2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Designing Latent Safety Filters using Pre-Trained Vision Models

Ihab Tabbara[1*], Yuxuan Yang[1*], Ahmad Hamzeh[1], Maxwell Astafyev[1], and Hussein Sibai[1]

Abstract— **Ensuring safety of vision-based control systems remains a major challenge hindering their deployment in critical settings. Safety filters have gained increased interest as effective tools for ensuring the safety of classical control systems, but their applications in vision-based control settings have so far been limited. Pre-trained vision representations (PVRs) have been shown to be effective perception backbones for control in various robotics domains. In this paper, we are interested in examining their effectiveness when used for designing vision-based safety filters. We use them as backbones for classifiers defining failure sets, for Hamilton–Jacobi (HJ) reachability-based value functions, and for latent world models. We discuss the trade-offs between training from scratch, fine-tuning the PVRs, and freezing the PVRs when training the models they are backbones for. We also evaluate whether one of the PVRs is superior across all tasks, evaluate whether learned world models or Q-functions are better for switching decisions to safe policies, and discuss practical considerations for deploying these PVRs on resource-constrained devices. Our experiments show that compared to training representations from scratch, using PVRs as perception backbones for vision-based safety filters can reduce violation rates by 12.2%, and fine-tuning PVRs to the task can reduce them by 73.7%, while maintaining or improving task performance. Code is available at `https://github.com/tabz23/Latent-Safety-Filters`.**

I. Introduction

Computer vision plays a critical role in robotics applications, such as autonomous driving [1], manipulation [2], and navigation [3]. It is often the case that the robots do not have direct access to the underlying state of the environment. Instead, they rely on high-dimensional sensory inputs to perceive the world. Images are particularly valuable in this context: they provide rich information about both the robot and the environment states, while being inexpensive to collect compared to other sensing modalities. However, the safety of control systems following vision-based policies remains a major concern that limits their broader deployment in critical domains. Several approaches have been proposed to address this challenge including formal verification [4], [5], online monitoring [6], [7], safe reinforcement learning [8], and safety filtering [9], [10], [11], [12], [13].

Safety filters, such as those based on control barrier functions (CBFs) [14], [15], formally guarantee safety of control systems. CBFs can be used to specify controls that guarantee the forward invariance of a set of states. One can then formulate a quadratic program to generate controls that minimally deviate from reference ones while maintaining the invariance of a set that is deemed safe [16]. Moreover, Hamilton–Jacobi (HJ) reachability analysis [17], which we employ in our study, can also be used for safety filtering. Given a user-defined failure set (e.g., the set of points defining an obstacle in the state space), HJ reachability analysis computes the backward reachable set (BRS), i.e., the set of states from which the agent cannot avoid eventually entering the failure set, as the sublevel set of a value function. The complement of the BRS is the maximal controlled forward invariant set [18]. The analysis also produces a corresponding controller that ensures the forward invariance of that set by maximizing the value function and consequently steering the system away from the BRS and ensuring safety. However, traditional CBF design and HJ reachability analysis require the system dynamics and formally defined failure set and do not scale beyond few dimensions. In complex and dynamic environments, defining a proper state space, obtaining a dynamics model, and formally defining a failure set that has to be avoided becomes challenging. Consequently, deep learning approaches have been proposed to address this challenge [19], [9], [10], [11].

Deep learning approaches for training neural driving policies are either end-to-end or modular. The former train models that directly map sensor observations to control inputs, limiting the information loss attained in modular approaches [19]. For vision-based safety filters, most existing methods follow a modular approach, assuming the existence of a perception model that maps images to interpretable low-dimensional states over which dynamics are known or are approximated [11], [9]. Recently, PVRs, which are trained on large-scale datasets [20], have been shown to improve the performance and sample complexity of learning end-to-end control policies for various control tasks [21], [22], [23]. Such models learn to process images into low-dimensional, non-interpretable, vectors while preserving semantic features. A policy can then be trained on task-specific data to map these features to low-level control, without having to learn common image processing skills from the scarce and costly robotic task-specific data.

In this work, we systematically evaluate five state-of-the-art PVRs across diverse safety-related tasks and environments. We compare frozen, fine-tuned, and trained-from-scratch variants as backbones for both failure classifiers and HJ reachability-based safety filters, and analyze their performance under different tasks. Our findings are as follows: (1) a task-specific representation model can match or even surpass PVRs on safety-control tasks when the PVRs are not fine-tuned; (2) fine-tuning the PVRs during safety-filter training substantially improves collision-avoidance performance; (3) safety filters equipped with DINOv2 [24] consistently per-

[1]Department of Computer Science and Engineering, Washington University in St. Louis, MO 63130, USA. {i.k.tabbara, y.yuxuan, a.h.hamzeh, astafyev, sibai}@wustl.edu

*Equal contribution.

form well across all tasks; (4) when a PVR is well suited for learning a world model, assessing future-state safety with the world model likely outperforms using a Q function, whereas otherwise a Q function is preferable; and (5) PVR-based safety filters are suitable for real-world deployment. To the best of our knowledge, no prior work has evaluated PVRs in the context of safety-critical tasks, nor have existing PVRs been explicitly trained to serve as perception backbones for safety filters rather than reward-driven control policies.

