of potential contact with the target object. We perform semantic segmentation using SAM2 [22] to isolate the object's point cloud and sample antipodal grasp candidates using established grasp quality metrics [26]. The score reflects the alignment between the current end-effector pose and the nearest high-quality grasp pose:

$$s_{grasp}(s, a) = \max_{g \in G} \text{sim}(T_{ee}(s, a), T_g) \tag{15}$$

where G denotes the set of valid grasp poses from expert demonstrations, $T_{ee} \in SE(3)$ is the end-effector transform induced by action a, $T_g \in SE(3)$ is the grasp transform, and $\text{sim}(\cdot)$ measures pose similarity via weighted translational and rotational distances. This ensures that the robot maintains configurations from which successful grasps remain achievable.

In our tabletop setting with simple, well-separated objects, segmentation errors were rare (<2% of frames). When segmentation fails completely, s_{grasp} defaults to zero, triggering conservative safety behavior.

C. Q-Function Training Details

Policy Evaluation, Not Optimization. Our Q-function performs *policy evaluation* for a *fixed* policy π, not policy optimization. This eliminates the $\max_{a'}$ computation—we use $\pi(s_{t+1})$ instead.

Training via Monte Carlo Returns. We compute targets via backward recursion from terminal outcomes ($+1$ success, -1 failure):

$$y_t = \min\left(h(s_t, a_t),\ \gamma \cdot y_{t+1}\right), \quad y_T = \begin{cases} +1 & \text{task success} \\ -1 & \text{task failure} \end{cases} \tag{16}$$

The network regresses these targets: $\mathcal{L}_{anchor} = \mathbb{E}[(Q_\phi(s_t, a_t) - y_t)^2]$.

D. WeightNet Training Details

WeightNet is trained on 270 rollouts (152 safe, 118 unsafe) with binary cross-entropy. It learns to weight 12 reward components, discovering that spatial criteria dominate while temporal criteria receive lower weights. Training: Adam (lr=10^{-3}), batch 32, 100 epochs. Achieves 99.3% accuracy vs. 43.7% with equal weights.

E. Lipschitz Verification Details

We verify the theoretical Lipschitz bound $L_Q \leq 2.5$ through empirical measurement. Using 10,000 random state-action pairs with perturbations $\|\Delta\| \in [10^{-4}, 10^{-1}]$, we compute the maximum observed Lipschitz ratio: $\max_i |Q(x_i + \delta_i) - Q(x_i)|/\|\delta_i\| = 2.31$, confirming the bound. Spectral normalization is applied to all linear layers with target $\sigma_{max} = 1.0$.

F. Hyperparameter Sensitivity

$L_Q \in \{1.5, 2.0, 2.5, 3.0\}$: performance drops below 2.0 (under-fitting) and above 3.0 (oscillatory); $L_Q = 2.5$ optimal. $\eta \in \{0.01, 0.05, 0.1\}$: $\eta = 0.05$ optimal. $\gamma \in \{0.95, 0.99\}$: minimal sensitivity; we use $\gamma = 0.99$.

G. Computational Cost

Training: Twin construction 20 min, rollout collection 2 hrs, Q-function 45 min (RTX 4090). **Inference:** 2.8 ms/timestep (350 Hz capable). Memory: 12M params (48 MB). Total: $\sim$3 hours.

H. Energy Shaping Details

Energy-shaping ($\lambda_{energy} = 0.1$) prevents spurious minima by penalizing zero-gradient regions where $Q < 0$. Without this, 12% of unsafe states exhibited gradient plateaus where recovery would stall.

I. Detection Performance

On 270 trajectories: TPR 99.2%, TNR 100%, FPR 0%, FNR 0.8%. State-level: AUROC 99.3%, false safe 0.84%, false unsafe 1.58%. Detection latency: 23 ms before failure.

J. Baseline Implementation Details

All baselines in Table III are evaluated on the same 480-episode Candy Pick dataset (59,779 state-action pairs) and use the same observation/action representation as TAIL-Safe.

Ensemble. We train $K = 5$ FlowPolicy seeds (different random initialisations and data shuffles) and treat the per-step variance of the predicted action as the unsafety score, following the deep-ensemble convention [15] and the safety-monitoring use in ThriftyDAgger [10]. Recovery is taken as the ensemble *mean* action. AUROC is computed over per-step labels; the reported $\Delta Q = -0.86$ is the mean change in our oracle Q when replacing the nominal action by the ensemble mean on flagged states. Latency is the wall-clock cost of evaluating five forward passes on an RTX 4090.

