How to Train Your Latent Control Barrier Function: Smooth Safety Filtering Under Hard-to-Model Constraints

Kensuke Nakamura KENSUKEN@CMU.EDU
Arun L. Bishop ARUNLEOB@CMU.EDU
Steven Man STEVENWMAN@CMU.EDU
Aaron M. Johnson AMJ1@CMU.EDU
Zachary Manchester ZACM@CMU.EDU
Andrea Bajcsy ABAJCSY@CMU.EDU
Carnegie Mellon University

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

Latent safety filters extend Hamilton-Jacobi (HJ) reachability to operate on latent state representations and dynamics learned directly from high-dimensional observations, enabling safe visuomotor control under hard-to-model constraints. However, existing methods implement "least-restrictive" filtering that discretely switch between nominal and safety policies, potentially undermining the task performance that makes modern visuomotor policies valuable. While reachability value functions can, in principle, be adapted to be control barrier functions (CBFs) for smooth optimization-based filtering, we theoretically and empirically show that current latent-space learning methods produce fundamentally incompatible value functions. We identify two sources of incompatibility: First, in HJ reachability, failures are encoded via a "margin function" in latent space, whose sign indicates whether or not a latent state is in the constraint set. However, representing the margin function as a classifier yields saturated value functions that exhibit discontinuous jumps. We prove that the value function's Lipschitz constant scales linearly with the margin function's Lipschitz constant, revealing that smooth CBFs require smooth margins. Second, reinforcement learning (RL) approximations trained solely on safety policy data yield inaccurate value estimates for nominal policy actions, precisely where CBF filtering needs them. We propose the **LatentCBF**, which addresses both challenges through gradient penalties that lead to smooth margin functions without additional labeling, and a value-training procedure that mixes data from both nominal and safety policy distributions. Experiments on simulated benchmarks and hardware with a vision-based manipulation policy demonstrate that **LatentCBF** enables smooth safety filtering while doubling the task-completion rate over prior switching methods. Project Page: https://cmu-intentlab.github.io/latent_cbf/

Keywords: Safety Filtering, World Models, Reachability, Control Barrier Functions

1. Introduction

In theory, safety filters—such as Hamilton–Jacobi (HJ) reachability (Mitchell et al., 2005), control barrier functions (CBFs) (Ames et al., 2017), or model-predictive shielding (Bastani, 2021)—can monitor and correct *any* nominal policy to prevent safety violations. Yet a substantial gap remains between theory and practice. One cause for this gap is a shift in how robot policies are designed. For example, visuomotor manipulation policies (Chi et al., 2024; Intelligence et al., 2025) increasingly operate end-to-end, performing complex tasks directly from RGB camera inputs. Such policies are

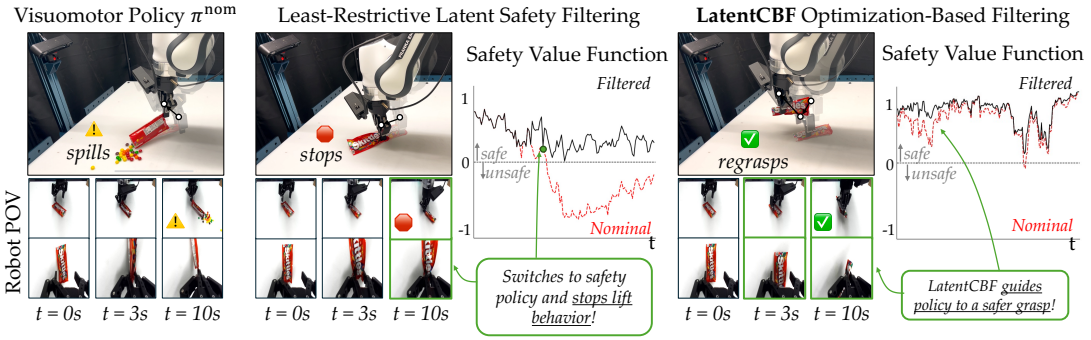

Figure 1: **Safety Filtering a Visuomotor Manipulation Policy.** Both the nominal policy and the safety filters take as input the RGB images shown on the bottom. (left) Unfiltered nominal diffusion policy spills the bag's contents. (center) Least-restrictive latent safety filter prevents spilling, but also stops the diffusion policy from lifting the bag. (right) Our **LatentCBF** guides the diffusion policy to a safe grasp and it completes the pickup task.

deployed in conditions that violate many assumptions underlying classical safety filters: state representations are complex and partially observable (e.g., deformables), dynamics models or simulators may be unavailable, and the safety constraints are often extremely hard to specify analytically (e.g., spilling, as seen in left of Figure 1).

To align a safety filter's assumptions with those of a nominal visuomotor policy, recent work introduced *latent safety filters* that operate on learned latent state spaces and dynamics (world) models trained directly from high-dimensional observations (Nakamura et al., 2025). These methods represent safety constraints as the level set of a "margin function," implemented as a classifier trained on labeled failure observations whose sign indicates a failure state. An approximate HJ-reachability problem is then solved in the latent space to derive a safety value function and fallback policy, enabling visuomotor filters that, for example, prevent a manipulator from spilling a bag's contents using only wrist and third-person RGB images (Figure 1). However, existing approaches rely on "least-restrictive" filtering that discretely switches between nominal and safety policies, degrading the nominal policy's performance. A natural alternative is optimization-based filtering, as in CBF methods (Ames et al., 2019), to minimally alter the nominal policy while maintaining safety. Since recent work has adapted HJ-based value functions into CBF-like formulations (Choi et al., 2021; Tonkens and Herbert, 2022; Oh et al., 2025), it is compelling to try to use the latent-space value function directly for such filtering.

In this paper, we theoretically and empirically show why existing latent safety filters are *incompatible* with smooth, CBF-style filtering in latent space, and we propose algorithmic solutions that make such filtering feasible in practice. The incompatibility arises from how reachability-based value functions are approximated in latent space. First, encoding constraints via a classifier prevents the HJ value function from learning smooth gradients, leaving safety filters unable to assess relative action safety until the last moment (see Figure 2). Second, in high-dimensional latent spaces (≥ 512), actor–critic reinforcement learning (RL) is typically used to approximate the safety value function. However, this creates a distribution mismatch at deployment: the value function, trained on conservative data from the safety policy, is used to evaluate a nominal policy that explores different states and actions, yielding poor value estimates where CBF-style filters need them the most.

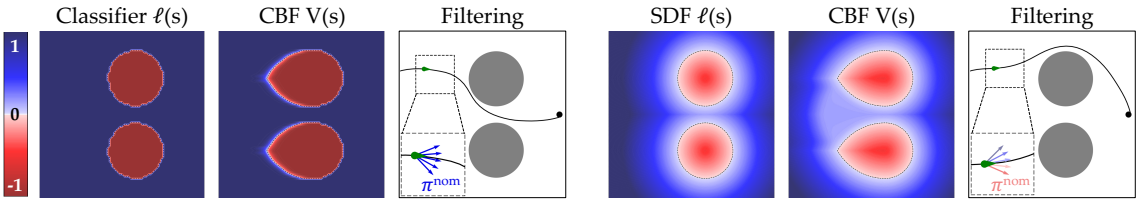

Figure 2: **CBFs as a Function of the Margin Function.** Even with a perfect model, a classifier-based $\ell(s)$ yields a CBF with poor signal during action filtering (left). A smooth margin function provides a rich signal for the CBF to evaluate alternative actions (right).

Our contributions address both sources of incompatibility and yield latent discrete-time CBFs (called **LatentCBF**) suitable for optimization-based safety filtering directly from observations (Figure 1, right). First, we prove that, under mild conditions, the Lipschitz constant of the safety value function scales linearly with that of the margin function, showing that smooth value functions require smooth margin functions—a property violated by binary classifiers. This insight motivates our second contribution: a Wasserstein GAN-inspired (Arjovsky et al., 2017; Gulrajani et al., 2017) training method that learns smooth margin functions "for free," using only the same binary safe/fail labels as done in prior work. Third, we introduce a mixed exploration strategy that blends state–action samples from both nominal and safety policies, ensuring accurate value estimates across the regions critical for CBF filtering. We evaluate LatentCBF in a benchmark simulation with privileged CBF access and in a vision-based Franka manipulation task, finding that it enables 45% smoother interventions in simulation and doubles ($38\% \rightarrow 80\%$) the safe-task success rate of visuomotor manipulation policies on hardware compared to least-restrictive safety filters.

2. Mathematical Background

Consider the discrete-time system governed by bounded dynamics $s_{t+1} = f(s_t, a_t)$ where $s_t \in \mathcal{S}$ is the state of the system at time t and the action $a_t \in \mathcal{A}$ is selected from a bounded control set, $\mathcal{A}$. Let the set of states that are already in failure (i.e., the system has already violated safety) be denoted by the failure set: $\mathcal{F} \subset \mathcal{S}$. For example, $\mathcal{F}$ could represent states where a robot vehicle is in collision, or a state where the contents of a bag have already spilled out during robot manipulation.

Definition 1 (Discrete-time Control Barrier Function (Agrawal and Sreenath, 2017)) *A function $B : \mathcal{S} \rightarrow \mathbb{R}$ is a discrete-time control barrier function (CBF) if $\forall s \in \mathcal{S}$ the function B satisfies*[1]

$$\exists a \in \mathcal{A} \text{ s.t. } B(f(s,a)) \geq \alpha B(s), \text{ for } \alpha \in [0,1), \tag{1}$$

and $\Omega \cap \mathcal{F} = \emptyset$ where $\Omega := \{s \mid B(s) > 0\}$.