II. Preliminaries

A. Hamilton–Jacobi reachability analysis

Hamilton–Jacobi (HJ) reachability is a control-theoretic approach for safety verification and control synthesis [17]. Consider a dynamical system of the form $s_{t+1} = f(s_t, a_t)$, where $s_t \in S$ and $a_t \in A$. We call the trajectory of the system starting from state s and following a policy $\pi : S \to A$ by $\xi_s^\pi : \mathbb{R}^{\geq 0} \to S$. Given a set of states $\mathcal{F} = \{s \mid h(s) < 0\}$, where $h : S \to \mathbb{R}$, HJ reachability analysis computes the optimal value function $V : S \to \mathbb{R}$, where $V(s) := \sup_{\pi(\cdot)} \inf_{t \geq 0} h(\xi_s^\pi(t))$, which satisfies the fixed-point Bellman equation: $V(s) := \min\{h(s), \max_{a \in A} V(f(s,a))\}$. The associated optimal policy is $\pi(s) := \arg\max_{a \in A} V(f(s,a))$. The set of states in the zero-sublevel set of V, i.e., $\{s \mid V(s) < 0\}$, is called the *backward reachable set* (BRS) of the failure set $\mathcal{F}$. It consists of the states starting from which the system will inevitably reach the failure set and are thus *unsafe*.

For practical implementation in high-dimensional state spaces, we use reinforcement learning (RL) to approximate a time-discounted version of the HJ Q-function [25]. We employ actor-critic methods such as DDPG [26] and DDQN [27]. We optimize the Q-function parameters θ by minimizing the loss function:

$$L(\theta) = E_{(s_t, a_t, s_{t+1}) \sim D}\left[(Q_\theta(s_t, a_t) - y_t)^2\right] \qquad (1)$$

where the target is computed as: $y_t = (1-\gamma)h(s_t) + \gamma \min\{h(s_t), \max_{a \in A} Q_\theta(s_{t+1}, a)\}$ .When the Q-function is learned, the optimal safety-preserving policy can be retrieved as follows: $\pi_{\text{safe}} := \arg\max_{a \in A} Q(s,a)$ and the corresponding HJ value function evaluated at state s would be: $V(s) = \max_{a \in A} Q(s,a)$.

B. Pre-trained vision models

We evaluate several popular PVRs which are trained using different datasets, objectives, and methods: VC-1 [22], a Vision Transformer (ViT), was trained on a union of robotic and natural images using masked autoencoding (MAE). We use its last-layer CLS token as its encoding of the input image. R3M [23], based on ResNet-50, was trained on the large-scale Ego4D egocentric video dataset with time-contrastive learning, video-language alignment, and an L1 penalty to encourage sparse representations. DINOv2 [24], a self-supervised ViT trained with a teacher–student distillation framework on the LVD-142M dataset (a dataset selected from a corpus comprising benchmarks for image classification, image segmentation, image retrieval, and depth estimation). We consider both the concatenation of last-layer patch embeddings (DINO) and the CLS token output (DINO-CLS) as representations. Finally, ResNet-50 [28] serves as a baseline convolutional encoder trained on ImageNet [29] and CIFAR-10 with supervised classification objectives.

C. Latent world models and DINO-WM

World models aim to predict future states of an environment given previous observations and actions, allowing agents to simulate outcomes without direct interaction. Latent world models encode high-dimensional observations in compact representations and model the dynamics in the latent space. A latent world model consists of three components: (i) an **encoder** $z_t \sim \text{enc}_\theta(z_t \mid o_t)$ that maps high-dimensional observations $o_t \in O$ to latent states $z_t \in Z$, (ii) a **transition model** $z_{t+1} \sim p_\theta(z_{t+1} \mid z_{t-H:t}, a_{t-H:t})$ that predicts future latent states given previous and current latents and actions, and (iii) a **decoder** $\hat{o}_t \sim q_\theta(o_t \mid z_t)$ that reconstructs observations from the latent states, where H is the temporal history length.

DINO World Model (DINO-WM) [30] builds on this framework by leveraging pretrained visual representations to enable task-agnostic dynamics learning. Unlike approaches such as DreamerV3 [31], which train encoders from scratch with a reward signal to extract task-specific features, DINO-WM employs frozen pretrained encoders, providing rich semantic and spatial priors without reward supervision. Training proceeds as follows: input RGB images are encoded with a frozen pretrained encoder to produce embeddings that define the latent space; the transition model, implemented as a Vision Transformer (ViT), processes sequences of latent embeddings; and action conditioning is achieved by mapping K-dimensional action vectors through a multilayer perceptron and concatenating them to the encoded visual features. Proprioceptive data—information about the ego agent's internal state (e.g., joint positions, velocities) is incorporated in the same manner. Teacher forcing is used during training, where ground-truth observations are fed instead of predictions, together with a latent consistency loss:

$$\mathcal{L}_{\text{pred}} = \left\| p_\theta(\text{enc}_\theta(o_{t-H:t}), \phi(a_{t-H:t})) - \text{enc}_\theta(o_{t+1}) \right\|_2^2, \quad (2)$$

where ϕ is the action encoder. This design allows DINO-WM to exploit pretrained visual priors while learning predictive dynamics directly in latent space, providing strong representations without requiring reward-based supervision. In our work, we adapt DINO-WM to use any of R3M, Resnet, VC1, DINO, and DINO-CLS as the pre-trained vision encoders.

III. Related Work

PVR as backbones for control Several works have shown the benefits of using PVRs as backbones for control policies. The authors in [21] showed that frozen PVR backbones can be competitive with, and sometimes outperform, policies with access to ground truth states in various benchmarks. The authors in [22] further compared frozen and fine-tuned PVRs across control tasks, finding no single dominant model and introducing VC1, which outperforms others on average. In

this work, we evaluate state-of-the-art PVRs as backbones for safety filters, comparing frozen, fine-tuned, and models trained from scratch across multiple tasks.