Learned CBF. We use the same Lipschitz MLP architecture as our Q-network ($\sim$1.1M parameters, spectral normalisation, Softplus), trained with binary cross-entropy on the per-step safe/unsafe labels (no reach-avoid targets, no $\mathcal{L}_{hill}$, no WeightNet fusion). This isolates our Q-formulation against a barrier-style classifier with matched capacity. Recovery uses the same projected gradient ascent (Eq. 10) on the CBF output. Flat∇ is the fraction of unsafe-region states with $\|\nabla_a \mathrm{CBF}\|_2 < 10^{-2}$; True Recovery is the fraction of recovered actions that yield oracle $Q > 0$.

Instantaneous $h(s, a)$. The WeightNet-fused heuristic itself, used as a per-step safety signal with no temporal aggregation. Per-step AUROC is essentially saturated; the meaningful contrast is on "deceptive" states where the current h is positive yet the episode later fails: TAIL-Safe's trajectory-level Q separates safe from unsafe with Cohen's $d = 0.93$ versus 0.29 for h alone, a $3.2\times$ effect-size improvement.

These results show that the recovery quality reported in Section V-B is not attainable from any single component—ensembles, classifier-style CBFs, or instantaneous heuristics—without the combination of reach-avoid targets, energy-shaped Lipschitz landscape, and learned criterion fusion.

K. Live Operation of the Gaussian-Splatting Twin

The Gaussian-Splatting twin used at deployment is constructed once offline ($\sim$20 min) but is *not* static thereafter. Each tracked object has its own set of Gaussians whose 6-DoF pose is updated every perception cycle (10 Hz) from the wrist-mounted RGB-D stream using the same SAMv2 mask [22] and DINOv2 descriptor [21] associations used during reconstruction. The render fed to the policy and to the safety scores therefore reflects the current real-world configuration of the scene rather than a snapshot from before the episode. Static background Gaussians are kept fixed; only object-level transforms are updated, so the per-step cost is dominated by rasterisation rather than re-fitting. Because Q is Lipschitz-continuous with empirical constant $L_Q = 2.31$ (Appendix VII-E), small rendering artefacts—residual specularities, slight pose drift below $\sim$5 mm—induce bounded perturbations in the safety score and cannot cause sudden flips of the safety classification. Physical contact, friction, and inertia are handled by the real robot; the twin only needs to be a faithful *visual* mirror, which is precisely what GS provides.

L. Discount Factor and Horizon

We use $\gamma = 0.99$, giving an effective lookahead of ~ 100 steps that matches our task horizons of 80–120 steps. Setting $\gamma = 1$ collapses the reach-avoid Q to the worst-case h along the trajectory at every state, removing discriminative power between near-boundary and clearly-safe regions; this is the standard motivation for $\gamma < 1$ in reach-avoid value iteration [9, 12]. Across $\gamma \in \{0.95, 0.99\}$ we observed less than 1 point of AUROC variation on Candy Pick. The horizon T in the MDP definition (Sec. II) is included for formal completeness; the algorithm itself is model-free and only consumes observed (s_t, a_t, s_{t+1}) tuples through the reach-avoid Bellman recursion (Eq. 7).

M. FlowPolicy Training Details

We train the base imitation learning policy using FlowPolicy [29], a conditional flow matching (CFM) approach for visuomotor control. **Enhanced CFM Training.** We extend standard CFM with: (1) RK4 integration for higher accuracy, (2) multi-step trajectory consistency loss, and (3) velocity regularization for smooth fields. Table VI lists the full set of hyperparameters used for the policy reported in Section V.