The condition (1) renders the zero superlevel set Ω a *control invariant safe set*. This means for any state $s \in \Omega$, then there exists an action a s.t. $f(s,a) \in \Omega$. Since this control invariant set does not intersect with $\mathcal{F}$, states in this set can avoid entering $\mathcal{F}$ for all future time. Discrete-time CBF filters find the minimal action adjustment satisfying these constraints through the following optimization:

$$a^\star = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \, ||a - \pi^{\text{nom}}(s)||, \quad \text{s.t. } B(f(s,a)) \geq \alpha B(s) \tag{2}$$

1. In full generality, the right side of the inequality in (1) can be an extended class κ_∞ function. For ease of presentation, we restrict the notation to linear functions of $B(s)$ parameterized by α.

where $\alpha \in [0, 1)$ is a parameter that dictates how "quickly" the safety filter will begin to override the nominal policy π^{nom} as the system approaches the boundary of the control invariant set.

Hamilton-Jacobi (HJ) Reachability. While the optimization problem in (2) defines a "minimally invasive" safety filter problem, a well-known drawback of CBFs is the challenge of constructing a valid function $B(\cdot)$ that satisfies Definition 1. We build upon a recent line of work (Choi et al., 2021; Tonkens and Herbert, 2022; Oh et al., 2025) that proposes the use of Hamilton-Jacobi value functions as a CBF[2]. HJ reachability begins by defining a *margin function* $\ell(s) : \mathcal{S} \to \mathbb{R}$, which implicitly defines the failure set[3] as the zero-sublevel set $\mathcal{F} := \{s \mid \ell(s) < 0\}$. For example, in hand-designed state spaces, this is often a signed distance function to the boundary of the constraint. Then one can obtain a safety value function $V^{\textbf{🛡}} : \mathcal{S} \to \mathbb{R}$ satisfying the discrete-time Hamilton-Jacobi fixed-point equation (Mitchell et al., 2005; Fisac et al., 2019):

$$V^{\textbf{🛡}}(s) = \min\{\ell(s), \max_{a \in \mathcal{A}} V^{\textbf{🛡}}(f(s, a))\}, \tag{3}$$

with a zero-superlevel set representing the *maximal* control-invariant safe set. Mathematically, $V^{\textbf{🛡}}(s) \geq 0 \iff s \in \Omega$ and $s \notin \mathcal{F}$, and $V^{\textbf{🛡}}$ satisfies (1) for all states in its zero-superlevel set (Oh et al., 2025). This implies that $V^{\textbf{🛡}}(s)$ computed via reachability-based techniques can be used as a discrete-time CBF (i.e., $B(s) = V^{\textbf{🛡}}(s)$). However, solving (3) faces the *curse of dimensionality*, as computation grows exponentially with state dimension. To address this, neural approximations (Bansal and Tomlin, 2020; Hsu et al., 2024) have emerged, which, despite lacking formal guarantees, scale safety filtering to high-dimensional systems with strong empirical safety.

3. Problem Formulation: Safety Filtering via Latent Control Barrier Functions

Unlike traditional safety filters from Section 2 that assume access to fully-observed dynamical systems, we do *not* assume access to a hand-designed model of $s \in \mathcal{S}$, dynamics $f(s, a)$, or the failure set $\mathcal{F}$. Instead, we only assume access to a dataset of trajectories consisting of observations, $o_t \in \mathcal{O}$, (e.g., RGB images and proprioceptive state), and robot actions, $a_t \in \mathcal{A}$. Let this dataset of diverse observation-action trajectories be $\mathcal{D} = \{(o_t, a_t, o_{t+1})_i\}_{i=1}^N$ collected from the robot interacting with its environment. This setting is common modern visuomotor policies such as diffusion or flow-matching policies (Chi et al., 2024; Intelligence et al., 2025). We also assume that we can identify if $s_t \in \mathcal{F}$ solely from the observation o_t (e.g., whether the contents of a bag spilled on the table) and have access to binary labels indicating as such. Our goal is to bring optimization-based filtering, like that of CBFs, closer to the vision-based capabilities of modern visuomotor policies. To do this we leverage the paradigm of latent world models.

Latent State & Dynamics. Latent world models (Ha and Schmidhuber, 2018) jointly learn a latent state space representation $z \in \mathcal{Z}$, an encoder from observations to latent states $\mathcal{E}(o) : \mathcal{O} \to \mathcal{Z}$, and a (potentially stochastic) dynamics model $f_z : \mathcal{Z} \times \mathcal{A} \to \Delta(\mathcal{Z})$, where Δ denotes a distribution, given real-world robot dataset, $\mathcal{D}$. For brevity, we only introduce the necessary mathematical models here and provide additional background in the Appendix.

Latent-Space Safe Control. Prior work has shown it is possible to approximately solve (3) in latent spaces using actor-critic reinforcement learning (Nakamura et al., 2025). These methods

2. These works relax the discrete-time CBF constraint to hold only for $\{s \in \mathcal{S} \mid B(s) \geq 0\}$, which still ensures control invariance of the safe set at the expense of the set attractivity property of traditional CBFs (Cortez et al., 2021).
3. In general, the zero-superlevel set of $\ell(s)$ is *not* control-invariant, and thus cannot be used directly as a CBF.

approximate the safety state-action value function $Q^{\shield} : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ (critic) and a learned safety fallback policy $\pi^{\shield} : \mathcal{Z} \to \mathcal{A}$ (actor) via the latent time-discounted HJ fixed point equation:

$$Q^{\shield}(z,a) = (1-\gamma)\ell(z) + \gamma \min\left\{\ell(z),\ \max_{a' \in \mathcal{A}} \mathbb{E}_{z' \sim f_z(z,a)} Q^{\shield}(z',a')\right\}, \tag{4}$$

where the safety fallback policy $\pi^{\shield}(z)$ learns $\operatorname{argmax}_{a \in \mathcal{A}} Q^{\shield}(z,a)$. Here, $\gamma \in [0,1)$ is the discount factor that recovers (3) as $\gamma \to 1$, in the sense that $V^{\shield}(z) = \max_{a \in \mathcal{A}} Q^{\shield}(z,a)$ (Fisac et al., 2019).

Latent Discrete-Time CBF. At runtime, our goal is to solve the following latent discrete-time CBF optimization problem to minimally steer any visuomotor policy $\pi^{\text{nom}} : \mathcal{O} \to \mathcal{A}$ such that it can effectively perform its task while staying within the zero-superlevel set of $V^{\shield}(z)$:

$$a^\star = \operatorname*{argmin}_{a \in \mathcal{A}} ||a - \pi^{\text{nom}}(o)||, \quad \text{s.t. } Q^{\shield}(z,a) - \epsilon \geq \alpha\left(Q^{\shield}(z, \pi^{\shield}(z)) - \epsilon\right), \tag{5}$$

where $z = \mathcal{E}(o)$ is obtained by querying the world model's encoder on the same observations that are passed into the nominal visuomotor policy. The hyperparameter ϵ is a small positive constant used to account for any learning inaccuracies of the zero-level set. Note that, if we query the state-action value function with the *safety fallback policy's* action, then we obtain $V^{\shield}(z) \approx Q^{\shield}(z, \pi^{\shield}(z))$; this is the "safest" we could ever be at this latent state. If we want to evaluate the safety of any *other* action $a \in \mathcal{A}$, we can query $Q^{\shield}(z,a) \approx V^{\shield}(f_z(z,a))$, which forms the left-hand side of the inequality constraint. Thus, we reformulate the CBF-style safety filter from (2) where the latent state-action value function learned via (4) is our discrete-time CBF (Oh et al., 2025).

4. The Theory-Practice Gap for Latent Control Barrier Functions

Recall that, for HJ reachability to compute the discrete-time CBF via (4), we need a margin function $\ell(z)$ that encodes the set of latent states that "appear" to be in failure, i.e., $\ell(z) \leq 0 \iff z \in \mathcal{F}_z$. How can we obtain the latent margin function from a dataset of observation-action tuples in $\mathcal{D}$? Prior work (Nakamura et al., 2025) asks a stakeholder to look at the observations $o \in \mathcal{D}$ and assign binary labels to those that look like they are in failure (e.g., contents of the bag are spilled on the table). This is because, in practice, it is far easier to label an image from the robot's point of view (as shown in the bottom row of Figure 1) as a failure or not than to provide per-frame and real-valued labels indicating *how close* the robot is to failure.

Classification-Based Latent Margin Functions. This labeling procedure results in a dataset of $z^+ \in \mathcal{D}_{\text{safe}}$ and $z^- \in \mathcal{D}_{\text{fail}}$ for safe and failed latent states, respectively, obtained from labeled observations encoded via the world model's encoder $\mathcal{E}(o) = z$. This is used to train a classifier that learns to discriminate latent states coming from visually failed observations from visually safe ones. The training objective for a bounded[4] classifier-based $\ell_\mu(z)$ with trainable parameters μ is:

$$\mathcal{L}^{\delta}_{\text{sign}}(\mu) = \mathbb{E}_{z^+ \sim \mathcal{D}_{\text{safe}}}\left[\min\{0, \delta - \ell_\mu(z^+)\}\right] + \mathbb{E}_{z^- \sim \mathcal{D}_{\text{fail}}}\left[\min\{0, \delta + \ell_\mu(z^-)\}\right], \tag{6}$$

where parameter $\delta \in \mathbb{R}^+$ prevents $\ell_\mu(z)$ from converging to degenerate solution. Since we assume that we can identify if $s \in \mathcal{F}$ from the observation o, this encourages $\ell_\mu(z) \leq 0 \iff s \in \mathcal{F}$.

4. To ensure the existence and uniqueness of the solution of (4), the margin function $\ell_\mu(z)$ must be bounded, which can be enforced by using $\tanh(\cdot)$ as the final activation of the margin function or clipping its final output.

Challenge 1: Smooth Value Functions Need Smooth Margin Functions. In theory, the loss function in (6) is sufficient for obtaining a margin function that makes the HJ fixed point solution from (3) well-defined (Bellman, 1952). In practice, however, the margin function has a large Lipschitz constant due to the near-discrete jumps at the boundary of the failure set. Together with the need for $\ell_\mu(z)$ to be bounded, this results in "saturated" value functions uninformative for solving (5). We show a classifier-based margin function and the resulting discrete-time CBF obtained via grid-based numerical solvers (Schmerling, 2024) for a 3D Dubins' car (details in 6.1). During optimization-based filtering, saturated gradients prevent the CBF from assessing how action changes affect long-term safety (center, Figure 2, where all sampled actions yield similarly high safety values). In contrast, a smooth margin function produces a more sensitive CBF that distinguishes safer from riskier actions, scoring roughly half of the sampled actions as less safe (right, Figure 2).

