From Foresight to Forethought: VLM-In-the-Loop Policy Steering via Latent Alignment

Yilin Wu[1], Ran Tian[2], Gokul Swamy[1], Andrea Bajcsy[1]
[1]Carnegie Mellon University [2]UC Berkeley
{yilinwu, gswamy, abajcsy}@andrew.cmu.edu, rantian@berkeley.edu

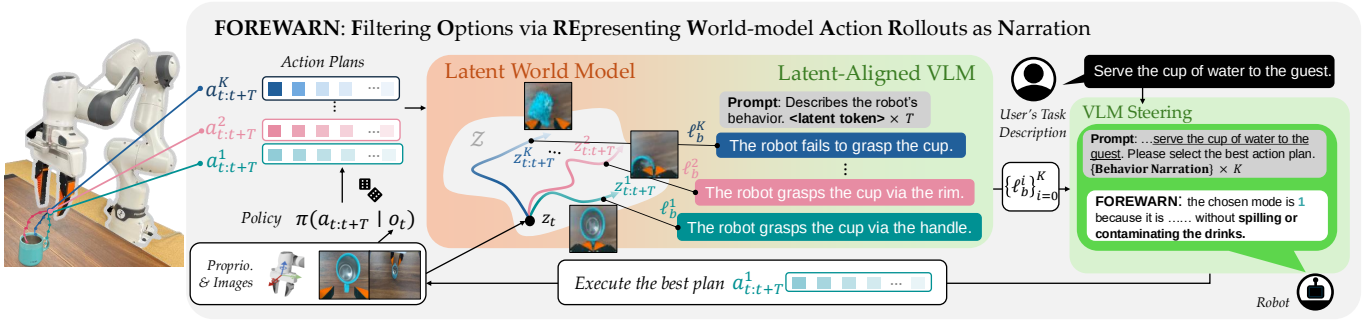

Fig. 1: We present **FOREWARN**, an VLM-in-the-loop policy steering algorithm for multi-modal generative robot policies. Our key idea is to decouple the VLM's burden of predicting action outcomes from evaluation. By predicting action outcomes with a pre-trained latent dynamics model and aligning a VLM to reason about these latent states in text, FOREWARN can select action plans at runtime that are most appropriate for new task contexts and user needs.

***Abstract*—While generative robot policies have demonstrated significant potential in learning complex, multimodal behaviors from demonstrations, they still exhibit diverse failures at deployment-time. Policy steering offers an elegant solution to reducing the chance of failure by using an external verifier to select from low-level actions proposed by an imperfect generative policy. Here, one might hope to use a Vision Language Model (VLM) as a verifier, leveraging its open-world reasoning capabilities. However, off-the-shelf VLMs struggle to understand the consequences of low-level robot actions as they are represented fundamentally differently than the text and images the VLM was trained on. In response, we propose FOREWARN, a novel framework to unlock the potential of VLMs as open-vocabulary verifiers for runtime policy steering. Our key idea is to decouple the VLM's burden of predicting action outcomes (*foresight*) from evaluation (*forethought*). For foresight, we leverage a latent world model to imagine future latent states given diverse low-level action plans. For forethought, we align the VLM with these predicted latent states to reason about the consequences of actions in its native representation—natural language—and effectively filter proposed plans. We validate our framework across diverse robotic manipulation tasks, demonstrating its ability to bridge representational gaps and provide robust, generalizable policy steering. Videos can be found on the project website: https://yilin-wu98.github.io/forewarn/.**

I. Introduction

Generative imitation-based policies are an increasingly powerful way to learn low-level robot behaviors from multi-modal[1] expert demonstrations [6, 14, 51]. Despite their impressive ability to learn diverse behaviors directly from high-dimensional observations, these policies still degrade in nuanced and unexpected ways at runtime. For example, consider the robot in the left of Figure 1 that must pick up a mug from the table. At training time, the generative policy learns a distribution over useful interaction modes such as grasping the cup by different parts (e.g., handle, lip and interior, etc.) shown in wrist camera photo in Figure 1.

However, at runtime, the policy exhibits a range of degradations, from complete task failures (such as the robot knocking down the cup during grasping, shown in the center of Figure 1), to inappropriate behaviors that are misaligned with the deployment context or preferences of an end-user (such as the robot placing its gripper inside of a cup of water when serving a guest shown in the middle of Figure 1). While a common mitigation strategy involves re-training the policy via more demonstrations [37] or interventions [34, 25], runtime failures are not always an indication that the base policy is fundamentally incapable of producing the desired behavior. In fact, the base policy may already contain the "right" behavior mode within its distribution (e.g., grasping the cup by the handle), but due to putting too much probability mass on an undesired mode, the robot does not reliably choose the correct action plan at runtime.

Runtime policy steering [27, 47] offers an elegant solution to this mode-selection problem. By using an external *verifier* to select candidate plans proposed by an imperfect generative policy, the robot's behavior can be "steered" at runtime without any need for re-training. Despite the initial successes demonstrated by the policy steering paradigm, the question still remains of how to fully unlock autonomous policy steering in the open world when the robot's environment, task context, and base policy's performance are constantly changing.

Policy steering can be approached via the stochastic model-

[1]Here, multimodality only refers to the training data's action distribution.

predictive control framework of modern control theory, which decomposes the optimal action selection (i.e. *generation*) problem of runtime policy steering into *(a) predicting* the outcomes of a given action plan and *(b) verifying* how well they align with user intent. However, this approach is only feasible when a physics-based, low-dimensional dynamics model is available for outcome prediction and a well-defined reward function can be specified for verification. In open-world environments, both of these requirements are challenging to fulfill due to the complexity of dynamics modeling and the difficulty of hand-crafting rewards to evaluate nuanced task requirements [17].

Our core idea is to leverage world models, which are well-suited for predicting action outcomes in open world settings, and VLMs, which have great potential as verifiers due to their commonsense reasoning abilities, to develop a divide-and-conquer approach to open-world policy steering. However, doing so naively is challenging as world models and VLMs operate on fundamentally different representations of reality.

To address this concern, we propose **FOREWARN**: **F**iltering **O**ptions via **RE**presenting **W**orld-model **A**ction **R**ollouts via **N**arration. To predict challenging action outcomes (e.g., interaction dynamics of a manipulator and a deformable bag), we use state-of-the-art world models [25, 50] to predict lower-dimensional latent state representations from high-dimensional observation-action data (shown in orange in the center of Figure 1). To critique behavior outcomes under nuanced task specifications (e.g., "Serve the cup of water to the guest"), we leverage vision-language models (VLMs) [11, 29] as our open-world verifiers (shown in green in the center of Figure 1). Importantly, we demonstrate that *aligning* the VLM to reason directly about the predicted latent states from the world model enables it to understand fine-grained outcomes that it cannot directly predict zero-shot nor understand from image reconstructions. Ultimately, this alignment step enables our "VLM-in-the-loop" policy steering approach to interpret action plans as behavior narrations and select high-quality plans by reasoning over those narrations even under novel task descriptions (shown on the right of Figure 1).

We evaluate **FOREWARN** on a suite of robotic manipulation tasks, demonstrating how it can robustly filter proposed action plans to match user intent and task goals even when faced with variations not seen during training. In summary, our main contributions are:

- Formalizing runtime policy steering a stochastic model-predictive control problem, revealing the *generation-verification gap* [15, 9, 43] and where state-of-the-art models have maximal potential to shine.
- A latent space alignment strategy that enables a VLM to more reliably verify action plan outcomes predicted by a low-level, action-conditioned world model.
- A novel, fully-autonomous policy steering framework that improves a base generative imitation-based policy by over 30%, even in novel task descriptions.
- Extensive experiments in hardware showing that our latent-aligned VLM approach outperforms (by $\sim 40\%$) altnerative VLM approaches that do *not* decouple the prediction of outcomes from verification.