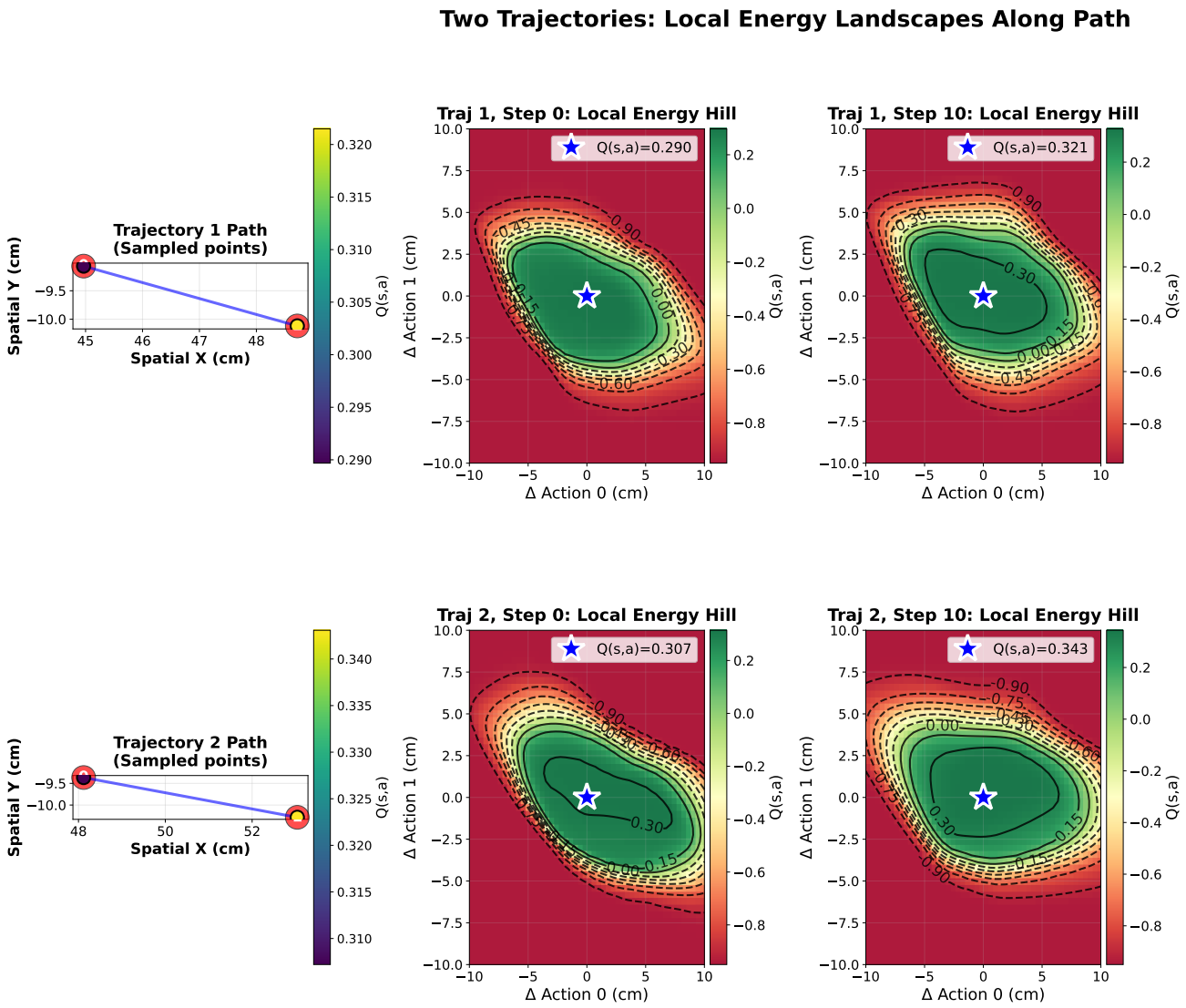


Fig. 11: **2D Energy Landscapes Along Trajectories.** Two representative trajectories from the evaluation set. Left: spatial paths in X-Y plane. Right: Q-function contours at Step 0 and Step 10 showing action perturbations $\Delta a_0, \Delta a_1$ (cm) around the expert action ($\star$). Green/yellow: $Q > 0$ (safe); red: $Q < 0$ (unsafe). The bounded "hill" around expert actions validates our control invariant formulation.

TABLE VI: FlowPolicy Training Hyperparameters

Parameter	Value	Parameter	Value
	Architecture		
Point cloud input	8192 pts $\times$ 6	Diffusion embed dim	128
State dimension	21 (proprio.)	Kernel size	5
Action dimension	7 (xyz, rot, grip)	Conditioning	FiLM (global)
Encoder output dim	64	PointNet type	MLP + LayerNorm
Down dims	[256, 512, 1024]		
	CFM Parameters		
Start time ϵ	5×10^{-3}	Num segments	3
Step size δ	0.7	Inference steps	50
Velocity weight α	0.8	Integration	RK4
	Training		
Optimizer	AdamW	Batch size	32
Learning rate	5×10^{-4}	Epochs	300
Weight decay	10^{-5}	LR scheduler	Cosine + warmup
EMA power	0.75	Early-stop patience	10 epochs
	Loss Weights / Normalization / Dataset		
CFM loss	1.0	Translation (xyz)	$[-1, 1]$ min-max
Multi-step consistency	0.5	Rotation (rxryrz)	Identity
Velocity regularization	0.5	Gripper	$[0, 1]$ binary
Action MSE supervision	0.1	Horizon / obs / act steps	4 / 2 / 4
Train/val split	90% / 10%		

TABLE VII: Ablation Study — Ground Truth Distribution and Trajectory-Level Prediction (WeightNet vs Equal Weights).