We characterize this relationship by proving that even under the perfect state and deterministic dynamics, the Lipschitz constant $L_{V^{\shield}}$ of the discounted HJ value function scales *linearly* in the Lipschitz constant L_ℓ of the margin function.

Definition 2 (Lipschitz Continuity) *A function $f : \mathcal{S} \to \mathbb{R}$ is* Lipschitz continuous *if there exists a constant $L_f \in \mathbb{R}_{\geq 0}$ such that $|f(s) - f(\tilde{s})| \leq L_f \|s - \tilde{s}\|, \quad \forall s, \tilde{s} \in \mathcal{S}$. The smallest such L_f is called the* Lipschitz constant *of f.*

Theorem 3 (Margin-to-Value Lipschitz Bound) *Let the margin function $\ell(s)$ and time discounted HJ value function $V^{\shield}(s)$ be Lipschitz continuous with constants L_ℓ and $L_{V^{\shield}}$, respectively. Let the discrete-time dynamics $f(s, a)$ be uniformly Lipschitz in s with constant L_f such that for a fixed discount factor $\gamma \in [0, 1)$, $\gamma L_f < 1$. Then the Lipschitz constant of $V^{\shield}(s)$ scales linearly in L_ℓ :*

$$L_{V^{\shield}} \leq L_\ell \cdot \max\left\{1, \frac{1-\gamma}{1-\gamma L_f}\right\}.$$

Proof: See Appendix 9.4.

Challenge 2: Distribution Mismatch Between Safety and Nominal Policies. Even with a smooth margin function, another practical issue arises when reinforcement learning (RL) is used to approximate the safety value function via (4). Prior works use actor-critic RL algorithms which jointly learn a safety policy (actor) $\pi^{\shield}_\nu(z)$ with parameters ν and critic (value function) $Q^{\shield}_\phi(z, a)$ with parameters ϕ by iteratively "rolling out" the current actor policy within a world model and fitting the critic (Sutton and Barto, 2018). Specifically, at each step within the world model, a transition (z, a, l, z') is saved in a replay buffer, $\mathcal{B}$, that stores a dataset of past latents, actions, and the margin function label $l = \ell_\mu(z)$. The critic is trained via supervised learning to the Bellman equation target:

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_{(z,a,l,z')\sim\mathcal{B}}\left[(Q^{\shield}_\phi(z, a) - y_{\text{target}})^2\right], \;\; y_{\text{target}} = (1-\gamma)l + \gamma \min\{l, Q^{\shield}_\phi(z', a')\}, \tag{7}$$

where $a' = \pi^{\shield}_\nu(z')$. Given the current estimated critic, the actor is then optimized by minimizing the loss $\mathcal{L}_{\text{actor}}(\nu) = \mathbb{E}_{z\sim\mathcal{B}}\left[-Q^{\shield}_\phi(z, \pi^{\shield}_\nu(z))\right]$. Here is where we can see our second challenge. Since the replay buffer $\mathcal{B}$ contains *only safe actions and latent transitions* obtained from $\pi^{\shield}$ (e.g., actions that may never grasp the bag), the critic only learns a high-quality estimate of how far the safety policy can get from the failure set. However, the critic rarely evaluates the safety outcomes of any other task-oriented action (e.g., dragging the bag along the table), since these state-action samples are rarely visited and put into the buffer $\mathcal{B}$. This off-the-shelf training process is at odds with how the critic will be used at deployment in CBF-style filtering, where we must evaluate the safety of task-relevant actions near π^{nom}—actions the critic has likely *never* encountered during training.

5. How to Train Your Latent Control Barrier Function

Motivated by our analysis in Section 4, we propose two algorithmic modifications to make latent HJ value functions informative latent CBFs. First, we improve the optimization landscape of (5) by reducing the Lipschitz constant of the margin function $\ell_\mu(z)$, and use knowledge of a nominal policy π^{nom} to diversify the state-action coverage during critic learning to improve the value estimates.

Smooth Margin Functions via Smooth Safety Discriminators. Our key idea for learning smooth margins *without dense supervision* is to draw inspiration from Wasserstein GANs (WGAN) (Arjovsky et al., 2017). This method learns a smooth discriminator (i.e., our margin function $\ell_\mu(z)$) that distinguishes between two classes of samples by regularizing its Lipschitz constant. We employ the objective function from WGAN-GP (Gulrajani et al., 2017):

$$\mathcal{L}_{\text{WGAN}}(\mu) = \lambda_{\text{w}} \cdot \Big(\mathbb{E}_{z^- \sim \mathcal{D}_{\text{fail}}}[\ell_\mu(z^-)] - \mathbb{E}_{z^+ \sim \mathcal{D}_{\text{safe}}}[\ell_\mu(z^+)] \Big) + \lambda_{\text{gp}} \cdot \mathbb{E}_{\hat{z} \sim \mathcal{D}_{\text{interp}}} \big[\, (\|\nabla_{\hat{z}} \ell_\mu(\hat{z})\|_2 - \beta)^2 \, \big], \quad (8)$$

($\ell_\mu(z)$ ranks safe > unsafe; Penalize norm of the gradient)

where $\hat{z} \sim \mathcal{D}_{\text{interp}}$ is obtained by sampling $z^+ \sim \mathcal{D}_{\text{safe}}$, and $z^- \sim \mathcal{D}_{\text{fail}}$ and linearly interpolating $\hat{z} = \eta z^- + (1-\eta) z^+$ such that $\eta \sim U(0,1)$. This objective encourages $\ell_\mu(z)$ to assign higher values to safe samples while the gradient penalty objective regularizes the Lipschitz constant toward $\beta \in \mathbb{R}_{>0}$ over straight lines connecting safe and unsafe latent states (Gulrajani et al., 2017).

Note, however, that the margin function which minimizes (8) is defined only up to an additive constant and will *not* result in a zero sublevel set which semantically corresponds to $\mathcal{F}$. We remedy this by including the sign loss $\mathcal{L}^{\delta}_{\text{sign}}$ from (6) with $\delta = 0$ since the WGAN loss already prevents a degenerate $\ell_\mu(z)$. This results in the overall loss function for training our latent margin function:

$$\mathcal{L} = \mathcal{L}_{\text{WGAN}} + \lambda_{\text{sign}} \cdot \mathcal{L}^{\delta=0}_{\text{sign}}. \quad (9)$$

where $\lambda_{\text{w}}, \lambda_{\text{gp}}$, and λ_{sign} are positive scalars that balance the contribution of the loss terms[5].

Mixing Safety & Nominal Policy Trajectories to Address Distribution Mismatch. We propose a simple modification to the actor-critical RL pipeline to address the distributional issues that impair the quality of $Q^{\shield}(z,a)$ when queried with actions that differ from $\pi^{\shield}(z)$. We populate the replay buffer $\mathcal{B}$ with *an equal proportion* of safety-oriented trajectories induced by the co-optimized actor $\pi^{\shield}$ and trajectories generated by a task-oriented policy[6] π^{nom}. At each timestep, we store transitions of the form $(z, a, l, z', a') \in \mathcal{B}$, where both a and a' are sampled from the same policy (either $\pi^{\shield}$ or π^{nom}) to fit the critic[7]. This enables the critic to learn safety estimates that remain accurate for task-relevant actions likely to be encountered during deployment.

Sampling-based Safety Filtering via LatentCBFs. Unlike the continuous-time CBF with control-affine dynamics, the discrete-time CBF optimization problem in (5) does *not* admit a quadratic program and is in general nonconvex, making it challenging to optimize efficiently in high-dimensional action spaces. To make (5) tractable we use zeroth-order optimization and sample the space of actions $\mathcal{A}$ using a mixture of π^{nom} and $\pi^{\shield}$, from which we create a subset that satisfy the CBF constraint, $\mathcal{A}_{\text{CBF-Safe}}$. The filter returns the most similar task-driven action from $\mathcal{A}_{\text{CBF-Safe}}$. Evaluating the objective and constraints of (5) can be accelerated with the parallelization capabilities of modern hardware (e.g., the entire process takes 10 ms for 7.6k samples for our 7DOF manipulator). If no valid sample exists, we default to $\pi^{\shield}$. Additional details can be found in Appendix 9.6.

5. This function is trained in isolation without a separate generator network, bypassing unstable adversarial training.
6. We assume that the task-oriented policy is given to us and does not change over the course of RL training.
7. Unlike approaches that learn a CBF via policy evaluation (So et al., 2024), our approach still performs full deep RL.

6. Simulation & Hardware Experiments

We conduct simulation and hardware experiments to evaluate each component of **LatentCBF** for safety filtering. All experiments assume access to a labeled offline dataset $\mathcal{D}$, a latent world model (e.g., $\mathcal{E}(o), f_z(z,a)$) trained on this data, and a nominal diffusion policy $\pi^{\text{nom}}(o)$ that performs task-oriented actions from raw images. π^{nom} is trained with both safe and unsafe demonstrations to test how effectively our safety filters guide an erroneous visuomotor policy. Details in Appendix 9.2.

6.1. Simulation: Vision-Based Dubins' Car Navigation

Let a Dubins' car system have state $s = (x, y, \theta)$ given by its position and steering angle θ and continuous-time dynamics $\dot{s} = [\cos(\theta), \sin(\theta), a]$ where $a \in [-2, 2]$ is turn rate. The dynamics are discretized using RK4 and timestep $dt = 0.1$. The x and y positions are bounded in $[-1.5, 1.5]$. The true failure set $\mathcal{F}$ is two circles centered at $(0.25, 0.65)$ and $(0.25, -0.65)$, with radius $r = 0.5$. Observations $o \in \mathcal{O}$ are RGB images and the angle θ (see Figure 2). We label ground truth failures using privileged simulator information. We train two margin functions and respective value functions using our proposed method (**GP**) and a baseline (**NoGP**) from (Nakamura et al., 2025).