Due to space constraints, a full discussion of related work is provided in App. A.

II. Problem Formulation

We formalize the general problem of policy steering as a stochastic model-predictive control problem over the set of action plans proposed by a base action generation model.

Setup & Notation. Let the robot's pre-trained multimodal imitative action generation model [6, 23] be denoted by $\pi(\mathbf{a}_t \mid o_t)$. We will often refer to this model as the robot's "base policy" throughout this paper with which it performs a task for an end-user. The robot's observations $o \in \mathcal{O} := \mathcal{I} \times \mathcal{Q}$ combine RGB image data $I \in \mathcal{I}$ and proprioceptive states $q \in \mathcal{Q}$ (e.g., end-effector pose, gripper state), and $\mathbf{a}_t := a_{t:t+T}$ denotes a robot's T step action plan, with each action in the sequence specifying end-effector positions and rotations. Similarly, $\mathbf{o}_t := o_{t:t+T}$ is an observation sequence. In the real world, after observing some o_t, generating some action plan $\mathbf{a}_t \sim \pi(\cdot \mid o_t)$ and executing it open-loop, the robot observes a sequence of observations $\mathbf{o}_t \sim \mathcal{P}(\cdot|o_t, \mathbf{a}_t)$.

Problem. Given the robot's current observation o_t at timestep t and K *i.i.d.* samples from the base policy, $\{\mathbf{a}_t^i\}_{i=1}^K \sim \pi(\mathbf{a}_t \mid o_t)$, the *policy steering* problem seeks to return the action plan $\mathbf{a}_t^\star$ which optimizes the following objective:

$$\mathbf{a}_t^\star = \arg \max_{\mathbf{a}_t \in \{\mathbf{a}_t^i\}_{i=1}^K} \mathbb{E}_{\mathbf{o}_t \sim \mathcal{P}(\cdot|o_t, \mathbf{a}_t)} \Big[R(\mathbf{o}_t; \ell) \Big]. \tag{1}$$

Fundamentally, solving the *(behavior) generation* problem specified in Eq.1 requires two abilities: *prediction* and *verification*. The *prediction* problem can be clearly seen in the expectation of Eq.1, wherein we have to forward simulate the outcomes of any action plan $\mathbf{a}_t^i$ and "imagine" the potential future observations, $\mathbf{o}_t \sim \mathcal{P}(\mathbf{o}_t \mid o_t, \mathbf{a}_t^i)$. The *verification* problem lies inside of the expectation and with the reward function $R(\mathbf{o}_t; \ell)$. Here, $\ell \in \mathcal{L}$ is the task description, represented via a language description (e.g., "Serve the cup of water to the guest."), and R verifies how well or how poorly the future observations align with the behavior specified by ℓ.

While Eq.1 characterizes the underlying policy steering problem, it is extremely challenging to solve end-to-end because of the coupled prediction and verification steps. This is where we seek to leverage vision-language models in-the-loop to obtain a practical solution with the potential to generalize to new environments and hard-to-model steering criteria. Initially, it may be tempting use the VLM directly as a black-box solver of Eq.1 (i.e. to solve the overarching behavior generation problem) by simply passing it the K action plan options, the current observation o_t, and the task description ℓ, and having it directly return $\mathbf{a}_t^\star$. However, low-level action data is beyond the training distribution of current VLMs, which primarily focus on high-level semantic understanding and not embodied control [21]. Alternatively, we could fine-tune the VLM to directly select action plans via labeled observation-action-samples datasets (labeled with

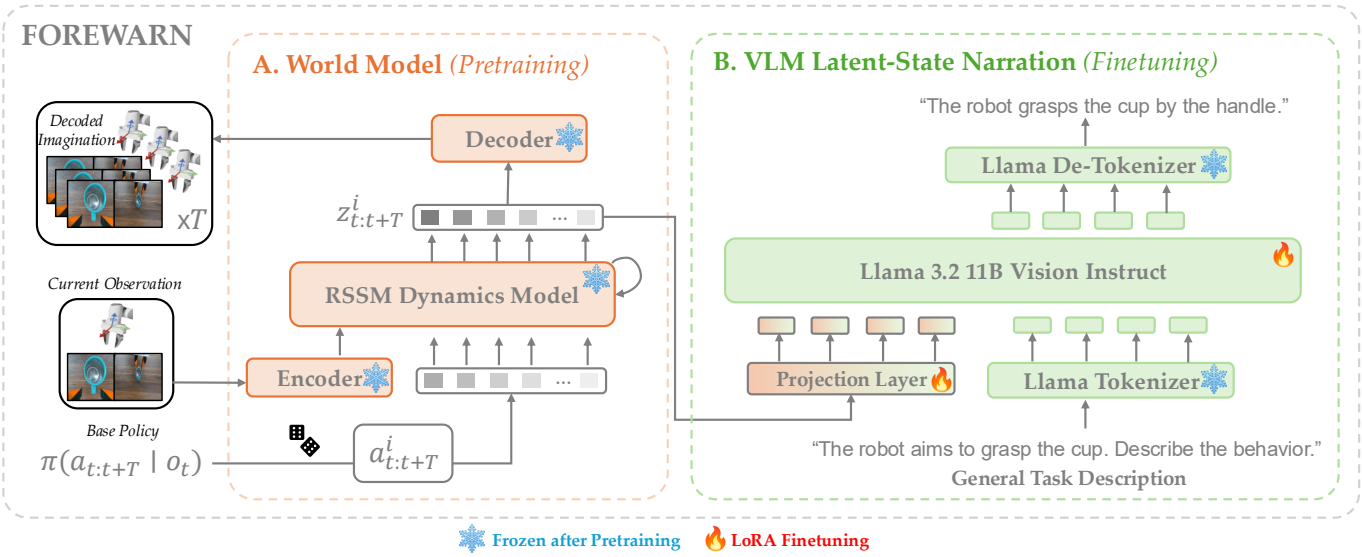

Fig. 2: **Training FOREWARN.** In part A (Sec. III-A), a Recurrent State Space Model (RSSM) is pretrained to learn good latent embeddings of the dynamics conditioned on the observations and actions. In part B (Sec. III-B), the sequence of learned latent embeddings is projected through a linear layer to the text embedding space, similar to the original vision token processing in the Llama-3.2 Model. The projection layer and Llama model are finetuned using LoRA [20] with a frozen world model.

optimal action trajectories). However, this strategy is sample-inefficient, requiring extensive embodied rollouts and human annotations to generate labels. Instead, we propose tackling the problem in Eq.1 in a way that leverages the unique strengths of a VLM. We hypothesize that the right place to put VLMs into the loop is specifically in the *verification* step above, as it leverages the model's strong reasoning abilities *given predicted outcomes*. Then, to complement the VLM, we propose that *prediction* should be handled by an embodied world model that can reason about hard-to-model outcomes directly from low-level actions. It is well known that verification is significantly easier than generation for many problems [9, 43], boding well for the sample-efficiency of our modular approach.

III. Our Approach: FOREWARN

Our key idea is to adopt a divide-and-conquer strategy, explicitly decoupling the VLM's burden of predicting action outcomes from evaluation during policy steering. Specifically, we take advantage of recent advances in latent dynamics models [25, 50] which can learn lower-dimensional latent state representations from high-dimensional observation-action data collected on a robot. By passing possible action generations to the latent dynamics model, we can efficiently predict diverse future outcomes that would hard to model otherwise. Importantly, even though the world model learns a latent state that is an approximate sufficient statistic of the dynamics, one still needs to teach the VLM to evaluate outcomes *in the latent representation of the world model*, rather than from the raw image observations, as we will address in detail below.