Vision-based safety filters HJ-based safety filters in latent space have recently been explored. The works of [32] and [33] demonstrate this using DINO-WM [30] with a frozen DINO backbone, as well as DreamerV3 [31], which instead learns a task-specific image encoder during training. Aside from HJ-based latent safety filters, CBFs have also been used in vision-based settings [34], [11].

IV. Methodology

In this section, we first describe the training of DINO-WM, latent-space failure classifiers, and safety filters. We then outline our evaluation methodology for the latter two.

A. Learning latent dynamics

We adapt DINO-WM [30] to learn dynamics as well as proprioception and action encoders across five PVRs: VC1, R3M, ResNet, DINO, and DINO-CLS. For each environment, we also train a new vision encoder, denoted WM-R, by randomly initializing a ResNet architecture and training DINO-WM end-to-end, thereby learning the representation and the dynamics jointly. We set the history length to $H = 3$ and train each model for 100 epochs on datasets we collected with the environment's reference controller.

B. Learning the failure classifier in latent space

For the open-loop evaluation, we learn a function h defined over the latent space, whose 0-sublevel set specifies the failure region, i.e., $\mathcal{F}_{o_f} = \{o_f \in O \mid h(\text{enc}_\theta(o_f)) \leq 0\}$. We train h as a multi-layer perceptron (MLP), and employ a hinge-style loss to encourage margin separation between the failure region and the non-failure region: $\mathcal{L}_h = \sum_{o_f \in \mathcal{F}_{o_f}} \sigma(\alpha - h(\text{enc}_\theta(o_f))) + \sum_{\bar{o}_f \in \bar{\mathcal{F}}_{o_f}} \sigma(\alpha + h(\text{enc}_\theta(\bar{o}_f)))$, where σ is the ReLU function and the hyperparameter $\alpha = 0.75$ to prevent the neural network from predicting single value, o_f corresponds to the observation inside the failure set $\mathcal{F}_{o_f}$, and $\bar{o}_f$ corresponds to the observation inside the compliment set of the failure set $\bar{\mathcal{F}}_{o_f}$. We access the failure labels of the observations using the simulator of each environment.

To improve the learned boundary, we incorporate a *gradient penalty* term on interpolated latent features between failure and non-failure states: $\mathcal{L}_{\text{gp}} = \mathbb{E}_{\tilde{z}}\left[(\|\nabla_{\tilde{z}} h(\tilde{z})\|_2 - \lambda)^2\right]$, where $\tilde{z} = \alpha_{\text{gp}} h(\text{enc}_\theta(o_f)) + (1 - \alpha_{\text{gp}}) h(\text{enc}_\theta(\bar{o}_f))$, with $\alpha_{\text{gp}} \sim \mathcal{U}(0, 1)$ and $\lambda = 2.1$ being the target gradient norm. The final training objective is $\mathcal{L} = \mathcal{L}_h + \beta \mathcal{L}_{\text{gp}}$, where $\beta = 0.1$ controls the strength of the penalty. This setup encourages h to be discriminative and smooth near failure set boundary.

C. Learning the HJ value function as the safety filter for closed-loop evaluation

We train the HJ value function as the safety filter using DDPG for environments with continuous action spaces and DDQN for environments with discrete action spaces and optimize the loss in (1). When the simulator provides a ground-truth distance function, we use it directly; otherwise, if it only provides boolean failure labels, we rely on the output of the learned classifier $h(\text{enc}_\theta(o))$ introduced in Section IV-B.

For closed-loop evaluation, we use a switching scheme that can operate in either `critic-only` or `dynamics lookahead` mode presented in Algorithm 1. Both variants decide whether to follow the nominal policy π_{nom} or switch to the safe policy π_{safe} generated by the HJ reachability analysis. In `critic-only` mode, the decision relies directly on the safety critic Q. In `dynamics lookahead` mode, the world model $\hat{f}_\eta$ predicts the next latent state under the nominal action, and the critic evaluates its safety based on this prediction.

Algorithm 1: Switching between reference and safe controller

Input: latent state z_t, nominal policy π_{nom}, safety critic Q, world model $\hat{f}_\eta$, margin $\tau \geq 0$, method $\in$ {`Critic-only`, `dynamics lookahead`}

Output: execute action a_t

```
a_nom ← π_nom(z_t);
if method = dynamics lookahead then
    ẑ_{t+1} ← f̂_η(z_t, a_nom); p ← max_a Q(ẑ_{t+1}, a);
else
    p ← Q(z_t, a_nom);
if p < τ then
    a_t ← π_safe(z_t) := arg max_a Q(z_t, a);
else
    a_t ← a_nom;
```

D. Fine-tuning PVRs and learning representation models

When learning the failure classifier in latent space, we first train while freezing the PVR backbones and then repeat the experiment with fine-tuning them. In the latter case, the loss backpropagates through all the encoders from DINO-WM. We follow the same steps when training the HJ value function: one variant with frozen backbones and another with fine-tuned backbones. Importantly, when fine-tuning the backbones for HJ training, the learned dynamics can no longer be used since they were trained on the frozen representation, which changed. In addition, for both the failure classifier and HJ value functions, we train ViTs from scratch to test whether the task-specific losses alone are sufficient to learn a good representation. We call the learned vision representation model in these tasks "Scratch". Altogether, for each of the failure classifier in latent space and HJ tasks we obtain 13 model variants: five frozen PVR backbones, five fine-tuned PVR backbones, one frozen WM-R backbone (a randomly initialized model with ResNet architecture trained jointly with dynamics when training DINO-WM), one unfrozen WM-R model, and one "Scratch" model, referring to a randomly initialized ViT model that is trained end-to-end for each task, where the loss is backpropagated through the backbone to jointly learn both the representation and either the latent failure classifier or the HJ value function.