Result: Gradient Penalties Yield Smoother Margin Functions Without Extra Labels. We report the F1 score and smoothness of each margin function evaluated over 100 trajectories generated by π^{nom} without filtering. Smoothness is measured as the maximum single-timestep change in $\ell(z)$ within a trajectory, i.e., $\max_t |\ell_\mu(z_{t+1}) - \ell_\mu(z_t)|$. Although both methods use the same dataset, **GP** reduces the largest margin function gradients by 86% from 1.2 ± 0.76 to 0.17 ± 0.065 while maintaining a similar F1 score to **NoGP** (**GP**: 0.991 vs. **NoGP**: 0.981).

Safety Filter	NoGP		GP		None
	Avg. $\lvert\Delta a\rvert$	Safety Rate	Avg. $\lvert\Delta a\rvert$	Safety Rate	Safety Rate
LR	1.5 ± 1.1	97%	2.0 ± 1.3	100%	41%
CBF	1.3 ± 1.0	97%	1.1 ± 0.9	100%	

Table 1: **Simulation: On the Quality of Latent Safety Filters.** CBFs trained with gradient-penalties yield smoother interventions, decreasing the average override magnitude relative to LR by 45% compared to only 13% for the CBF without gradient penalties.

Result: LatentCBFs Perturb the Nominal Policy Less While Staying Safe. We compare two CBF filters against least-restrictive (LR) filtering: $a^\star = \mathbb{1}_{\{Q^{\textcircled{\scriptsize V}}(z,\pi^{\text{nom}})<\epsilon\}} \cdot \pi^{\textcircled{\scriptsize V}} + \mathbb{1}_{\{Q^{\textcircled{\scriptsize V}}(z,\pi^{\text{nom}})\geq\epsilon\}} \cdot \pi^{\text{nom}}$ for both **GP** and **NoGP** value functions with $\epsilon = 0.2$. Whenever the filter modifies the action, we record the action difference with respect to the nominal policy $|\Delta a| = |a^\star - \pi^{\text{nom}}|$, referred to below as the action override magnitude. We also measure the safety rate via $\%$ of trajectories without a collision between the car and obstacles. For both methods, we ablate the effect of the gradient penalty on $\ell_\mu(z)$ and report the results across 100 trajectories in Table 1. The nominal policy only achieves a 41% safety rate. We find that CBF-style filtering decreases the average override magnitude no matter how the margin function is shaped; however, the CBF trained with **GP** yields a 45% decrease in action override magnitude relative to the corresponding LR filter. The **NoGP** CBF only reduces action magnitude by 13% relative to the LR filter. Even with the reduced action override magnitude, the CBF methods maintain high safety rates matching their LR counterpart.

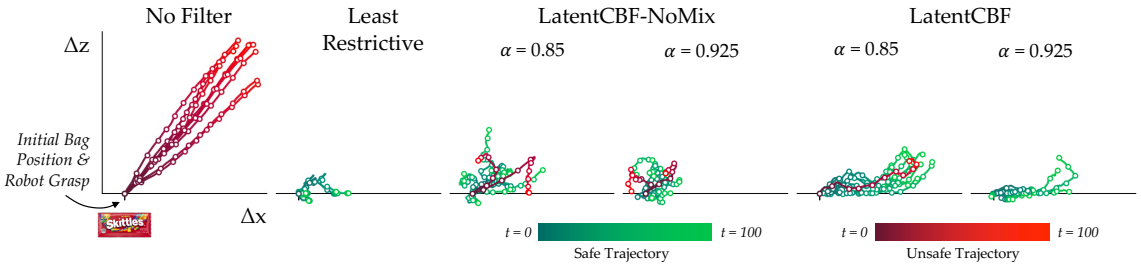

Figure 3: **Hardware: End-effector Trajectories in the X-Z Plane.** LatentCBF maintains control authority and safety compared to LR and unfiltered baselines. Without fitting the critic as in Sec. 5, LatentCBF-NoMix permits erratic actions and spills in 20% of trials.

6.2. Hardware: Safety Filtering a Visuomotor Manipulation Policy

We next scale our method to a manipulation task from Nakamura et al. (2025) where a robot arm must pick up an opened bag of small round candies (Skittles) without spilling them. The robot has a 7-D action space $a \in \mathcal{A}$ consisting of end-effector (EEF) translation and axis-angle rotation, along with binary gripper actions. The observation space $o \in \mathcal{O}$ consists of the EEF pose and RGB images from a wrist-mounted and 3rd person camera (see Figure 1). We use DINO-WM, a vision transformer that predicts DINOv3 embeddings (Zhou et al., 2025) as our world model.

Result: LatentCBF Needs Data from π^{nom} and $\pi^{\shield}$ to Effectively Filter High-D Actions. We train two value functions: using our RL strategy from Section 5 (**LatentCBF**) and a baseline that does not learn from task-oriented trajectories (**LatentCBF-NoMix**), both with gradient penalties. The robot is initialized with its end-effector grasping the closed end of a bag resting on the table and executes an *unsafe* open-loop diagonal lift. All CBF filters are deployed with $\alpha \in \{0.85, 0.925\}$ and compared against the unfiltered sequence (**No Filter**) and a least-restrictive safety filter (**Least Restrictive**) using the same value function as **LatentCBF** with $\epsilon = 0.05$. Figure 3 shows 10 end-effector trajectories in the X–Z plane for each method. **No Filter** always lifts and spills the bag. The **Least Restrictive** filter offers limited control: it lifts the bag the least ($+\Delta z$) and allows minimal translation along the table ($+\Delta x$). In contrast, **LatentCBF** projects away unsafe action components and slides the bag along the table. Without mixing trajectories from π^{nom} and $\pi^{\shield}$ during RL training, **LatentCBF-NoMix** produces erratic actions and more safety violations than **LatentCBF**.

Result: LatentCBF Removes Unsafe Visuomotor Policy Modes Without Hindering the Task. Finally, we evaluate our method when deployed in-the-loop with a visuomotor nominal policy. Here, $\pi^{\text{nom}}(a \mid o)$ is the previously trained diffusion policy, which has both safe and unsafe interaction modes. We compare filtering this policy with three methods: the proposed **LatentCBF**, a baseline **LatentCBF-NoGP** (trained *without* the gradient penalty for the margin function), and the **Least Restrictive** filter (the switching filter from the previous subsection). For each method, we run 20 trials from three initial end-effector and bag configurations designed to elicit different nominal policy behaviors: IC1-consistently *unsafe* actions, IC2-consistently *safe* actions, and IC3-*multimodal* grasping behavior. We measure the percentage of spills (Fail), safe trials where the robot does *not* lift the bag (Stall), and safely lifting without a spill (Success).

Table 2 shows that none of the filters are overly conservative, preserving π^{nom}'s success rate in IC2. In IC1 and IC3, **LatentCBF** reduces failures while improving task success, whereas baseline methods reduce failures at the cost of increased stalling. In IC3, **LatentCBF** occasionally drives the policy out-of-distribution (OOD) of the imitation-based visuomotor policy, which lowers task suc-

Safety Filter	$\ell_\mu(z)$	Init Cond 1 (%)			Init Cond 2 (%)			Init Cond 3 (%)			Aggregate (%)		
		Fail	Stall	Success	Fail	Stall	Success	Fail	Stall	Success	Fail	Stall	Success
None	–	100	0	0	0	0	100	85	0	15	62	0	38
LR	GP	0	90	10	0	0	100	0	95	5	0	62	38
CBF	No GP	0	100	0	0	0	100	0	85	15	0	62	38
	GP	0	0	100	0	0	100	0	60	40	0	20	80

Table 2: **Hardware: Safely Guiding a Diffusion Policy.** Results are averaged over three initial conditions with 20 trials each. All safety filters prevent spilling, but **LR** and **LatentCBF-NoGP** do so at the cost of task performance, whereas **LatentCBF** more consistently steers the policy toward both safe and successful executions.

# Samples	Filtering Speed (ms) as a Function of # Samples				
	10	30	50	3,800	7,600
Model-based	40.70 ± 0.76	114.6 ± 2.6	`out-of-mem`	`out-of-mem`	`out-of-mem`
Model-free	0.33 ± 0.12	0.32 ± 0.08	1.85 ± 0.32	3.96 ± 0.19	9.60 ± 0.57

Table 3: **Hardware: LatentCBF Filtering Speed.** Model-based filtering is infeasible for over 50 actions (out of memory). Model-free can evaluate thousands of actions in 10 ms.

cess despite preventing failures. Overall, **LatentCBF** more effectively guides the diffusion policy toward safe and successful lifts (80% vs. 38%).

Result: Model-Free Filtering Scales to Thousands of Action Samples. Since the action search space grows exponentially with dimension, our method must scale to many samples for high-dimensional manipulation tasks. We consider two filtering schemes: (1) *model-based*, which predicts $z' = f_z(z, a)$ and evaluates $Q^{\blacklozenge}(z', \pi^{\blacklozenge}(z'))$, and (2) *model-free* (ours), which directly evaluates $Q^{\blacklozenge}(z, a)$. In Table 3 we show that even with parallelization, model-based filtering is bottlenecked by inference through the large vision-transformer-based DINO-WM (with 19M parameters), preventing evaluation of $50+$ action samples on a NVIDIA A6000 ADA GPU with 48GB of VRAM. In contrast, model-free filtering scales to thousands of samples and takes about 10 ms. Exact details of our sampling and filtering procedure are provided in Appendix 9.7.