At the highest level, our mathematical formulation of model-predictive "VLM-in-the-loop" policy steering is:

$$\begin{aligned} \mathbf{a}_t^\star = \arg \max_{\mathbf{a}_t \in \{\mathbf{a}_t^i\}_{i=1}^K} \mathbb{E}_{\mathbf{z}_t \sim f_\phi(z_t, \mathbf{a}_t)} \left[R_\psi^{\text{VLM}}(\mathbf{z}_t; \ell) \right], \\ \text{s.t.} \quad z_t = \mathcal{E}_\phi(o_t), \end{aligned} \tag{2}$$

where $z_{t+1} \sim f_\phi(z_t, a_t)$ is a probabilistic latent dynamics model, $\mathbf{z}_t := z_{t:t+T}$, $\mathcal{E}_\phi : \mathcal{O} \to \mathcal{Z}$ is an observation encoder that maps the robot's current observations o_t into a latent space $\mathcal{Z}$, and $R_\psi^{\text{VLM}}(\mathbf{z}_t; \ell)$ represents an implicit reward function embedded in our VLM (parameterized by ψ) which evaluates the predicted outcomes given the task description. Both f_ϕ and $\mathcal{E}_\phi$ are part of our world model and parameterized by ϕ. In short, to realize our idealized policy steering objective (1), we approximate the expectation over future outcomes with world model rollouts and leverage a fine-tuned VLM as a latent-outcome-conditioned verifier, leading to (2). We call our overall policy steering approach **FOREWARN**: **F**iltering **O**ptions via **RE**presenting **W**orld-model **A**ction **R**ollouts as **N**arration, visualize it in Figure 1, and now discuss the details.

A. Foresight: Predicting Outcomes via Latent World Models

In this work, we adopt the Dreamerv3 world model [18] as our mechanism for predicting future outcomes of action plans. This world model (visualized on the left in Fig. 2) is pre-trained via an offline dataset that contains trajectories of the robot's observations and actions: $\mathbb{D}_{\text{WM}} = \{\{(o_\tau^j, a_\tau^j)\}_{\tau=0}^{H-1}\}_{j=1}^M$, where M is the total number of trajectories with each trajectory representing the robot's outcome $o_{0:H}^j$ induced by an action plan $a_{0:H}^j$, and H denotes the length of these trajectories. The training data consists of both successful and failed rollouts from the base policy $\pi(\mathbf{a} \mid o)$ and additional demonstration data. This allows the world model to accurately

predict the outcomes of both good and bad actions plans, a sufficient condition for successful behavior generation [32].

The world model training loss incentivizes the latent state to be informative for high-quality image decoding as well as highly predictive of the next latent state given an action (see [18] and App. C2 for more details). After training, the world model is frozen, and we utilize the trained $\mathcal{E}_\phi$ and f_ϕ throughout the rest of our policy steering algorithm.

B. Forethought: Latent-Text Alignment for Outcome Reasoning and Policy Steering

Once the world model encoder $\mathcal{E}_\phi$ learns an effective low-dimensional latent state representation z_t and the latent dynamics model f_ϕ learns to predict future latent states conditioned on the robot's actions, we can shift our focus to evaluating the imagined outcomes of any candidate low-level action plan. We hypothesize that the VLM's open-world reasoning capabilities could allow it to be an effective verifier here, enabling better decision-making downstream.

However, before we can unlock the potential of the VLM as a verifier, we need to first enable the VLM to reason directly about the predicted latent states (rather than in terms of raw image observations). We propose approaching this problem as a *latent-text alignment* problem: by mapping latent states generated by the world model to a textual representation, we can enable the VLM to more easily evaluate plans due to its strong natural language understanding abilities. We then further frame this alignment problem as a *visual question-answering (VQA)* task, where we prompt the VLM to narrate the real-world behavior of the robot conditioned on a sequence of latent states produced by the world model (visualized on the right of Figure 2). We now discuss this process in detail.

VLM Backbone. We use the Llama-3.2-11B-Vision-Instruct model [11] as our VLM backbone. The original Llama model utilizes a Vision Transformer (ViT) to tokenize visual inputs into latent tokens, which are then processed by the LLM backbone to generate text outputs. In our setting, we adapt the Llama model by replacing its observation (image) tokenization module with our world model's encoder $\mathcal{E}_\phi$ and latent dynamics model f_ϕ. To align the latent dynamics embedding space with the text embedding space, we introduce a linear layer as an adapter. Specifically, $\mathcal{E}_\phi$ encodes the current robot observation, and the forward dynamics model f_ϕ predicts future latent states based on a candidate action plan (left side of Figure 2). These latent states are passed through the linear layer to produce a sequence of latent tokens, which serve as input to the LLM backbone (right side of Figure 2).

VLM Finetuning. Our objective is to enable the VLM to narrate the real-world behavior of the robot, denoted as ℓ_b^i, based on a sequence of generated latent states, $z_{t:t+T}^i$ from the world model. These behavior narrations highlight fine-grained motion details (e.g., "the robot grasps the cup by the handle") rather than the high-level semantics of the motion (e.g., "the robot grasps the cup"). This design encourages the model to capture nuanced behaviors and identify potential failures in the robot's predicted outcomes. We construct a VQA dataset to finetune the VLM to achieve this objective (described in Sec. IV and App. C3).

VLM-In-the-Loop Policy Steering. Our fine-tuned VLM can now describe the outcomes of candidate action plan in natural language narration. However, our ultimate goal is to query the VLM to identify the best action sequence. To accomplish this, we propose querying the same VLM again (i.e. using it as a verifier), leveraging its open-world reasoning capabilities and natural language understanding to select the best action plan. Mathematically, policy steering can be expressed as:

$$\begin{aligned} \mathbf{a}_t^\star = \arg \max_{\mathbf{a}_t \in \{\mathbf{a}_t^i\}_{i=1}^K} \mathbb{E}_{\mathbf{z}_t \sim f_\phi(z_t, \mathbf{a}_t)} \Big[R_\psi^{\text{VLM}}\big(\mathcal{T}_\psi^{\text{VLM}}(\mathbf{z}_t; \ell); \ell\big) \Big], \\ \text{s.t.} \qquad z_t = \mathcal{E}_\phi(o_t). \end{aligned} \tag{3}$$

where $\mathcal{T}_\psi^{\text{VLM}} : \mathcal{Z}^T \to \mathcal{L}_b$ denotes the inference process that translates a candidate action plan into the associated robot behavior narration. Instead of instructing the VLM to explicitly assign the reward for each predicted action plan outcome, we leverage its implicit knowledge about the reward and directly query the VLM for the best action plan among the K candidates, conditioned on the task description l (see the example in right side of Figure 1), which can be seen as a form of multiple-choice question answering (MCQA).

In summary, by integrating the predictions from the world model with the VLM's open-vocabulary behavior narration generation and commonsense reasoning, *FOREWARN* guides the robot towards action plans that are aligned with the deployment context and task goals and enables the robot to proactively prevent failures.

IV. Experiments

In this section, we first instantiate several real-world manipulation tasks to study our method. We evaluate the closed-loop policy steering performance as well as our method's robustness to novel task descriptions, ℓ (Sec. IV-A). In App. D2, We also investigate how effectively we can translate low-level actions into high-level behavior descriptions (App. D2).

Real Robot Setup. We use a Franka Emika robotic manipulator equipped with a 3D printed gripper from [7] in our experiments. The base generative policy controls the low-level robot actions (which are the robot's end-effector pose and gripper opening) controlled at 15Hz. The robot uses RGB images from a wrist-mounted camera and a third-person camera mounted in front of the robot. See App. D1 for more details.