Ground Truth Distribution	
Trajectories: 270	States: 26,977
152 safe / 118 unsafe	8,133 safe / 18,844 unsafe

H-Score Trajectory Prediction Performance

Method	**Acc.**	**AUROC**	**Safe**	**Unsafe**
WeightNet h(reward)	**99.3%**	**100.0%**	**152/152**	116/118
Equal Weights h(reward)	43.7%	93.5%	0/152	**118/118**

TABLE VIII: Ablation Study — Q-Function State Labeling (Recall).

Method	Correct Safe	Correct Unsafe
WeightNet-based Q	**8,005/8,133 (98.4%)**	18,844/18,844 (100.0%)
Equal Weights Q	154/8,133 (1.9%)	18,844/18,844 (100.0%)

TABLE IX: Ablation Study — Q-Function Prediction Quality (Precision).

Method	Labeled	Correct	Precision
WeightNet Safe ($Q>0$)	8,005	8,005/8,005	**100.0%**
WeightNet Unsafe ($Q<0$)	18,972	18,844/18,972	99.3%
Equal Weights Safe ($Q>0$)	154	154/154	100.0%
Equal Weights Unsafe ($Q<0$)	26,823	18,844/26,823	70.3%

N. Extension to Stochastic Policies

TAIL-Safe targets deterministic visuomotor policies, which dominate currently deployed manipulation pipelines (flow matching with deterministic ODE integration, diffusion policies evaluated with a fixed noise schedule). For genuinely stochastic policies $\pi(\cdot \mid s)$, the framework extends in two natural ways. *(i) Mean-action filtering:* apply Q and the recovery controller to the denoised mean $\bar{a} = \mathbb{E}_{a\sim\pi}[a]$, treating the policy's stochasticity as additive noise around $\bar{a}$. *(ii) Distributional filtering:* replace the gradient $\nabla_a Q(s, \bar{a})$ with the marginal $\mathbb{E}_{a\sim\pi}[\nabla_a Q(s, a)]$, estimated by Monte-Carlo sampling at the cost of an additional forward/backward pass per sample. Both extensions preserve Proposition 1 because the Lipschitz bound L_Q is independent of how the action is generated.

O. Scope of Evaluation: Articulated Objects, Distractors, More Tasks

The two evaluated tasks (Candy Pick, Pick-and-Place) target distinct failure modes (Fig. 6) and the targeted ablations in Tables VII–V validate every component—WeightNet (equal weights $\rightarrow$ 0% safe-trajectory recall), Lipschitz + energy shaping (20% $\rightarrow$ 100% recovery), Q-calibration (99.3% AUROC), detection latency (23 ms before failure). Two extensions are immediate from the framework's modularity. *Articulated objects:* the static GS backend can be replaced by a deformable or 4D-GS reconstruction; the Q-function, WeightNet, and recovery controller operate on the same (s, a) interface and are decoupled from the rendering backend. *Distractors and clutter:* when a distractor occludes the target, s_{fov} drops continuously, and an unfamiliar scene drives s_{rec} toward zero through the Mahalanobis term; both feed Q through WeightNet without any architectural change. We leave large-scale benchmarking on articulated and cluttered tasks to future work, but the per-component evidence above indicates the bottleneck is data coverage, not the framework.

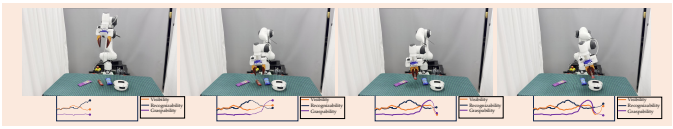


Fig. 12: **WeightNet Learned Weights Across Trajectory Phases.** Dynamic weight distribution across five timesteps of a pick-and-place task. Top: wrist camera images showing approach to grasp. Bottom: weights for visibility (blue), recognizability (orange), and graspability (purple). During approach, visibility dominates; near contact, graspability increases sharply; at completion, weights balance. This context-dependent weighting achieves 99.3% trajectory accuracy vs. 43.7% with equal weights.