E. Evaluation metrics

1) Classification with learned failure classifier: We probe each backbone's ability to distinguish failure from non-failure states with the learned failure classifiers. First, we compute the correlation between the learned h and the ground-truth distance function provided by the simulator. When the ground truth distance function is not accessible, we compute the correlation with the binary failure labels. A high correlation indicates that the learned representation is sufficiently rich to capture how h evolves along a trajectory (i.e., whether the agent is moving closer to an obstacle at any given timestep). We also probe the PVRs by showing classification accuracy for the failure and non-failure states.

2) HJ in Closed Loop: We evaluate two conditions: (1) **No Safety Filter**, where the nominal policy acts alone, and (2) **HJ Safety Filter Enabled**, where the reference policy is accompanied by the HJ safety filter. Each evaluation consists of 50 independent episodes with random initial states, which are kept fixed across all backbones and switching methods for fair comparisons. We report the **success rate** (percentage of episodes that reach the task goal), **violation count** (average number of states where a failure state is reached, e.g., collision), inference time, and the size of each safety filter.

V. Experimental Setup

We evaluate the above approach in four simulated environments. In each environment, we describe specific tasks, failure conditions, proprioception of the agent, and data collected to train the DINO-WM and the failure classifiers. All models were trained using either an NVIDIA A40 or RTX 5090, 200 GB of system RAM and between 2-16 CPU cores depending on the environment. Each world model was trained on a A40, requiring 1–2 days per environment–backbone pair on average. HJ training was conducted on an RTX 5090, taking 1–2 days on average per backbone.

A. Environments

ManiSkill (UnitreeG1PlaceAppleInBowl): We simulate a humanoid (UnitreeG1) robot in ManiSkill [35] to pick up an apple, with a random initial position, and place it in a bowl. The observation is an RGB image captured by a camera mounted on the robot's head. The proprioception $\mathbf{x}$ contains the positions and velocities of the 25 joints of the robot. $o_f \in \mathcal{F}_{o_f}$ if the robot's hand comes into contact with the bowl. We trained a PPO policy on the full state which includes $\mathbf{x}$ and environment information, such as the bowl's and the apple's position. In this task, where only binary collision labels are available from the simulator, we first learn the failure classifier as described in Section IV-B, then use it to learn the HJ safety filter as described in Section IV-C. We collect and label a dataset of 3,000 episodes using the trained PPO policy.

Dubins Car (2D Navigation): We simulate a Dubins car moving in a 2D plane at constant speed with two obstacles. The observation is an overhead RGB image of the environment. The task is to navigate to a goal location without colliding with obstacles. The proprioception is $\mathbf{x} = (x, y, \theta)$. We exclude the proprioception from all training, as (x, y, θ) can be inferred from the RGB image. The reference is a PID controller tracking the distance to the obstacle. We determine if $o_f \in \mathcal{F}_{o_f}$ using the ground-truth distance to the closest obstacle from the simulator. We collect and label a dataset of 2,000 episodes using the PID controller.

Safety Gymnasium CarGoal: An agent must navigate to a target goal position in an arena that contains hazardous areas in Safety Gymnasium [36]. The visual observation is an egocentric view from the car, and the proprioception $\mathbf{x} \in \mathbb{R}^{24}$ contains information about the agent, such as angular velocities, gyro, etc. We train a nominal policy to reach goals using DreamerV3 [31] without consideration of any obstacles (no cost signal is given while learning the nominal policy). We define $o_f \in \mathcal{F}_{o_f}$ if a collision occurs between the car and any hazard, which is determined using LIDAR measurements from the environment. We collect and label a dataset of 2,000 episodes using the policy obtained from DreamerV3.

CARLA: In CARLA [37], we used a scenario in which the ego vehicle followed a leading non-ego. The lead vehicle implemented a lane-following PID-driven controller, while the ego car uses a noisy PID controller (to generate trajectories with collisions). For simulated trajectories, vehicle models and spawn points are randomly chosen from CARLA's predefined vehicle blueprints and Town10 environment, respectively. Raw observations are RGB images captured from a camera mounted at the center of the windshield. The proprioception is $\mathbf{x} = (x, y, z, v, \theta, \psi, \phi)$, which are respectively the ego vehicle's position (x, y, z), velocity, and orientation (pitch, yaw, roll). $o_f \in \mathcal{F}_{o_f}$ when two vehicles collide. We collect and label a dataset of 2,000 episodes using the noisy PID controller.

VI. Results

In this section, we present and analyze our results by addressing a set of research questions.

A. Are PVRs sufficient as backbones for safety filtering, or do we train representations from scratch for every task?

The question can be divided into two parts: Are PVRs sufficient for (1) distinguishing safe from unsafe observations and (2) training safety filters?

We first evaluate which representation model is most suited as the vision backbone for the failure classifiers in latent space. As shown in Table II, when backbones are not fine-tuned, failure classifiers using WM-R achieve the highest correlation scores on average, and those using Scratch underperform. After fine-tuning the PVRs, they outperform both failure classifiers with WM-R and Scratch backbones. From Figure 1, we observe that failure classifiers with WM-R as the backbone achieve accuracies comparable to those using non-fine-tuned PVRs, while the Scratch backbone performs noticeably worse than both WM-R and the PVRs.