Manipulation Tasks. We consider three real-world robot manipulation tasks that exhibit underlying multi-modal behaviors, hard-to-model outcomes, and nuanced failures. In the **Cup** task, the robot must grasp a cup placed in front of it and in the **Bag** task, the robot must pick up a bag of chips from the table. Fundamentally, both tasks can be accomplished in diverse ways: for example, for the **Cup** task, the robot can pick up the cup by the handle or by placing its fingers inside the cup

by grasping the lip; for the **Bag** task, the robot can grab the bag by any edge, or by squeezing the center. Furthermore, the **Bag** task has more challenging dynamics and potential outcomes because the bag is deformable. We use this task to study how our framework performs when faced with harder-to-predict interaction outcomes and nuanced failures (e.g., crushing the chips inside the bag).

We also introduce a third task that features longer horizon and more complex interactions ***Fork-to-Bowl Transfer***. In this task, the robot must pick up a fork from the table and place it inside a bowl. This task is considerably more challenging than the other two tasks for three reasons: 1) it requires longer-horizon planning to effectively navigate distinct phases; 2) requires reasoning about interactions between objects (the fork and the bowl); and 3) it introduces different multi-modal motion choices, including selecting the optimal picking location (tines or handle) and determining the appropriate placement strategy (inside versus outside the bowl, low versus high release). In the top row of Figure 14, we visualize the camera observations of the robot interacting with these items for Bag and Cup tasks.

Base Policy. For our multi-modal imitative action generation model, $\pi(\mathbf{a}_t \mid o_t)$, we use a Diffusion Policy [6] trained on 100 teleoperated demonstrations per task. The policy takes inputs from wrist and third-person cameras, along with proprioceptive states, to predict a distribution over T-step action plan, where $T = 64$. Please see App. C1 for more details.

World Model Training. We collected 250 real-world trajectories per task, including both successful and failed rollouts from the base policy, along with additional 100 demonstrations used in base policy, for training and evaluating the world model (300 for training and 50 for testing). The world model is trained to predict $T = 64$ future latent states given the current o_t and an action plan $\mathbf{a}_t$.

VLM Fine-tuning. We construct our VQA dataset for fine-tuning from the same offline dataset, $\mathbb{D}_{\text{WM}}$, used to train the world model. For each T-step trajectory snippet $\{(o^j_\tau, a^j_\tau\}^{t+T}_{\tau=t}$ from the dataset, the encoder $\mathcal{E}_\phi$ processes the initial observation o^j_t at timestep t, and the forward dynamics model f_ϕ predicts latent states $z^j_{t:t+T}$ based on the action plan $a^j_{t:t+T}$. We note that in our specific implementation, although f_ϕ is a stochastic dynamics function, we use only the most likely prediction as the outcome associated with the action plan. To avoid "semantically repetitive" latent states between adjacent low-level actions, we downsample the T-step latent states to T/4. These downsampled latent states, along with the task description ℓ, are provided as input to the VLM, and we manually annotate the corresponding behavior narrations, $\ell_b \in \mathcal{L}_b$, for the associated observations $o^j_{t:t+T}$. We fine-tune the model using the Low-Rank Adaptation (LoRA) technique [20], keeping both the encoder $\mathcal{E}_\phi$ and the latent dynamics model f_ϕ frozen during the fine-tuning process.

VLM-In-the-Loop Policy Steering. When deploying **FOREWARN** for run-time policy steering, we begin by sampling 100 action plans from the base policy and aggregating them into $K = 6$ modes using the non-maximum suppression (NMS) scheme from [36]. These 6 aggregated action plans are then passed to **FOREWARN** for interpretation and evaluation. For each candidate action plan, we use only the most likely future latent state predictions from f_ϕ as input to the VLM for reasoning.

A. Policy Steering for Open-World Alignment

In this section, we compare our approach against several baselines to evaluate its system-level policy steering performance and robustness under **novel** task descriptions and specifications. Our experiments focus on evaluating the impact of leveraging the VLM as both an interpreter and evaluator of predicted action outcomes.

Baselines. We first keep the VLM's role as the verifier unchanged and ablate the effect of using an explicit world model to predict action outcomes. Specifically, we compare **FOREWARN** to **VLM-Act** (described in Figure D2), which directly fine-tunes the original Llama model to predict action outcome narrations without utilizing a world model. Similar to our approach, **VLM-Act** then queries the VLM to select the best motion plan based on the task description l. Next, we keep the world model unchanged and ablate the effect of alignment: mapping latent states generated by the world model to their underlying textual representation that the VLM can easily reason about.

We compare our approach against two more baselines: (3) **VLM-DynLat-Category**, which also leverages a world model for outcome predictions and a VLM for action plan selection. However, instead of decoding the predicted outcomes into text representations and leveraging the VLM's open-world knowledge for policy steering, it directly fine-tunes the VLM to predict a set of indices of the successful candidate action plans and randomly selects one index from the set to execute the corresponding action plan. (4) **Classifier-Dyn-Latent**, which is similar to **VLM-DynLat-Category**, but instead of relying on a VLM, it directly takes the predicted latent embeddings $\hat{z}_{t:t+T}$ as input and trains a transformer-based binary classifier (commonly used in prior failure-prediction work [25]) to predict success or failure for each candidate action plan and randomly selects one predicted to succeed.

Metrics. We evaluate each policy steering method using the **Success Rate**. For each method, we conduct 20 trials with randomly initialized task configurations and report the average success rate across these trials. A trial is considered successful if the robot successfully completes the task while aligning with the end-user's preferences.

Results: Policy Steering Performance. We first evaluate our approach and the baselines under task descriptions ℓ that fall within the training distribution. Table I shows the success rates of the base robot policy (without policy steering), our approach (**FOREWARN**), and the baselines for each task. Our results demonstrate that **FOREWARN** can effectively steer the policy towards safe and aligned behavior modes by leverging the

Method	Success Rate ↑					
	Training Task Description			Novel Task Description		
	Cup	Bag	Fork	Cup	Bag	Fork
Base Policy	0.30±0.14	0.20±0.13	0.10±0.09	0.50±0.16	0.40±0.15	0.30±0.14
FOREWARN (Ours)	**0.80±0.13**	**0.70±0.14**	**0.70±0.14**	**0.80±0.13**	**0.70±0.14**	**0.60±0.15**
VLM-DynLat-Category	**0.80±0.13**	0.40±0.15	0.50±0.16	0.30±0.14	0.40±0.15	0.30±0.14
Classifier-Dyn-Latent	**0.80±0.13**	**0.70±0.14**	**0.70±0.14**	0.00±0.00	0.10±0.09	0.20±0.13
VLM-Act	0.40±0.15	0.20±0.13	0.20±0.13	0.30±0.14	0.50±0.16	0.20±0.13

TABLE I: **Policy Steering.** The success rate is reported by averaging over 20 different rollouts. FOREWARN outperforms all the baselines in both training and novel task contexts.

VLM as an interpreter and evaluator of predicted latent action outcomes.

Interestingly, Classifier-Dyn-Latent achieves comparable performance to our approach in all tasks under this setting while VLM-DynLat-Category fails to match FOREWARN in more complicated **Bag** and **Fork** tasks. VLM-Act has similar performance as base policy because the behavior narrations, directly generated from low-level action sequences, fail to capture accurate motion details and thus provide no useful signal for VLM to steer the policy. Next, we assess their generalization abilities under novel task descriptions to evaluate the robustness of our approach.