7. Conclusion & Limitations

In this paper, we identify two key limitations in existing latent safety filters: non-smooth value functions from classifier-based margins and inaccurate estimates due to training–deployment distribution mismatch. Our **LatentCBF** addresses both through Lipschitz-constrained margin learning inspired by Wasserstein GANs and mixed-policy value-function training. **LatentCBF** enables optimization-based safety filtering from high-dimensional observations, and experiments show that both contributions are critical for achieving safe yet minimally invasive filtering.

Limitations. While **LatentCBF** scales up safety filtering for visuomotor policies, it is not without limitations. Its reliance on learned latent representations, dynamics, and RL approximations precludes formal safety guarantees, which is an future important direction (Lutkus et al., 2025). Successful filtering also depends on the base policy; if the CBF pushes the system into OOD states, the nominal policy may be unable to complete the task (Römer et al., 2026). Future work should integrate uncertainty estimation or OOD detection (Seo et al., 2025; Sun and Song, 2025).

References

Ayush Agrawal and Koushil Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *Robotics: Science and Systems*, volume 13, pages 1–10. Cambridge, MA, USA, 2017.

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, August 2017. ISSN 1558-2523. doi: 10.1109/TAC.2016.2638961. URL https://ieeexplore.ieee.org/document/7782377/?arnumber=7782377. Conference Name: IEEE Transactions on Automatic Control.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. Ieee, 2019.

Mehul Anand and Shishir Kolathaya. Safety certification in the latent space using control barrier functions and world models, 2025. URL https://arxiv.org/abs/2507.13871.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

Somil Bansal and Claire Tomlin. DeepReach: A Deep Learning Approach to High-Dimensional Reachability, November 2020. URL http://arxiv.org/abs/2011.02082. arXiv:2011.02082 [cs].

Osbert Bastani. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *American Control Conference (ACC)*, pages 3488–3494. IEEE, 2021.

Richard Bellman. On the theory of dynamic programming. *Proceedings of the national Academy of Sciences*, 38(8):716–719, 1952.

Bingqing Chen, Jonathan Francis, Jean Oh, Eric Nyberg, and Sylvia L Herbert. Safe autonomous racing via approximate reachability on ego-vision. *arXiv preprint arXiv:2110.07699*, 2021.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

Jason J Choi, Donggun Lee, Koushil Sreenath, Claire J Tomlin, and Sylvia L Herbert. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6814–6821. IEEE, 2021.

Wenceslao Shaw Cortez, Xiao Tan, and Dimos V Dimarogonas. A robust, multiple control barrier function framework for input constrained systems. *IEEE Control Systems Letters*, 6:1742–1747, 2021.

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. doi: 10.1109/TRO.2022.3232542.

Jaime F. Fisac, Neil F. Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J. Tomlin. Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556, May 2019. doi: 10.1109/ICRA.2019.8794107. URL https://ieeexplore.ieee.org/document/8794107. ISSN: 2577-087X.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

Chloe He, Borja G. Leon, and Francesco Belardinelli. Do androids dream of electric fences? safety-aware reinforcement learning with latent shielding, 2021.

Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile but safe: Learning collision-free high-speed legged locomotion. *Robotics: Science and Systems*, 2024.

Kai-Chieh Hsu, Allen Z Ren, Duy P Nguyen, Anirudha Majumdar, and Jaime F Fisac. Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees. *Artificial Intelligence*, 314:103811, 2023.

Kai-Chieh Hsu, Duy Phuong Nguyen, and Jaime Fernández Fisac. ISAACS: Iterative Soft Adversarial Actor-Critic for Safety, June 2024. URL http://arxiv.org/abs/2212.03228. arXiv:2212.03228 [cs].

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Matthew Kim, William Sharpless, Hyun Joe Jeong, Sander Tonkens, Somil Bansal, and Sylvia Herbert. Reachability barrier networks: Learning hamilton-jacobi solutions for smooth and flexible control barrier functions, 2025. URL https://arxiv.org/abs/2505.11755.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Somnath Sendhil Kumar, Qin Lin, and John Dolan. LatentCBF: A control barrier function in latent space for safe control, 2024. URL https://openreview.net/forum?id=30L0rr9W8A.

Albert Lin, Shuang Peng, and Somil Bansal. One filter to deploy them all: Robust safety for quadrupedal navigation in unknown environments. *arXiv preprint arXiv:2412.09989*, 2024.

Paul Lutkus, Kaiyuan Wang, Lars Lindemann, and Stephen Tu. Latent representations for control design with provable stability and safety guarantees. *arXiv preprint arXiv:2505.23210*, 2025.

I.M. Mitchell, A.M. Bayen, and C.J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7): 947–957, July 2005. ISSN 1558-2523. doi: 10.1109/TAC.2005.851439. URL https://ieeexplore.ieee.org/document/1463302/?arnumber=1463302. Conference Name: IEEE Transactions on Automatic Control.

Naoki Morihira. dreamerv3-torch: Implementation of dreamer v3 in pytorch. https://github.com/NM512/dreamerv3-torch, 2025. Accessed: 2025-01-25.

Kensuke Nakamura, Lasse Peters, and Andrea Bajcsy. Generalizing Safety Beyond Collision-Avoidance via Latent-Space Reachability Analysis, February 2025. URL http://arxiv.org/abs/2502.00935. arXiv:2502.00935 [cs].

Donggeon David Oh, Justin Lidard, Haimin Hu, Himani Sinhmar, Elle Lazarski, Deepak Gopinath, Emily S Sumner, Jonathan A DeCastro, Guy Rosman, Naomi Ehrich Leonard, et al. Safety with agency: Human-centered safety filter with application to ai-assisted motorsports. *Robotics: Science and Systems*, 2025.

Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In *International conference on tools and algorithms for the construction and analysis of systems*, pages 370–388. Springer, 2021.

Ralf Römer, Julian Balletshofer, Jakob Thumm, Marco Pavone, Angela P. Schoellig, and Matthias Althoff. From demonstrations to safe deployment: Path-consistent safety filtering for diffusion policies, 2026. URL https://arxiv.org/abs/2511.06385.

Ed Schmerling. hj_reachability: Hamilton-jacobi reachability analysis in jax, 2024. URL https://github.com/StanfordASL/hj_reachability. GitHub repository.

Junwon Seo, Kensuke Nakamura, and Andrea Bajcsy. Uncertainty-aware latent safety filters for avoiding out-of-distribution failures. *Conference on Robot Learning*, 2025.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. URL https://arxiv.org/abs/2508.10104.

Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11532–11539. IEEE, 2024.

Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *International Conference on Learning Representations*, 2023.

Mohit Srinivasan, Amogh Dabholkar, Samuel Coogan, and Patricio A Vela. Synthesis of control barrier functions using a supervised machine learning approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7139–7145. Ieee, 2020.

Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution, 2025. URL https://arxiv.org/abs/2508.05941.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

Mukun Tong, Charles Dawson, and Chuchu Fan. Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields. *International Conference on Robotics and Automation*, 2023.

Sander Tonkens and Sylvia Herbert. Refining control barrier functions through hamilton-jacobi reachability. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

Kim P. Wabersich, Andrew J. Taylor, Jason J. Choi, Koushil Sreenath, Claire J. Tomlin, Aaron D. Ames, and Melanie N. Zeilinger. Data-Driven Safety Filters: Hamilton-Jacobi Reachability, Control Barrier Functions, and Predictive Methods for Uncertain Systems. *IEEE Control Systems Magazine*, 43(5):137–177, October 2023. ISSN 1941-000X. doi: 10.1109/MCS.2023.3291885. URL https://ieeexplore.ieee.org/document/10266799/?arnumber=10266799. Conference Name: IEEE Control Systems Magazine.

Albert Wilcox, Ashwin Balakrishna, Brijen Thananjeyan, Joseph E. Gonzalez, and Ken Goldberg. Ls3: Latent space safe sets for long-horizon visuomotor control of sparse reward iterative tasks. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 959–969. PMLR, 08–11 Nov 2022.

Sinong Simon Zhan, Yixuan Wang, Qingyuan Wu, Ruochen Jiao, Chao Huang, and Qi Zhu. State-wise safe reinforcement learning with pixel observations. In *Learning for Dynamics and Control Conference (L4DC)*, 2024.

Hengjun Zhao, Xia Zeng, Taolue Chen, and Zhiming Liu. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, pages 1–11, 2020.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning, February 2025. URL http://arxiv.org/abs/2411.04983. arXiv:2411.04983 [cs].

8. Acknowledgments

KN would like to acknowledge Oswin So for discussions on policy neural CBFs and inspiring the title of this work. He also thanks Gokul Swamy for helpful discussions surrounding reinforcement learning theory.

9. Appendix

9.1. Extended Related Work

Scaling Safety Filters: From High-D States to Observations. Motivated by the difficulty of synthesizing control barrier functions by hand and numerically computing solutions to HJ reachability problems, an increasingly popular body of work involves scaling up safety filter synthesis with neural approximations. A plethora of methods for this exist, see (Wabersich et al., 2023) for a comprehensive survey. Neural CBF methods (Peruffo et al., 2021; Zhao et al., 2020; Srinivasan et al., 2020) directly optimize the CBF constraint over a dataset of safe and unsafe samples which typically suffer from handling input constraints in practice (Dawson et al., 2023). Alternative methods compute safe control-invariant sets with input constraints by approximating solutions to HJ reachability problems via self-supervised learning (Bansal and Tomlin, 2020; Kim et al., 2025) or reinforcement learning (Fisac et al., 2019; Hsu et al., 2024). Similar to our work, is (So et al., 2024) which learn *policy neural control barrier functions*; value functions obtained via policy evaluation with respect to a Hamilton-Jacobi Bellman equation. This method explicitly constructs a CBF around a nominal policy, but parametrizes the CBF as a purely state-value function and thus requires a forward simulation of the dynamics to evaluate the CBF constraint. Our work instead parameterizes the CBF as a state-action value function as in (Oh et al., 2025) which eliminates the need to forward simulation but adds the complexity of learning accurate state-action safety estimates. In general, all aforementioned works require access to explicit knowledge of the system state, dynamics, and failure constraints; an assumption relaxed in this work. Prior work synthesizing latent control barrier functions follow traditional neural CBF approaches (Kumar et al., 2024; Anand and Kolathaya, 2025), requiring labels corresponding to knowledge of a *safe control invariant set* which is assumed to be available but in practice difficult to find (So et al., 2024). In contrast to our work which only requires labels corresponding to the *failure set* and automatically synthesizes the CBF through reachability analysis.