To address the second question, we compare the PVRs with both the Scratch and the WM-R backbones. Since the dynamics model is unavailable for the Scratch backbone, we compare it only against safety filters evaluated using

Method	Fine-tuned PVR	PVR	ManiSkill		Dubins		CarGoal	
			Vio.	Succ.	Vio.	Succ.	Vio.	Succ.
Critic-only	No	DINO-CLS	**1.28**	**0.34**	4.96	0.98	48.62	0.44
		DINO	3.52	0.10	9.80	0.98	57.70	0.20
		VC1	4.76	0.02	8.92	0.98	53.48	0.38
		R3M	1.88	0.38	8.44	0.98	73.42	0.86
		ResNet	5.40	0.32	8.76	0.98	54.82	0.42
		WM-R	3.82	0.18	**4.66**	**0.98**	**48.26**	**0.32**
	Yes	DINO-CLS	**0.42**	**0.44**	**0.60**	**0.98**	30.68	0.26
		DINO	4.62	0.36	2.58	0.98	35.78	0.26
		VC1	1.74	0.00	7.26	0.98	**19.40**	**0.24**
		R3M	0.44	0.50	7.90	0.98	48.52	0.32
		ResNet	5.30	0.32	5.50	0.98	19.74	0.20
		WM-R	8.10	0.42	4.50	0.98	35.70	0.06
Dynamics lookahead	No	DINO-CLS	3.64	0.44	5.68	1.00	68.82	0.44
		DINO	**1.86**	**0.54**	9.44	0.98	**35.16**	**0.22**
		VC1	4.62	0.34	8.68	0.98	77.36	0.58
		R3M	8.14	0.32	8.18	0.98	35.60	0.00
		ResNet	3.74	0.50	8.18	0.98	60.36	0.98
		WM-R	3.56	0.54	**4.72**	**0.98**	79.62	0.42
Critic-only	-	Scratch	2.82	0.02	5.96	0.98	36.04	0.28
		Nominal	4.98	0.36	8.18	0.98	61.06	1.00

TABLE I: Success rates (Succ.) and total violations (Vio.) for safety filters with different backbones across tasks. "Method" indicates whether the critic-only or the dynamics lookahead algorithm was used in the evaluation. **green** indicates the lowest violation among all backbones per method and fine-tune setting, while underline denotes the lowest violation for each PVR across different methods and fine-tune settings. We also highlight the corresponding success rates of the highlighted lowest violation.

the `Critic-only` method. In Table I, the results show that, compared to filters without backbone fine-tuning and using the `Critic-only` method, the safety filters using the Scratch and WM-R backbones perform better on average. In particular, in the Dubins car experiment, the safety filters using the Scratch backbones achieve the lowest number of violations in CarGoal and the third lowest in both ManiSkill and Dubins compared to the safety filters with non-fine-tuned PVR backbones. Also, WM-R, which learns its representation during world-model training, achieves the lowest average number of violations on both Dubins car and CarGoal PVRs and performs relatively well in Maniskill compared to the non-fine-tuned PVRs. These findings suggest that when the backbone is not fine-tuned during safety-filter training, a task-specific representation such as Scratch and WM-R can match or even surpass PVRs trained on large datasets such as the ones we consider. Nevertheless, across all tasks, safety filters with fine-tuned PVRs generally reduce violations more than those with the Scratch and WM-R backbones. For example, Table I shows that after fine-tuning the DINO-CLS backbone during the HJ training, its safety filter achieves the safest performance across all tasks compared to safety filters using WM-R and Scratch as backbones.

Considering the `critic-only` method, across all tasks, the best frozen PVR (DINO-CLS) reduces violations by 12.2% over Scratch and 19.8% over WM-R on average, with success rates improving by 42.1% over WM-R and 552.4% over Scratch. The violation reduction is primarily driven by the ManiSkill experiment (54.6% vs. Scratch, 66.5% vs. WM-R), while the one with CarGoal shows no meaningful advantage, suggesting frozen PVRs might be more beneficial in visually complex manipulation tasks.

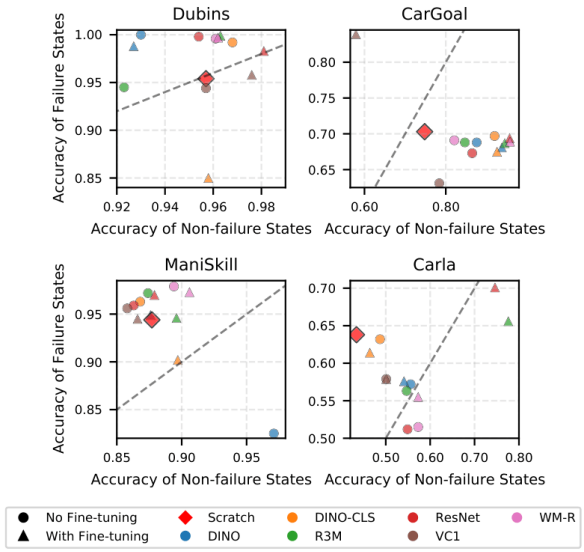
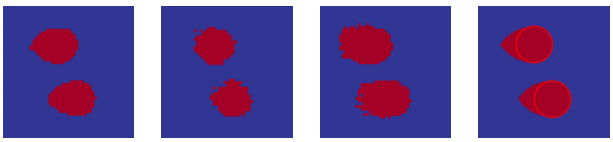


Fig. 1: Probing classification accuracy of the failure set $\mathcal{F}$ and the non-failure set $\bar{\mathcal{F}}$ for failure classifiers with and without backbone fine-tuning. Dashed lines denote $y = x$.