Results: Robustness to Novel Task Descriptions. We evaluate the success rate of each approach when the task description is altered to introduce novel scenarios. Specifically, in the **Cup** task, we modify the original task description from *"Please grasp a cup from the table and serve the cup of water to the guest"* to a novel description: *"Please grasp a cup from the table, but note that the handle is covered with oil."* This change makes behaviors where the robot grasps the cup by the handle unsafe, while grasping the cup by the rim becomes the desired behavior. In the **Bag** task, we modify the original task description from *"Please pick up a bag of chips from the table and minimize the contact region to avoid crushing contents inside"* to a novel description: *"Please pick up a bag of chips from the table and maximize stability without dropping the bag."* This change makes behaviors where the robot picks up the bag by the corner less preferred, as the bag may slip from the gripper, while squeezing the middle of the bag to secure it becomes the desired behavior. In the **Fork** task, we modify the original prompt from *"Please pick up the fork and place it in the bowl while maintaining the sanitation for eating."* to a novel description: *"Please pick up the fork and place it in the bowl while maximizing the contact region during grasping."* This change makes the behaviors where the robot picks up the handle and drop the fork high less preferred as the fork handle is more narrow than tines and dropping high can make fork fall out of the bowl while grasping by tines is more secure.

Interestingly, while both VLM-DynLat-Category and Classifier-Dyn-Latent improve task success rates when task descriptions fall within the training distribution, their performance drops greatly with novel task descriptions. This indicates that the VLM struggles to reason directly about predicted action outcomes from the world model's latent states and essentially degrades to a traditional end-to-end model. To fully leverage the VLM's open-world reasoning capabilities for generalized policy steering, it is essential for VLM to interpret predicted action outcomes through textual representations.

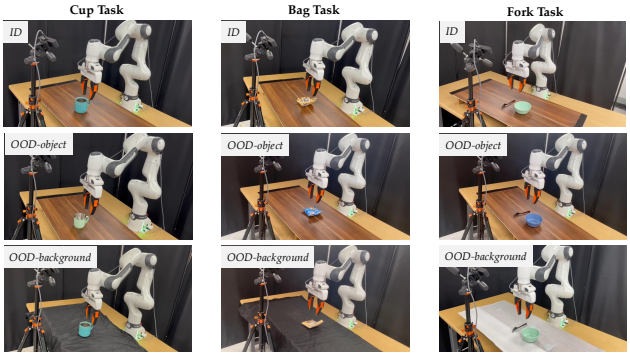

Fig. 3: **Generalization to Environmental Changes.** For each task, we test our method against similar objects of different colors and sizes and also change the table cover to test the system against background variations.

Results: Qualitative Examples. In Figure 15, we present examples of runtime policy steering using our approach for the **Fork** task and additional examples for **Cup** and **Bag** tasks are included in Appendix D2. FOREWARN consistently demonstrates superior performance by selecting motion plans that align with task descriptions and user preferences, even in scenarios with novel task specifications. In contrast, the baselines either fail to interpret action outcomes effectively, resulting in unsafe behaviors, or experience severe performance degradation in novel task specifications.

Results: Out-of-distribution Generalization. Table I has shown our system's robustness to novel task descriptions. Moreover, we further test our system's generalization capability by adding variations in the environment. Specifically, we study the variations in object appearances including colors and sizes and variations in background. Among the six tested scenarios shown in Fig 3, our method can generalize to those variations with roughly 10% performance drop in overall success rate while maintaining a similar performance improvement 40% to 50% from the base policy.

Supplementary Experiments & Analyses. To provide an in-depth analysis of our approach, we performed additional studies on behavior narration translation of VLM as well as the impact of each component on the overall system and a detailed comparison with baselines. We also showcase an additional application of our system as a runtime monitor, opening up potential future directions for soliciting supervisor assistance. Additional inference speed analysis is also included. All these experiments can be found in App. D2.

V. Conclusion

In this work, we investigate the problem of policy steering for multi-modal generative policies. We propose a novel framework **FOREWARN** that unlocks Vision Language Models (VLMs) to serve as open-vocabulary verifiers for run-

time policy steering. Our method decouples the steering problem as future prediction (foresight) and outcome assessment (forethought). Through an explicit world model and a latent-space alignment strategy, we enable VLMs to reason about sensorimotor data using natural language. Our experiments across diverse manipulation tasks confirm that **FOREWARN** not only provides interpretable and reliable failure detection, but also significantly enhances policy success rates through flexible, generalizable steering.

ACKNOWLEDGMENTS

This work is funded in part by a Google Research Scholar Award. Gokul Swamy is supported by STTR grant. We also want to thank Junwon Seo for providing feedback for the paper and Michelle Zhao, Pranay Gupta, Kensuke Nakamura, Lasse Peters for the helpful discussions of ideas.

REFERENCES

[1] Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *8th Annual Conference on Robot Learning*, 2024.

[2] anonymous authors. Anonymous title. In *Robotics: Science and Systems*, 2024.

[3] James Bagnell, Sham M Kakade, Jeff Schneider, and Andrew Ng. Policy search by dynamic programming. *Advances in neural information processing systems*, 16, 2003.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.

[6] 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.

[7] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[8] AgiBot World Colosseum contributors. Agibot world colosseum. https://github.com/OpenDriveLab/AgiBot-World, 2024.

[9] Stephen A Cook. The complexity of theorem-proving procedures. In *Logic, automata, and computational complexity: The works of Stephen A. Cook*, pages 143–152. 2023.

[10] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *International Conference on Learning Representations*, 2025.

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[12] Nicolas Espinosa-Dice, Sanjiban Choudhury, Wen Sun, and Gokul Swamy. Efficient imitation under misspecification. *arXiv preprint arXiv:2503.13162*, 2025.

[13] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A robotic dataset for learning diverse skills in one-shot. In *RSS 2023 Workshop on Learning for Task and Motion Planning*, 2023.

[14] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

[15] Kurt Godel. Letter to john von neumann, 1956. URL https://ecommons.cornell.edu/server/api/core/bitstreams/46aef9c4-288b-457d-ab3e-bb6cb1a4b88e/content.

[16] Lin Guan, Yifan Zhou, Denis Liu, Yantian Zha, Heni Ben Amor, and Subbarao Kambhampati. Task success is not enough: Investigating the use of video-language models as behavior critics for catching undesirable agent behav-

iors. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=otKo4zFKmH.

[17] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. *Advances in neural information processing systems*, 30, 2017.

[18] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[19] Asher J Hancock, Allen Z Ren, and Anirudha Majumdar. Run-time observation interventions make vision-language-action models more visually robust. *arXiv preprint arXiv:2410.01971*, 2024.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[21] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Hao-Shu Fang, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*, 2023.

[22] Motonari Kambara and Komei Sugiura. Future success prediction in open-vocabulary object manipulation tasks based on end-effector trajectories, 2025. URL https://arxiv.org/abs/2412.19112.

[23] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Forty-first International Conference on Machine Learning*, 2024.

[24] Huihan Liu, Shivin Dass, Roberto Martín-Martín, and Yuke Zhu. Model-based runtime monitoring with interactive imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[25] Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-task interactive robot fleet learning with visual world models. In *8th Annual Conference on Robot Learning*, 2024.

[26] Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. In *Conference on Robot Learning*, pages 3468–3484. PMLR, 2023.

[27] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *Conference on Robot Learning (CoRL)*, 2024.

[28] Andrew Y Ng et al. Algorithms for inverse reinforcement learning.

[29] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam,

Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2023.

[31] Nathan D Ratliff, David Silver, and J Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27:25–53, 2009.

[32] Juntao Ren, Gokul Swamy, Zhiwei Steven Wu, J Andrew Bagnell, and Sanjiban Choudhury. Hybrid inverse reinforcement learning. *arXiv preprint arXiv:2402.08848*, 2024.

[33] Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. In *Robotics: Science and Systems*, 2024.

[34] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[35] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[36] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8579–8590, 2023.