Observations: Our work is not the first to incorporate observation as an input to safety filters: prior work have used LIDAR as an input to the safety value function (Lin et al., 2024; He et al., 2024) which can be trained in simulation and effectively deployed to real. Prior work has also explored simulated RGB data and conducted sim2real for least-restrictive safety filters that can operate directly from RGB (Hsu et al., 2023; Chen et al., 2021). While our work trains a latent world model for forward simulating the environment, (Tong et al., 2023) trains a NERF representation raw perception data for forward simulating the control-barrier function condition. However these representations are more poorly suited for constraints beyond collision avoidance where the robot interacts with the environment via manipulation.

Safe Latent-space Control. While our work primarily focuses on latent safety filtering following Nakamura et al. (2025), safe latent-space control does not necessarily have to be implemented as a filter. Prior work has explored using latent-space constraints to learn a task-oriented policy that

is safe *during* RL training Zhan et al. (2024); He et al. (2021) or model-based planning in the latent-space Wilcox et al. (2022) instead of learning a policy via RL. In contrast, this work assumes a nominal visuomotor policy is given a priori and is fixed. Our contribution in this work is the synthesis of a latent control barrier function that can safeguard this base policy even when operating directly from high-dimensional observations.

Critic Learning in Reinforcement Learning. The primary concern in reinforcement learning is obtaining a performant *policy*. Thus, the quality of the critic in actor-critic methods typically only matters insofar as it provides an effective source of supervision for optimizing this policy. While out-of-distribution actions are known to produce poor value estimates (the problem of overestimation bias), existing approaches to this problem instead introduce *underestimation* (Fujimoto et al., 2018) or *conservative* critic updates (Kumar et al., 2020) that maintain high-performance without addressing the inaccuracies of the critic itself. This is a non-issue in standard sum-of-reward RL that cares only about a highly performant policy but is unfavorable in our setting where we wish to use the critic for safety filtering. We draw inspiration from Hybrid RL (Song et al., 2023) which enables learning a critic from an offline dataset consisting of trajectories of varying quality, and online interaction. In our setting, we fit our critic using a mixture of both the safety fallback actions from $\pi^{\blacksquare}$ and nominal task-oriented actions from π^{nom}.

9.2. Implementation Details

Simulation: World Model and Nominal Policy. For our simulated Dubins' car experiment, we use an open source implementation of the recurrent state space model (Hafner et al., 2025) in PyTorch Morihira (2025) and a diffusion policy following the implementation from Chi et al. (2024). We list all hyperparameters in Table 4 and 5. These models take as input a (128x128x3) RGB image of the environment, and the steering angle θ of the system. For learning the margin function, we parameterize it as an MLP with two hidden layers [512, 512] and SiLU activations. We apply the $\texttt{stopgrad}[z]$ to the latent states when optimizing the margin function with (6) and (9). For the gradient penalty, we use parameters $\lambda_{\text{w}} = 0.1$, $\lambda_{\text{sign}} = 1$, $\lambda_{\text{GP}} = 10$ with the gradient threshold $\beta = 0.1$. These loss weights are chosen to prioritize $\mathcal{L}_{\text{sign}}$ and $\mathcal{L}_{\text{gp}}$. We choose a small weight for $\mathcal{L}_{\text{w}}$ since we observe that the gradient regularization can be lost when $\mathcal{L}_{\text{w}}$ is too high. For the baseline margin function without the gradient penalty, we only optimize $\mathcal{L}^{\delta}_{\text{sign}}$ with $\delta = 0.75$.

We begin by training the diffusion policy: we collected 200 demonstrations using an MPPI controller with stochastic collision avoidance costs reaching a randomized goal. The initial conditions are concentrated between $x \in [-1.5, -1], y \in [-1, 1], \theta \in [-\frac{\pi}{3}, \frac{\pi}{3}]$ and goal location is fixed at $x = 1.3$ and the y position is sampled from $[-0.6, 0.6]$ This provides a dataset of task-oriented trajectories that are both safe and unsafe. To collect our world-model training data, we collect 4000 trajectories of randomly executed actions by sampling initial conditions uniformly over the state space and actions uniformly from the action space. We also collect 3800 trajectories of rolling out intermediate DP checkpoints at OOD initial conditions following Sun and Song (2025).

Hardware: World Model and Nominal Policy. We train a nominal visuomotor diffusion policy $\pi^{\text{nom}}(a \mid o)$ (Chi et al., 2024) using 215 teleoperated demonstrations and same hyperparameters as Table 5. In order to test the effectiveness of our safety filter in guiding an erroneous visuomotor policy, we intentionally include demonstrations of both *unsafe* grasps from the closed-end of the bag (where the robot may spill) and *safe* grasps from the opened end of the bag that prevents spilling. For

HYPERPARAMETER	VALUE
RGB IMAGE DIMENSION	[128, 128, 3]
IR IMAGE DIMENSION	[128, 128, 1]
ACTION DIMENSION	1
STOCHASTIC LATENT	Gaussian
LATENT DIM (DETERMINISTIC)	512
LATENT DIM (STOCHASTIC)	32
ACTIVATION FUNCTION	SiLU
ENCODER CNN DEPTH	32
ENCODER MLP LAYERS	5
FAILURE PROJECTOR LAYERS	2
BATCH SIZE	32
BATCH LENGTH	16
OPTIMIZER	Adam
LEARNING RATE	1e-4
ITERATIONS	40000

Table 4: Dubins' car RSSM Hyperparameters

Hyperparameter	**Values**
State Normalization	Yes
Action Normalization	Yes
Action Chunk	16
Image Chunk	2
Image Size	256
Batch size	100
Training Iterations	500000
Learning Rate	1e-4
Learning Rate Schedule	Cosine
Optimizer	AdamW

Table 5: Hyperparameters for Diffusion Policy

our hardware experiments, we use DINO-WM Zhou et al. (2025), a vision-transformer-based world model that predict future DINO embeddings conditioned on proprioception, robot action, and a history of $H = 3$ timesteps. For this world model, we use frozen pre-trained `DINOv3-vits16plus` backbone (Siméoni et al., 2025) as our vision encoder. This compresses a $[256, 256, 3]$ RGB image into a $[256, 384]$ tensor of dense patch token, capturing fine-grained detail in both the wrist and camera views. We train on $|\mathcal{D}| = 735$ trajectories consisting of nominal policy rollouts from π^{nom}, random actions, and exploratory teleoperated trajectories following (Sun and Song, 2025).

We concatenate the patch tokens for both camera views, 10-d action embedding, and 8-dim state (eef position, quaternion, gripper width) along the feature dimension. This leads to a $[256, 786]$ latent representation of the state at each timestep. When training the world model, we normalize the dataset of actions to $\mathcal{N}(0, 1)$ along each non-gripper action dimension. For ease of computation, we

pre-compute the DINOv3 embeddings offline without image augmentations. This allows us bypass DINOv3 forward passes at each training iteration. Training the world model for 100000 iterations took about 12 hours on an Nvidia A6000 ADA GPU. We include all hyperparameters in Table 6.

To train the margin function and value function, we average-pool the latent state (average over patch tokens) before passing it as input to their respective MLPs. We parameterize $\ell(z)$ as a two-layer MLP with LayerNorm and ReLU activations. For the gradient penalty, we use parameters $\lambda_{\text{w}} = 0.2$, $\lambda_{\text{sign}} = 100$, $\lambda_{\text{GP}} = 10$ with the gradient threshold $\beta = 0.02$. These loss weights are chosen to prioritize $\mathcal{L}_{\text{sign}}$ and $\mathcal{L}_{\text{gp}}$. For the baseline margin function without the gradient penalty, we once again we only optimize $\mathcal{L}^{\delta}_{\text{sign}}$ with $\delta = 0.75$.

Hyperparameter	**Values**
Image size	256
DINOv3 patch size	$(16 \times 16, 384)$
Optimizer	AdamW
Predictor lr	5e-5
Decoder lr	3e-4
Action Encoder lr	5e-4
Action emb dim	10
Proprioception emb dim	10
Batch size	16
Batch len	4
Training iterations	100000
ViT depth	6
ViT attention heads	16
ViT MLP dim	2048

Table 6: Hyperparameters for DINO-WM

9.3. Latent Space Reinforcement Learning

We list all parameters for our actor and critic networks in Table 7. We parameterize our safety policy $\pi^{\shield}$ as a deterministic policy and use a single critic. During RL training, we assume access to a policy π^{nom} that takes as input image observations. In practice, we use a diffusion policy that outputs an H-timestep action chunk $a_{t:t+H}$ for H=16 steps.

To perform reinforcement learning, one has to "reset" the environment at the beginning of each trajectory. Due to the high-dimensionality of the latent state, this reset must be done carefully to ensure resetting onto the data manifold. In practice, this is done by encoding a randomly sampled observation from an offline dataset $o \sim \mathcal{D}$. We generate $a_{t:t+H} \sim \pi^{\text{nom}}(o)$ with 50% probability and execute the first T timesteps before terminating the episode. We do not explicitly require this form of nominal policy, in principle one could repeat a single action $a \sim \pi^{\text{nom}}(o)$ for T steps for other parameterizations of the nominal policy. Otherwise, we execute $\pi^{\shield}$ the entire T timestep episode. In practice, we set $T = 8$ to prevent learning on world model imaginations that degrade for long time horizons. The actor is constrained to output actions within $[-1, 1]$ per dimension, which on hardware corresponds approximately to ± 1 standard deviation of the dataset actions, given that

the world model operates under normalized action scaling. When storing the (z, a, l, z, a') tuple into the dataset we use $l = \tanh(\ell(z))$ to ensure boundedness.