(a) DINO (FT) (b) DINO (c) Scratch (d) GT

Fig. 2: BRS recovered from HJ value functions learned in latent space of different PVRs. θ is fixed to 0, and red pixels denote states within the BRS (i.e., $V(z) < 0$). GT denotes the ground truth BRS.

Success improvements vs. Scratch are heavily inflated by ManiSkill's near-zero Scratch baseline (0.02). Fine-tuning the PVRs substantially amplifies these gains: the best fine-tuned PVR reduces violations by 73.7% over Scratch and 75.7% over WM-R (fine-tuned) and improves success rates across all three tasks.

To understand why Scratch performs better compared to some frozen PVRs but subpar compared to fine-tuned ones while training the HJ, we visualize the learned BRS for the Dubins task in Figure 2. It can be seen how, for the HJ value function using DINO as the backbone, the recovered BRS without fine-tuning is worse than that recovered from the filter trained with the Scratch backbone. On the other hand, the recovered BRS from the HJ value function after fine-tuning the DINO backbone is substantially more accurate. Since HJ training time is comparable for frozen and fine-tuned backbones, fine-tuning is better than training new backbones from scratch for each task, as we do when training "Scratch" and "WM-R".

B. Should we fine-tune the PVRs when learning the safety filters?

Table I shows that during the closed-loop evaluation, only two of the 18 safety filters exhibit an increase in

the average number of violations when their backbones are fine-tuned, which means fine-tuning the backbone during training generally improves safety filters' collision-avoidance performance. This finding is consistent with prior work on robot control tasks that do not consider safety [22]. However, fine-tuning the backbones can render the safety filter overly conservative. On the CarGoal task, all safety filters (except the DINO-based filter) experience a decrease in success rate when their backbones are fine-tuned.

Considering the `critic-only` method, fine-tuning the PVRs yields a 71.7% reduction in constraint violations over the best frozen PVR on average over all tasks. The magnitude of improvement varies across backbones and tasks, with different PVRs benefiting to different degrees from fine-tuning. Success rates are roughly preserved on average (-5.4%).

C. Does any PVR consistently outperform the others across all tasks?

We observe that safety filters using DINO-CLS as the backbone, especially when it is fine-tuned, consistently perform well across all tasks. Among filters with fine-tuned backbones, DINO-CLS-based filters achieve the lowest average number of violations on Dubins car and ManiSkill, and the third-lowest on CarGoal. Among filters with frozen backbones, they achieve the second-lowest number of violations on Dubins car and CarGoal, and the lowest on ManiSkill.

Prior work reports VC1's superiority on control benchmarks, outperforming R3M across general control tasks [22]. It is therefore surprising that in our evaluations, VC1 does not consistently achieve top-tier performance and occasionally underperforms R3M. This finding suggests that a PVR well-suited for control policies is not necessarily well-suited for safety filtering.

Another finding is that, despite sharing the same visual backbone, safety filters using DINO as a backbone perform worse than those using DINO-CLS. We hypothesize that DINO's representation, formed by concatenating all patch embeddings, introduces redundancy, inflates the dimensionality of the state space, and dilutes safety-relevant signals, making it harder to learn an HJ value function, which typically does not scale well beyond a few dimensions. In contrast, DINO-CLS relies on the CLS-token embedding, yielding a compact, semantically aggregated representation that is easier for the filter to use. Because DINO's higher-dimensional representation requires larger models and more data to be effective, holding model size and training data fixed makes DINO-CLS better as a backbone for safety filtering.

D. Critic-only, or dynamics lookahead for switching between the safe and the nominal policies ?

Fine-tuning the backbones renders the pre-trained dynamics models ineffective as it alters their encoders. Although the dynamics model can be retrained with the fine-tuned encoder, this incurs long additional training time. This issue raises another question: if the dynamics model is not retrained after encoder fine-tuning, does a safety filter trained without fine-tuning the backbone and evaluated with the `dynamics lookahead` method outperform a filter with a fine-tuned backbone evaluated with the `Critic-only` method?

In both CarGoal and ManiSkill, safety filters with non-fine-tuned backbones using the `dynamics lookahead` method generally exhibit substantially more violations compared to their fine-tuned counterparts evaluated with the `Critic-only` method (Table I). In contrast, on the Dubins car task, safety filters with the `dynamics lookahead` method slightly outperform those that fine-tune the backbones and use the `Critic-only` method. Overall, these results demonstrate that fine-tuning the PVRs of the safety filters and evaluating them with the `Critic-only` method is generally better than using `dynamics lookahead` and not fine-tuning the PVRs.

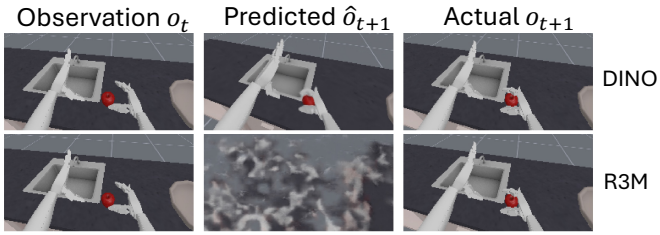


Fig. 3: Visual observations predicted using DINO-WM

The previous conclusion does not hold for all backbones. For example, the DINO backbone, whose representations concatenate patch embeddings leading to a very large vector representation of the size of 75284, the learned dynamics model is particularly accurate, as shown in Figure 3. In these cases, its performance, as shown in Table I, is consistently higher when evaluated with the dynamics model rather than the critic, indicating that when the backbone is well-suited to learning dynamics, the dynamics model might be the better choice.