[37] Nur Muhammad Mahi. Shafiullah, Siyuan. Feng, Lerrel. Pinto, and Russ. Tedrake. Supervised policy learning for real robots, July 2024. URL https://supervised-robot-learning.github.io. Tutorial presented at the Robotics: Science and Systems (RSS), Delft.

[38] Shai Shalev-Shwartz and Amnon Shashua. On the sample complexity of end-to-end training vs. semantic abstraction training. *arXiv preprint arXiv:1604.06915*, 2016.

[39] Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Ed Schmerling, and Marco Pavone. Real-time anomaly detection and reactive planning with large language models. In *Robotics: Science and Systems*, 2024.

[40] 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. *arXiv preprint arXiv:2210.06718*, 2022.

[41] Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pages 10022–10032. PMLR, 2021.

[42] Gokul Swamy, David Wu, Sanjiban Choudhury, Drew Bagnell, and Steven Wu. Inverse reinforcement learning without reinforcement learning. In *International Conference on Machine Learning*, pages 33299–33318. PMLR, 2023.

[43] Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025.

[44] Open X-Embodiment Team. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[45] Anirudh Vemula, Yuda Song, Aarti Singh, Drew Bagnell, and Sanjiban Choudhury. The virtues of laziness in model-based rl: A unified objective and algorithms. In *International Conference on Machine Learning*, pages 34978–35005. PMLR, 2023.

[46] Joseph A. Vincent, Haruki Nishimura, Masha Itkina, Paarth Shah, Mac Schwager, and Thomas Kollar. How generalizable is my behavior cloning policy? a statistical approach to trustworthy performance evaluation. *IEEE Robotics and Automation Letters*, 9(10):8619–8626, 2024. doi: 10.1109/LRA.2024.3445635.

[47] Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Perez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions. *arXiv preprint arXiv:2411.16627*, 2024.

[48] Zihan Wang, Brian Liang, Varad Dhat, Zander Brumbaugh, Nick Walker, Ranjay Krishna, and Maya Cakmak. I can tell what i am doing: Toward real-world natural language grounding of robot experiences. In *8th Annual Conference on Robot Learning*, 2024.

[49] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Inter-

active videogpts are scalable world models. In *Advances in Neural Information Processing Systems*, 2024.

[50] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

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

[52] Brian D Ziebart, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. 2008.

APPENDIX

A. Related Work

Generative Imitation-Based Policies. With the rise of large-scale open-source datasets of expert demonstrations [2, 4, 5, 8, 13, 44], imitation learning (IL) has become a popular way to learn low-level robot control policies from data. In particular, recent advances in generative modeling have unlocked policy architectures that can model diverse, multi-modal behaviors directly from high-dimensional observations [6, 23, 33]. At the same time, generative IL policies still exhibit nuanced, hard-to-anticipate performance degradations during deployment time. These degradations range from complete task failures (e.g., inability to grasp a cup, knocking it down, or dropping it [27]) potentially due to distribution shifts or visual distractors [19, 46], to inappropriate behaviors that are misaligned with the deployment context or an end-user's preferences (e.g., placing the gripper inside of a cup filled with water during grasping) [1]. In this work, our goal is to leverage the diverse low-level behavior distribution that the base policy has learned, but prevent these nuanced performance degradations at runtime via our novel policy steering method.

Failure Detection, Monitoring & Prediction. The handling of generative policy failures can be grouped into three broad categories: *posthoc* detection, *runtime* monitoring, and failure *prediction*. *Posthoc* approaches identify and explain failures present in offline robot datasets, and have recently leveraged Vision Language Models (VLMs) to accelerate this process via video captioning, highlighting critical data frames, and providing human-interpretable summaries of failures [10, 16, 26, 48]. In contrast, *runtime* monitoring aims to detect failures as they happen during robot deployment. To quickly identify nuanced failures, recent methods propose a "fast and slow" approach: a fast online detector flags unusual situations (e.g., binary anomaly classifier), while a slower VLM-based reasoner provides a deeper understanding of the event and if it is a relevant failure [1, 39]. Although these strategies can effectively identify failures, they fundamentally require the robot to start failing for the runtime monitor to activate. The final category, failure *prediction* methods, anticipate failures before they occur and unlock the potential for preemptive correction of the base policy. Here, existing approaches [22, 24, 25] often rely on out-of-distribution (OOD) detection in a latent space or dense human labels to train a binary classifier that distinguishes failures from successes. In this work, we contribute to the *predictive* category of methods. Our method anticipates future outcomes of the policy's actions via a latent world model, and reasons about the outcomes via a VLM that is aligned with the predicted latent states.

Policy Steering. A traditional method to improve a base IL policy is to fine-tune it with additional intervention data [25] or recovery behaviors [37]. However, recently, runtime policy steering has become an attractive alternative to improving a generalist IL policy [27, 47] without needing any additional and expensive demonstration data. Runtime policy steering assumes that the base policy is capable of generating the correct behaviors, but fails to select them reliably. Here, an external verifier can be used to re-rank (i.e., "steer") the generations towards ones with good outcomes. Previous methods have explored humans-in-the-loop [47] or a Q-function learned from very large offline datasets labeled with sparse rewards. [27] as the verifier. In our framework, by using a VLM-in-the-loop as our verifier, we can perform policy steering autonomously after finetuning VLM on a small dataset and provide human-like interpretable guidance.

Learning to Search. From an algorithmic perspective, our approach fits within the paradigm of *learning to search* (L2S) [31]. In L2S, one learns the components required to, at test time, plan a sequence of actions, rather than merely imitating what was in the training data. L2S provides two key advantages over direct imitation algorithms like behavioral cloning. First, the agent gains the ability to reason about the consequences of its actions and potentially recover from mistakes, avoiding *compounding errors* [35, 41]. We see this manifest in our system's ability to avoid or correct from failures the base policy would have produced otherwise. Second, *verifying* whether a plan is good is often easier to learn than *generating* a good plan in the first place [43, 28]. We see this manifest in the fact that our system only requires a limited amount of data to fine-tune our verifier VLM, rather than the larger amount of data an end-to-end approach would have required [38].

In contrast to more classical L2S approaches that require extensive *global* search in the real world [31, 52], we instead perform *local* search [3] against a learned verifier [42, 12] inside a learned world model [32]. This allows us to avoid the computational expense of global search and the potential safety violations incurred by real-world interaction. As argued by [42, 32], doing so still matches many of the guarantees of classical L2S methods if one fits the world model on a mixture of on-policy and off-policy trajectories (i.e., in a *hybrid* fashion [40, 45]), as we do consistently across all of our experiments.

B. Limitations

While **FOREWARN** exhibits strong policy steering across diverse task settings, it is not without limitations. First, it assumes base policy is sufficiently competent—i.e., already containing the correct behavior. Future work should investigate how to detect if none of the policy's generated action plans are suitable for the deployment context, and how to improve the base policy via targeted fine-tuning data.

A second limitation lies in modeling real-world interaction dynamics. Our component-level analysis in App. D2 revealed that our system's primary failures stem from the world model's imprecise "imagination", exacerbated by our limited training data. More advanced visual features (e.g., DINO features [30]) might improve the world model's robustness to visual distractors. Ongoing research on large-scale, generalizable world models [49] for manipulation may also inform future extension of this framework.

Another bottleneck of the current system is the inference overhead, a common challenge for large autoregressive models

like LLMs and world models. Some approaches to this include: 1) hierarchical decomposition of *FOREWARN* running high-level reasoning at low frequency while maintaining low-level control at high frequency like the common practice of hierarchical Vision Language Action Models (VLAs), 2) using advancement in quantization and caching techniques, 3) distilling our latent-aligned VLM into a model with smaller size, 4) strategically allocate inference time by identifying keypoints when it is "worth" reasoning for longer or deliberately.