HYPERPARAMETER	VALUE
ACTOR ARCHITECTURE	[512, 512, 512]
CRITIC ARCHITECTURE	[512, 512, 512]
NORMALIZATION	LayerNorm
ACTIVATION	ReLU
DISCOUNT FACTOR γ	0.995
LEARNING RATE (CRITIC)	3e-4
LEARNING RATE (ACTOR)	1e-4
OPTIMIZER	Adam
NUMBER OF ITERATIONS	120000
REPLAY BUFFER SIZE	100000
BATCH SIZE	512
MAX IMAGINATION STEPS	8

Table 7: RL hyperparameters.

9.4. Proof of Margin-to-Value Lipschitz Bound

For clarity, we restate the statement we wish to prove.

Theorem 4 (Margin-to-Value Lipschitz Bound) *Let the margin function $\ell(s)$ and time discounted value function $V^{\textcircled{\scriptsize 0}}(s)$ be Lipschitz continuous with constants L_ℓ and $L_{V^{\textcircled{\scriptsize 0}}}$, respectively. Let the discrete-time dynamics $f(s, a)$ be uniformly Lipschitz in s with constant L_f such that for a fixed discount factor $\gamma \in [0, 1)$, $\gamma L_f < 1$. Then the Lipschitz constant of $V(s)$ scales linearly in L_ℓ :*

$$L_{V^{\textcircled{\scriptsize 0}}} \leq L_\ell \cdot \max\left\{1, \frac{1-\gamma}{1-\gamma L_f}\right\}.$$

Proof: We seek an upper bound on $L_{V^{\textcircled{\scriptsize 0}}}$ which is defined as the smallest constant such that $\forall s, \tilde{s} \in \mathcal{S}$, $|V^{\textcircled{\scriptsize 0}}(s) - V^{\textcircled{\scriptsize 0}}(\tilde{s})| \leq L_{V^{\textcircled{\scriptsize 0}}}|s - \tilde{s}|$. We begin by expanding $V^{\textcircled{\scriptsize 0}}(s) - V^{\textcircled{\scriptsize 0}}(\tilde{s})$.

$$V^{\textcircled{\scriptsize 0}}(s) - V^{\textcircled{\scriptsize 0}}(\tilde{s}) = (1-\gamma)(\ell(s) - \ell(\tilde{s})) + \gamma\big[\min\left\{\ell(s),\ \max_{a\in\mathcal{A}} V^{\textcircled{\scriptsize 0}}(f(s,a))\right\} - \min\left\{\ell(\tilde{s}),\ \max_{\tilde{a}\in\mathcal{A}} V^{\textcircled{\scriptsize 0}}(f(\tilde{s},\tilde{a}))\right\}\big]$$

Case 1. $\ell(\tilde{s})$ minimizes the second term.

$$V^{\textcircled{\scriptsize 0}}(s) - V^{\textcircled{\scriptsize 0}}(\tilde{s}) \leq (1-\gamma)(\ell(s) - \ell(\tilde{s})) + \gamma(\ell(s) - \ell(\tilde{s}))$$

since $\min\left\{\ell(s),\ \max_{a\in\mathcal{A}} V^{\textcircled{\scriptsize 0}}(f(s,a))\right\} - \ell(\tilde{s}) \leq \ell(s) - \ell(\tilde{s})$.

Case 2. $\max_{\tilde{a}\in\mathcal{A}} V^{\textcolor{black}{\blacklozenge}}(f(\tilde{s},\tilde{a}))$ minimizes the second term.

$$V^{\textcolor{black}{\blacklozenge}}(s) - V^{\textcolor{black}{\blacklozenge}}(\tilde{s}) \leq (1-\gamma)(\ell(s)-\ell(\tilde{s})) + \gamma(V^{\textcolor{black}{\blacklozenge}}(f(s,a')) - V^{\textcolor{black}{\blacklozenge}}(f(\tilde{s},a')))$$

where $a' = \text{argmax}_{a\in\mathcal{A}} V^{\textcolor{black}{\blacklozenge}}(f(s,a))$ and we use similar logic to Case 1 to upperbound the difference.

Note that $V^{\textcolor{black}{\blacklozenge}}(\tilde{s}) - V^{\textcolor{black}{\blacklozenge}}(s)$ yields symmetric bounds. This implies that we are left with two cases:

$$|V^{\textcolor{black}{\blacklozenge}}(s) - V^{\textcolor{black}{\blacklozenge}}(\tilde{s})| \leq |\ell(s)-\ell(\tilde{s})| \leq L_\ell |s-s'|$$

and

$$\begin{aligned}|V^{\textcolor{black}{\blacklozenge}}(s) - V^{\textcolor{black}{\blacklozenge}}(\tilde{s})| \leq& (1-\gamma)L_\ell|s-\tilde{s}| + \gamma L_{V^{\textcolor{black}{\blacklozenge}}}|f(s,a')-f(\tilde{s},a')| \\ \leq& (1-\gamma)L_\ell|s-\tilde{s}| + \gamma L_{V^{\textcolor{black}{\blacklozenge}}}L_f|s-\tilde{s}| = ((1-\gamma)L_\ell + \gamma L_{V^{\textcolor{black}{\blacklozenge}}}L_f)|s-\tilde{s}|\end{aligned}$$

Since by definition $L_{V^{\textcolor{black}{\blacklozenge}}}$ is the smallest constant L such that $|V^{\textcolor{black}{\blacklozenge}}(s) - V^{\textcolor{black}{\blacklozenge}}(\tilde{s})| \leq L|s-\tilde{s}|$, rearranging results in

$$L_{V^{\textcolor{black}{\blacklozenge}}} \leq \max\{L_\ell, \frac{1-\gamma}{1-\gamma L_f} L_\ell\}.$$

In both cases, $L_{V^{\textcolor{black}{\blacklozenge}}}$ varies linearly in L_ℓ. While the assumptions made have no guarantee to hold in latent spaces induced by a world model (whose continuity properties are poorly understood in general), or may lead to weak bounds if the Lipschitz constant of the dynamics is high (e.g., due to contacts), this motivates restricting the Lipschitz constant[8] of $\ell(z)$ to improve the gradient landscape of the value function.

9.5. Additional Simulation results

We report the confusion matrix (where TP corresponds to classified as safe and ground truth is safe, etc.) for margin function accuracy, and our smoothness metric in Table 8. The gradient penalty drastically reduces the largest $\Delta_t \ell(z_t)$ without significantly degrading classification accuracy.

$\ell_\mu(z)$	TP (%)	TN (%)	FP (%)	FN (%)	$\max_t \Delta_t \ell(z_t)$
NoGP	84	13	0.63	2.5	1.2 ± 0.76
GP	86	13	1.0	0.60	0.17 ± 0.065

Table 8: **Simulation: Accuracy & Smoothness of Latent $\ell_\mu(z)$.** In the Dubins' car simulation, we compare the latent failure set encoded via $\mathcal{F}_z = \{z : \ell_\mu(z) < 0\}$ to the ground-truth failure set, $\mathcal{F} \subset \mathcal{S}$, as well as the maximum $\ell_\mu(z)$ gradient along trajectories generated by π^{nom} with no filtering, averaged across 100 initial conditions.

Ablating α. We ablate the choice of α on our simulated Dubins' example in Section 6.1. We note that for the NoGP baseline, the CBF is not sensitive to lower values of α while the gradient penalty allows us to tune the aggressiveness of the filter (as shown by $|\Delta a|$).

Hyperparameter Selection. Our WGAN inspired method for learning a smooth margin function has several weighting terms. At a high level, λ_{w} dictates encourages safe states to take large positive

8. The only assumption we required of $\ell(z)$ is boundedness for the existence and uniqueness of the value function. However, under simple neural network parameterizations this function is indeed Lipschitz continuous in practice.

Safety Filter	NoGP		GP		None
	Avg. $\lvert\Delta a\rvert$	Safety Rate	Avg. $\lvert\Delta a\rvert$	Safety Rate	Safety Rate
LR	1.5 ± 1.1	97%	2.0 ± 1.3	100%	41%
CBF ($\alpha = 0.7$)	1.3 ± 1.1	96%	1.3 ± 1.1	100%	
CBF ($\alpha = 0.95$)	1.3 ± 1.0	97%	1.1 ± 0.9	100%	

Table 9: **Simulation: On the Quality of Latent Safety Filters.**

values and unsafe states to take large negative values. However, since this does not semantically encode that $\ell(z) < 0 \iff z \in \mathcal{F}$, we use λ_{w} to weigh how much we require the margin function to abide by this constraint. To discourage the Lipschitz constant of this network from growing too large, we add a gradient penalty (Gulrajani et al., 2017) with weight λ_{gp} and Lipschitz constant target β. We analyze the margin function's sensitivity with respect to a range of λ_{w}, λ_{sign}, and λ_{gp} in Figures 4, 5, and 6 using 100 trajectories collected by the base task policy.