On the other hand, when the backbone is not well-suited to learning dynamics, the critic is preferable. As shown in Figure 3, the dynamics model's predicted observations for R3M are worse than those for DINO, leading the safety filter with an R3M backbone to incur an average of 8.14 violations per episode, substantially higher than the 1.86 observed with DINO. When we use the `Critic-only` method instead of `dynamics lookahead`, the safety filter using R3M as a backbone achieves substantially fewer violations compared to that using DINO in both fine-tuned and non-fine-tuned PVR settings.

Given the drastic difference in R3M's performance using the `dynamics lookahead` and the `Critic-only` methods in ManiSkill (both fine-tuned and non-fine-tuned), we can deduce that R3M serves as an effective PVR for encoding safety-related features but is not well-suited as a representation model for predicting dynamics in the ManiSkill setup, in contrast to DINO. These observations highlight the importance of a PVR that not only supports effective feature extraction but also facilitates accurate next-state prediction.

E. Do we need a world model?

We observe in our closed-loop experiments that critic-based estimation of next-state HJ values for the fine-tuned PVR outperforms the dynamics-based approach on the non-fine-tuned PVR. If the dynamics is not used, and we can instead directly train an HJ while fine-tuning the backbone and using the critic to estimate the HJ value of the next state as in the `critic-only` method, why do we need to train a world model?

For safety filters, simply encoding the visual observations and combining them (through concatenation or other techniques) with raw proprioception inputs to form the latent state is not sufficient. Recent work has demonstrated that fusing vision and proprioceptive information in a shared latent space outperforms simple concatenation of raw proprioceptive data with vision embeddings [38]. Wu et al. [39] achieve superior performance by encoding proprioceptive information through an MLP and aligning it with vision embeddings, compared to direct concatenation. Similarly, both Dreamerv3 [31] and DINO-WM [30] incorporate dedicated proprioceptive encoders in their architectures. When training our world models, we also observed that naive concatenation of proprioceptive and action information with vision embeddings yields poor predictive performance for future proprioception in world model training.

In our experiments, each environment provides proprioceptive inputs that provide the model with historical context (e.g., velocities in CarGoal, Carla, and ManiSkill), which cannot be inferred from a single observation. In settings without such proprioception information, the latent state must encode dynamics-related features such as velocities for the HJ value function to accurately estimate the BRS in latent space. One can either concatenate representations from H previous timesteps or use a Recurrent State-Space Model (RSSM) such as the Dreamerv3 [31] world model, which maintains a recurrent state that compresses historical observations. In the latter setting, the `Critic-only` method cannot be used without the world model, since the latent state depends on the recurrent state r_t, i.e., $z_t \sim \text{enc}_\theta(z_t \mid o_t, r_t)$ where r_t is updated at every time step by the world model.

F. If the learned failure classifier is good, does it imply learning a good safety filter?

Our results indicate that obtaining a strong failure classifier does not necessarily translate into an effective safety filter when using the same backbone. As an example, as seen in Table II, we observe that DINO performs well on ManiSkill, Dubins car, and CarGoal, attaining one of the highest correlation scores when fine-tuned. However, when used as the backbone for training safety filters, which is shown in Table I, it does not achieve the lowest number of violations on any of these tasks, regardless of fine-tuning.

G. Are PVR-based safety filters lightweight enough for deployment?

In practice, inference speed and model size are critical, as they determine whether a model can be deployed in real-world applications and on edge devices. Table III reports the total inference time and model size of the different filters. The maximum inference time of all the safety filters is 1.93×10^{-2} s, which is sufficient for real-time operation at 50 Hz. The largest model size of all the safety filters is 611 MB, which fits within the storage capacity of most edge devices. Combining the above information, we conclude that the safety filters with PVR backbones can be deployed in real-world applications.

PVR	ManiSkill		Dubins	
	FT	**No FT**	**FT**	**No FT**
DINO-CLS	0.449	0.451	0.908	0.976
DINO	0.617	**0.770**	0.949	**0.981**
VC1	0.489	0.461	0.956	0.958
R3M	0.542	0.514	0.942	0.963
ResNet	**0.620**	0.451	0.966	0.974
WM-R	0.517	0.442	**0.978**	0.935
Scratch	0.494	-	0.922	-
	CarGoal		**Carla**	
	FT	**No FT**	**FT**	**No FT**
DINO-CLS	0.899	0.901	0.600	0.238*
DINO	**0.927**	0.923	0.314	0.305
VC1	0.789	0.785	0.428	0.556
R3M	0.924	0.871	0.496	0.488
ResNet	0.902	0.899	0.551	0.395
WM-R	0.909	**0.930**	**0.615**	**0.638**
Scratch	0.801	-	0.436	-

TABLE II: Correlation scores for failure classifiers across environments and backbones. Backbones in **green** are best backbone for each task in the case of fine-tuning (FT) and freezing the backbone (NoFT) while training the latent failure classifer. Underlined backbones correspond to whether the FT or the Non-FT version performed better for each task. * denotes statistically insignificant correlation ($p \geq 0.05$).

	DINO-CLS	DINO	VC1	R3M	Resnet	WM-R
Inference Time (ms)	16.1	19.3	16.7	16.3	8.9	11.9
Model Size (MB)	291	510	611	275	275	275

TABLE III: Inference time and model size for safety filters with different backbones. Model size includes the world model (encoders and predictor) and the safety filter. Inference time accounts for all steps: encoding the observation, predicting the next latent state, and computing safe action.