Finally, our VLM is built upon an open-source LLama-3.2-11B-Vision-Instruct model, whose visual reasoning capabilities lag behind its language-based commonsense reasoning. A stronger backbone could yield higher-quality behavior descriptions, especially if it excels at video captioning tasks.

C. Algorithm & Implementation Details

1) Base Policy:

We use Diffusion Policy [6] as our robot action generation model due to its strong ability to capture multimodal and complex robot behaviors.

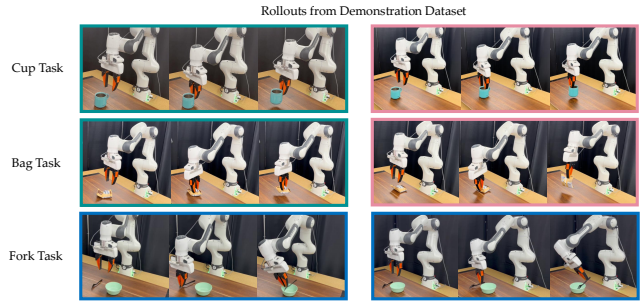

Fig. 4: **Multimodality in Demonstration Datasets.**

Training dataset. For each task, we collect 100 multimodal demonstrations and train the policy on an NVIDIA A6000 GPU following the procedure in [6]. In **Cup Task**,the training dataset consists of 50 demonstrations where the robot grasps the cup by the rim while touching the inner surface, and 50 demonstrations where the robot grasps it by the handle. In **Bag Task**, the training dataset consists of 50 demonstrations where the robot grasps the middle part of the bag and 50 demonstrations where it grasps the top edge without crushing the chips. In **Fork Task**, the training dataset consists of 4 modes, including 1) picking up the fork by tines and dropping high; 2) picking up the fork by the handle and dropping high; 3) picking up the fork by the tines and dropping low; 4) picking up the fork by the handle and dropping low. Each mode has 25 demonstrations. Figure 4 presents multimodal demonstrations for each task, and Table II details the hyperparameters used for training.

Observation and action spaces. The robot's policy receives inputs from a wrist-mounted camera, a third-person camera, and proprioceptive states—including the end-effector position, orientation quaternion relative to the base, and gripper opening—to predict actions that control the end-effector's absolute pose and gripper opening.

Hyperparameter	Value
State Normalization	Yes
Action Normalization	Yes
Image Chunk	2
Image Size	256
State Dimension	8
Action Dimension	10
Action Execution Horizon (Bag)	140
Action Execution Horizon (Cup)	120
Prediction Horizon (Bag)	120
Prediction Horizon (Cup)	140
Batch Size	100
Training Epochs	600
Learning Rate	1e-5
Number of Worker	16
Train Diffusion Step	100
Inference Diffusion Iteration	16

TABLE II: **Base Policy Hyperparameters.**

Action plan aggregation. During policy steering, we aggregate 100 sampled action plans into 6 modes. We represent each action trajectory as a T $\times$ 7 array, where T is the trajectory length, and each action consists of the gripper's position (3D coordinates) and orientation (quaternion with four components). To cluster these trajectories into 6 modes, we apply Time Series K-Means with Dynamic Time Warping (DTW) as the distance metric. This allows us to group similar motion patterns while accounting for temporal variations in trajectory execution. The average trajectory within each cluster is used as the aggregated action plan. In the subsequent stages of our system, these 6 aggregated action plans serve as candidate options for policy steering selection.

2) World Model:

Motivation. The effectiveness of world models has been demonstrated across various embodied domains [25, 50]. In our problem setting, it provides several key advantages: 1) it grounds low-level actions, difficult for a VLM to interpret—by predicting future image observations from action plans, bridging the gap between low-level actions and visual observations; 2) it compresses information into latent states that not only retain essential details for high-quality image decoding but also effectively predict the next latent state given an action.

Architecture. We use DreamerV3, a state-of-the-art recurrent world model from [18]. Our world model, denoted as $\mathcal{W}_\phi = (\mathcal{E}_\phi, f_\phi, \mathcal{D}_\phi)$, consists of three key components: an encoder network $\mathcal{E}_\phi$, a recurrent dynamics model f_ϕ that operates over a stochastic continuous latent space, and a decoder network $\mathcal{D}_\phi$ that projects latent embeddings back into observations.

Below, we provide a detailed definition of each module:

$$z_\tau := \begin{cases} \mathcal{E}_\phi(o^i_\tau), & \tau = t. \\ \mathcal{E}_\phi(o^i_\tau, z^i_{\tau-1}, a^i_{\tau-1}), & t < \tau < t+T. \end{cases} \tag{4}$$

$$\hat{z}^i_{\tau+1} \sim \begin{cases} f_\phi(z^i_\tau, a^i_\tau), & \tau = t. \\ f_\phi(\hat{z}^i_\tau, a^i_\tau), & t < \tau < t+T. \end{cases} \tag{5}$$

$$\hat{o}^i_\tau := \mathcal{D}_\phi(\hat{z}^i_\tau), \quad t \le \tau < t+T. \tag{6}$$

Hyperparameter	Value
Observation Normalization	Yes
Action Normalization	Yes
Image Size	(64, 64)
Batch Length	64
Batch Size	16
Training Step (Cup)	100000
Training Step (Bag)	150000
Hidden State h Dimension	512
Stochastic Representation z Dimension	32
Dynamics Loss Ratio α_{dyn}	0.5
Representation Loss Ratio α_{rep}	0.1
Reconstruction Loss Ratio α_{pred}	1.0
CNN Encoder Depth	32
CNN Encoder Kernel Size	4

TABLE III: **World Model Hyperparameters.**

Training. The world model $\mathcal{W}_\phi$ is pretrained using an offline dataset of policy's rollouts and demonstrations $\mathbb{D}_{\text{WM}} = \{\{(o^j_\tau, a^j_\tau, o^j_{\tau+1})\}_{\tau=0}^{H-1}\}_{j=1}^{M}$. Apart from 100 demonstrations, we collect 250 rollouts for each task, including both the success and failure experiences of the generative policy π deployed on the real robot.

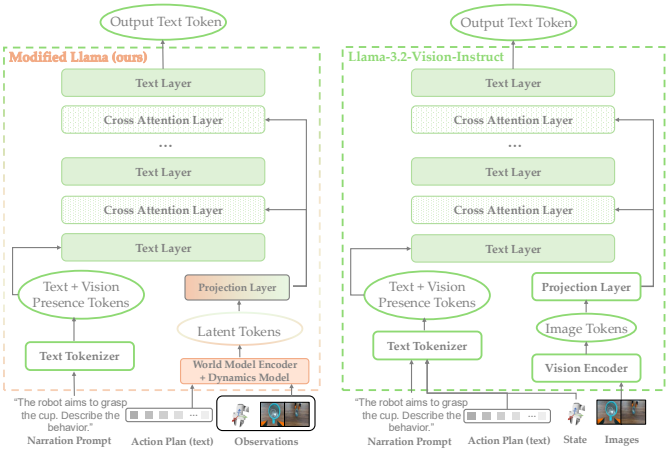

Fig. 5: **VLM Architecture**. On the left is detailed architecture **FOREWARN**, an modified VLM with an explicit world model. On the right is original Llama-3.2-Vision-Instruct model, which is used as our baseline **VLM-Act**. This is an elaborated version of Fig. 2

Since offline datasets lack reward signals, we omit the reward loss. Instead, the pretraining of the world model is supervised using a modified loss function: $\mathcal{L}_\mathcal{W} = \alpha_{dyn} \times \mathcal{L}_{dyn} + \alpha_{rep} \times \mathcal{L}_{rep} + \alpha_{pred} \times \mathcal{L}_{pred}$. Here, the dynamic loss $\mathcal{L}_{dyn}$ incentivizes accurate forward predictions in the latent space while $\mathcal{L}_{rep}$ ensures that latent states are informative for reconstructing observations and learning good representation. Finally, the prediction loss $\mathcal{L}_{pred}$ minimizes the error between decoded observations of the predicted latent states and the ground-truth observations. The exact loss calculation is shown in [18]. The hyperparameters used for training world model is shown in Table III.