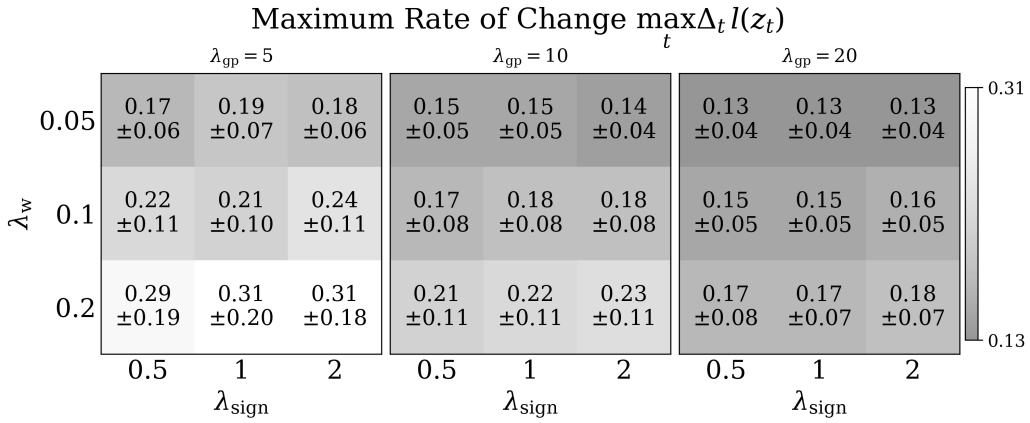

Figure 4: Sensitivity Study: Smoothness

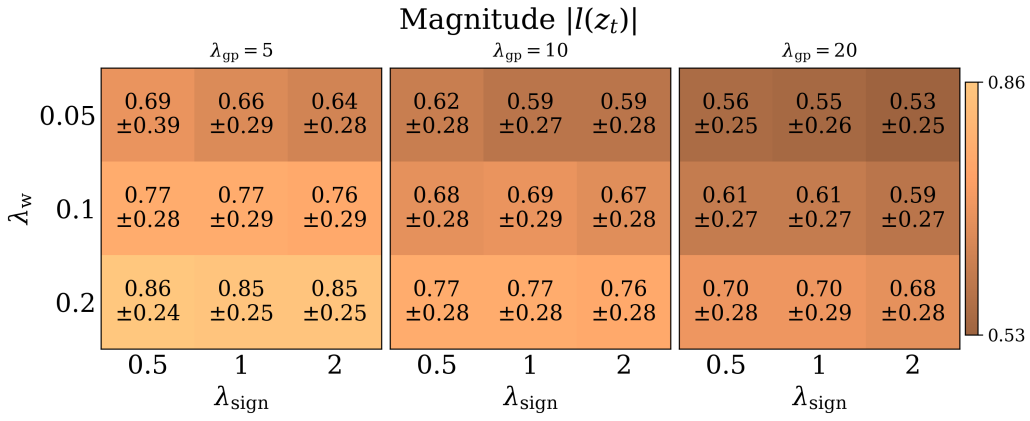

Figure 5: Sensitivity Study: Margin magnitude.

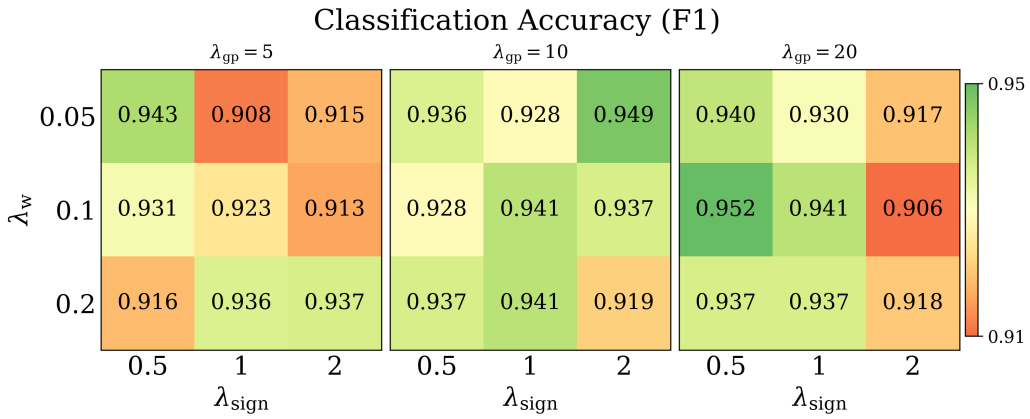

Figure 6: Sensitivity Study: Classification Accuracy.

We consistently find that increasing λ_w increases average magnitude $|l(z_t)|$ and the maximum single-timestep change $\max_t \Delta_t l(z_t)$, while these metrics are relatively insensitive to choice of λ_{sign}. Increasing λ_{gp} generally reduces the sensitivity of $\max_t \Delta_t l(z_t)$ with respect to λ_w and lowers the magnitude $|l(z_t)|$. On the other hand, the F1 score is consistently high across a range of hyperparameters, supporting our claim that the WGAN-GP training objective (8) is independent of additive shifts of $\ell(z)$. In practice, we tuned the hyperparameters in the main body by fixing $\lambda_{gp} = 10$ and selecting the target Lipschitz constant β according to the average gradient norm at the beginning of training. We then tuned λ_w so that the maximum magnitude of the margin function did not greatly exceed 1, (as we bound the margin function between $[-1, 1]$, as noted in Section 9.3). Afterwards, λ_{sign} was tuned so the resulting margin function had high classification accuracy.

9.6. Sampling-Based Optimization of the Latent Discrete-time CBF

Unlike the continuous-time CBF with control-affine dynamics, the discrete-time CBF optimization problem in (5) does *not* admit a quadratic program and is in general nonconvex, making it challenging to optimize efficiently in high-dimensional action spaces. We use a zeroth-order optimization procedure, accelerated with the parallelization capabilities of modern hardware. Specifically, we sample a large number of actions in parallel (e.g., in hardware its 7,600) from a mixture of the nominal task-driven policy π^{nom} and the safety policy $\pi^{\blacklozenge}$ to form a set of actions $\mathcal{A}_{\text{sample}} \subset \mathcal{A}$. We then find a subset of the samples which satisfy the CBF constraint

$$\mathcal{A}_{\text{CBF-Safe}} = \{a_i \in \mathcal{A}_{\text{sampled}} \mid (Q^{\blacklozenge}(z, a_i) - \epsilon) \geq \alpha(V^{\blacklozenge}(z) - \epsilon)\} \tag{10}$$

by evaluating the learned $Q^{\blacklozenge}(s, a_i)$ for each sampled action a_i and comparing it to the safety value assuming we switched to the safety-preserving policy, $V^{\blacklozenge}(z) = Q^{\blacklozenge}(z, \pi^{\blacklozenge}(z))$. Here the hyperparameter ϵ is a small positive constant which accounts for learning inaccuracies of the zero-level set. Finally, we execute actions via:

$$a^\star = \begin{cases} \pi^{\blacklozenge}(\mathcal{E}(o)) & \textbf{if } \mathcal{A}_{\text{CBF-Safe}} = \emptyset \\ \text{argmin}_{a \in \mathcal{A}_{\text{CBF-Safe}}} ||a - \pi^{\text{nom}}(o)|| & \textbf{else} \end{cases} \tag{11}$$

where $\pi^{\blacklozenge}$ is executed in the case where none of the action samples satisfy the discrete-time CBF constraint; otherwise, we return the most similar action sample from $\mathcal{A}_{\text{CBF-Safe}}$ to the nominal policy by evaluating all samples in parallel (e.g., in hardware, the overall process of finding $\mathcal{A}_{\text{CBF-Safe}}$ and evaluating the nearest action takes about 10 ms for 7.6k samples for our 7DOF manipulator).

9.7. Sampling Distributions

For our simulated scenarios, we sample $N = 25$ equally spaced action samples $a \in [-2, 2]$ at each timestep. We also include the actions from $\pi^{\blacklozenge}$ and π^{nom} for a total of 27 action samples.

For our hardware manipulation example, we have a 7-dimensional action space (delta position, axis angle rotation, gripper open/close) which is intractable to sample over exhaustively. Instead, we chose our sampling scheme by carefully linearly interpolating between key vectors over specific action dimension. We fully describe the sampling process in Table 10. For each set of samples we interpolate the specified subset of action dimensions (for example just x action, or x, y, and z) between two actions while holding the remaining dimensions at the action specified by **Static**. For example, while the top row results in a $(400, 7)$ action tensor that interpolates all actions between actions returned by π^{nom} and $\pi^{\blacklozenge}$, the second row only interpolates along the $(\Delta x, \Delta y, \Delta z)$ components while keeping the rotation and gripper actions fixed to that of π^{nom}.

#	Interp. Dims	Interp. From	Interp. To	Static
1	$[\mathbf{x}, \mathbf{y}, \mathbf{z}, \omega_{\mathbf{x}}, \omega_{\mathbf{y}}, \omega_{\mathbf{z}}]$	π^{nom}	$\pi^{\text{◐}}$	π^{nom}
2	$[\mathbf{x}, \mathbf{y}, \mathbf{z}]$			
3	$[\omega_{\mathbf{x}}, \omega_{\mathbf{y}}, \omega_{\mathbf{z}}]$			
4	$[\mathbf{x}]$			
5	$[\mathbf{y}]$			
6	$[\mathbf{z}]$			
7	$[\mathbf{x}, \mathbf{y}, \mathbf{z}, \omega_{\mathbf{x}}, \omega_{\mathbf{y}}, \omega_{\mathbf{z}}]$	$\pi^{\text{◐}}$	$\mathbf{0}$	$\pi^{\text{◐}}$
8	$[\mathbf{x}]$			
9	$[\mathbf{y}]$			
10	$[\mathbf{z}]$			
11	$[\mathbf{x}]$	π^{nom}	$\pi^{\text{◐}}$	
12	$[\mathbf{y}]$			
13	$[\mathbf{z}]$			
14	$[\mathbf{x}]$	$\mathbf{0}$		
15	$[\mathbf{y}]$			
16	$[\mathbf{z}]$			
17	$[\mathbf{x}]$	$\boldsymbol{\mu} - \boldsymbol{\sigma}$	$\boldsymbol{\mu} + \boldsymbol{\sigma}$	π^{nom}
18	$[\mathbf{y}]$			
19	$[\mathbf{z}]$			

Table 10: Scheme for generating action samples for the manipulator. Each linear interpolation takes $N = 400$ samples for a total of 7600 samples. Each line is interpolated by first deciding which set of actions to interpolate from and to, as well as which dimensions participate in the interpolation; while the rest are held static. Bounds $\boldsymbol{\mu} + \boldsymbol{\sigma}$ and $\boldsymbol{\mu} - \boldsymbol{\sigma}$ represent actions $+1$ and -1 standard deviation away from the mean actions from the dataset.