VII. Conclusion

We presented an evaluation of pre-trained visual representations (PVR) as perception backbones for safety filters. We compared several PVRs, considering frozen, fine-tuned, and scratch-trained variants across multiple simulated environments and analyzed their efficacy for both failure classification and safety filter learning. Our results show that, compared with training from scratch, safety filters equipped with an appropriately chosen frozen PVR reduce the violation rate by 12.2% on average over the three safe control environments without degrading task achievement rates. Moreover, fine-tuning the PVR further improves safety, yielding a 73.7% reduction in violation rate, which highlights the importance of PVRs for safety filter design. Although our empirical results demonstrate the effectiveness of PVRs for designing safety filters, these safety filters currently lack

theoretical guarantees. Existing work on formally verifying neural safety filters remains largely limited to small-scale models. We believe that extending formal verification to vision-based neural safety filters is a promising direction.

REFERENCES

[1] Waymo, "Introducing the 5th-generation waymo driver," https://waymo.com/blog/2020/03/introducing-5th-generation-waymo-driver/, 2020, accessed: 2024-09-12.

[2] S. H. Bengtson, T. Bak, L. N. Andreasen Struijk, and T. B. Moeslund, "A review of computer vision for semi-autonomous control of assistive robotic manipulators (arms)," *Disability and Rehabilitation: Assistive Technology*, vol. 15, no. 7, pp. 731–745, 2020.

[3] B. E. Tweddle and A. Saenz-Otero, "Relative computer vision-based navigation for small inspection spacecraft," *Journal of guidance, control, and dynamics*, vol. 38, no. 5, pp. 969–978, 2015.

[4] S. M. Katz, A. L. Corso, C. A. Strong, and M. J. Kochenderfer, "Verification of image-based neural network controllers using generative models," *Journal of Aerospace Information Systems*, vol. 19, no. 9, pp. 574–584, 2022.

[5] C. Hsieh, Y. Li, D. Sun, K. Joshi, S. Misailovic, and S. Mitra, "Verifying controllers with vision-based perception using safe approximate abstractions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4205–4216, 2022.

[6] H. Torfah, A. Joshi, S. Shah, S. Akshay, S. Chakraborty, and S. A. Seshia, "Learning monitor ensembles for operational design domains," in *Runtime Verification*, P. Katsaros and L. Nenzi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 271–290.

[7] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone, "Real-time anomaly detection and reactive planning with large language models," *arXiv preprint arXiv:2407.08735*, 2024.

[8] J. García, Fern, and o Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, 2015. [Online]. Available: http://jmlr.org/papers/v16/garcia15a.html

[9] S. Dean, A. Taylor, R. Cosner, B. Recht, and A. Ames, "Guaranteeing safety of learned perception modules via measurement-robust control barrier functions," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 654–670. [Online]. Available: https://proceedings.mlr.press/v155/dean21a.html

[10] H. Abdi, G. Raja, and R. Ghabcheloo, "Safe control using vision-based control barrier function (v-cbf)," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 782–788.

[11] M. Tong, C. Dawson, and C. Fan, "Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 10 511–10 517.

[12] I. Tabbara and H. Sibai, "Learning conservative neural control barrier functions from offline data," *arXiv preprint arXiv:2505.00908*, 2025.

[13] I. Tabbara, Y. Yang, and H. Sibai, "Statistically assuring safety of control systems using ensembles of safety filters and conformal prediction," *arXiv preprint arXiv:2511.07899*, 2025.

[14] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[15] A. Alan, A. J. Taylor, C. R. He, A. D. Ames, and G. Orosz, "Control barrier functions and input-to-state safety with application to automated vehicles," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 6, pp. 2744–2759, 2023.

[16] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[17] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.

[18] I. Fialho and T. Georgiou, "Worst case analysis of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 6, pp. 1180–1196, 1999.

[19] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[20] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[21] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. Gupta, "The unsurprising effectiveness of pre-trained vision models for control," in *international conference on machine learning*. PMLR, 2022, pp. 17 359–17 371.

[22] A. Majumdar, K. Yadav, S. Arnaud, J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, T. Wu, J. Vakil *et al.*, "Where are we in the search for an artificial visual cortex for embodied intelligence?" *Advances in Neural Information Processing Systems*, vol. 36, pp. 655–677, 2023.

[23] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," *arXiv preprint arXiv:2203.12601*, 2022.

[24] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[25] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8550–8556.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[27] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[30] G. Zhou, H. Pan, Y. LeCun, and L. Pinto, "Dino-wm: World models on pre-trained visual features enable zero-shot planning," *arXiv preprint arXiv:2411.04983*, 2024.

[31] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," 2024. [Online]. Available: https://arxiv.org/abs/2301.04104

[32] J. Seo, K. Nakamura, and A. Bajcsy, "Uncertainty-aware latent safety filters for avoiding out-of-distribution failures," *arXiv preprint arXiv:2505.00779*, 2025.

[33] K. Nakamura, L. Peters, and A. Bajcsy, "Generalizing safety beyond collision-avoidance via latent-space reachability analysis," *arXiv preprint arXiv:2502.00935*, 2025.

[34] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "Barriernet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, pp. 1–19, 2023.

[35] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su, "Maniskill2: A unified benchmark for generalizable manipulation skills," in *International Conference on Learning Representations*, 2023.

[36] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang, "Safety gymnasium: A unified safe reinforcement learning benchmark," *Advances in Neural Information Processing Systems*, vol. 36, pp. 18 964–18 993, 2023.

[37] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[38] L. Wang, X. Chen, J. Zhao, and K. He, "Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers," *Advances in neural information processing systems*, vol. 37, pp. 124 420–124 450, 2024.

[39] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Conference on robot learning*. PMLR, 2023, pp. 2226–2240.