During training, the decoder $\mathcal{D}_\phi$ reconstructs observations from the latent space to compute the prediction loss and the visualization helps us select the best world model.

Hyperparameter	Value
LoRA (rank)	8
LoRA (dropout)	0.05
LoRA (alpha)	32
Precision	bfloat16
Batch Size	10
Learning Rate	1e-4
Epoch (Cup)	10
Epoch (Bag)	15
Finetuned Layers	["down_proj", "up_proj", "o_proj", "k_proj", "q_proj", "v_proj", "gate_proj", "linear_proj"]

TABLE IV: **VLM Hyperparameters of FOREWARN**

Deployment. Because we cannot reset the real-world environment to the exact same state, each action sequence $\mathbf{a}^i_t$ yields only a single observation sample $\mathbf{o}^i_t$. As a result, the expectation in Eq. 3 is approximated in practice. During finetuning, we make the dynamics model f_ϕ deterministic by taking the mean of its predicted distribution. Although the model provides a 64-step prediction horizon, we downsample these future latent states to 16 to reduce redundancy from minimal changes across adjacent steps. Each latent state is formed by concatenating the hidden state and the stochastic representation, and we keep the world model frozen throughout VLM finetuning and deployment.

You are an assistant for monitoring the robot's behavior.

Given the provided task description input and the images in time order of robot execution(combined wrist and third person view), respond with only one sentence that best describes the robot's behavior.

Your response must strictly align with the image observations. Do not include any additional text, explanations, or information.

Input: {general task description}

<latent token> x 16

Fig. 6: **Prompt Template for Behavior Narration in FOREWARN.**

3) Vision Language Model:

Architecture. We use Llama-3.2-11B-Vision-Instruct model as our VLM backbone. We modify the original Llama Model to incorporate the explicit world model to predict outcomes of the action plans first and then use VLM to reason about

the latent states to generate behavior narrations. The modified architecture is visualized in Fig. 5. Specifically, we replace the original vision encoder (ViT) of Llama with our world model's encoder as well as dynamics model, and project latent states as text tokens. We use one single linear layer to project latent tokens to text tokens.

You are an assistant providing help for household tasks. Based on the task description and the possible modes provided, your job is to select the best behavior mode that fulfills the task goal. The most important rule that needs to be considered first is in task description. Assume that each mode will be executed exactly as described, with no modifications.

Output Requirements:
- If neither mode fully meets the task goal or all modes present a high risk, you reject all of them.
- If exactly one mode is suitable, you choose this one.
- If multiple modes are equally suitable and do not conflict with constraints, you can choose any of them.

Response Format:
- Return the selected behavior mode by repeating the mode description. Use `none of them` for mode description if no mode is selected.
- Include one sentence of explanation for the choice made.

Task Description:{general task description}

Task Condition: {task context}

Behavior Modes:
`Mode 1. {Action Plan 1}`
`Mode 2. {Action Plan 2}`
`Mode 3. {Action Plan 3}`
`Mode 4. {Action Plan 4}`
`Mode 5. {Action Plan 5}`
`Mode 6. {Action Plan 6}`

Fig. 7: **Prompt Template for Policy Steering.**

Finetuning. We adopt LoRA to finetune our modified model, loading the base weights from Llama-3.2-Vision-Instruct and randomly initializing the new linear projection layer. This approach updates only 0.2664% of the original model's parameters. The finetuning hyperparameters are listed in Table IV, and we select the model with the lowest evaluation loss for deployment.

You are an assistant responsible for failure monitor during household tasks. Your task is to evaluate whether the policy's behavior is a success or failure based on the task description and task condition. The most important rule is in task condition. Assume the robot policy is going to be executed as described, except there is a clear failure stated in behavior mode. Do not infer unintended consequences or failure scenarios beyond what is explicitly described.

Output Requirements: - If the behavior mode is likely to achieve the desired goal in the task description, you confirm it is normal. Remember only when the behavior mode indicated failure given task condition or clearly against the task goal, you output abnormal.

Response Format:
- Return either `normal` or `abnormal` as the final judgment. Provide a one-sentence reason for the decision made, without speculating on additional failure scenarios. Your estimation should reason only with task condition. Don't be overconservative.

Provided Behavior Mode: `Mode 1. {Action Plan}`.

Task Condition:{task context}

Task Description:{general task description}

Fig. 8: **Prompt Template for Failure Monitoring.**

Prompt. The modified VLM is finetuned to generate behavior narration with the prompt template in Fig. 6, explaining the predicted latent states from the world model.

In policy steering. we sample and aggregate action sequences into 6 modes to query VLM for the best choice. Each mode's behavior narration replaces Action Plan in the prompt template in Fig. 7.

Another potential usage of our system is to evaluate and monitor different policies' performances under specific task description before execution. To showcase our system can be a reliable failure monitor across different tasks, we conduct additional experiments in App. D2. We list the prompts used for failure monitor in Fig. 8.

Different tasks have different task descriptions, and specifications (i.e., contexts), which are put in the prompt for policy monitoring and policy steering can be found in Sec. IV

You are an assistant for monitoring the robot's behavior. Given the provided task description input, the image of robot at the first step(combined wrist and third person view), the robot gripper state at the first step and the next 16 actions of the robot, respond with only one sentence that best describes the robot's future behavior of these 16 actions.

Note:
1. Remember to pay attention to the details of grasping part of the object, e.g. handle, rim. If gripper is not contacting the cup or the cup is lying down on the table, describe the behavior and must state it as a failure in the output.
2. Both the robot state and the actions have 8 dimensions. The first three dimensions are the x, y, z positions of the robot gripper. X direction is moving forward and backward in the image and Y is moving left and right. Z is moving up and down. The 4th to 7th values are the quaternion of the gripper. The last value is gripper opening or closing.
3. Your response must strictly align with the image observation. Do not include any additional text, explanations, or information.

Action Sequences: {Action Plan}

Current State: {proprioceptive states}

Input:{general task description}

Fig. 9: **Prompt Template for Behavior Narration for VLM-Act in Cup Task**

Hyperparameter	Value
LoRA (rank)	8
LoRA (dropout)	0.05
LoRA (alpha)	32
Quantization	4bit
Batch Size	1
Learning Rate	1e-5
Epoch (Cup)	10
Epoch (Bag)	15
Finetuned Layers	["down_proj", "up_proj", "o_proj", "k_proj", "q_proj", "v_proj", "gate_proj"]

TABLE V: VLM Hyperparameters of **VLM-Act**.

4) Additional Details of Baselines:

VLM-Act. This baseline is an ablated version of **FOREWARN** without the explicit world model. It uses the original Llama-3.2-11B-Vision-Instruct model as shown in right part of Fig. 5 and finetuned with the same labels as in **VQA Dataset**. The action plans and states are prompted as text and image observations are concatenated together and processed as image tokens. Details of the prompt are available in Fig. 9. Since VLM cannot directly interpret the low-level action control, we give some privileged information in the prompt, marked as red in the prompt template, to help it generate behavior narration. However, this baseline still struggles to generate accurate behavior narration as shown in Sec. D2, despite being finetuned on the same dataset. For policy monitoring and policy steering, **VLM-Act** uses the same prompt as **FOREWARN**. Hyperparamers used for finetuning are shown in Table. V

VLM-Img & VLM-Img-Oracle. We further evaluate an advanced VLM (GPT-4o) to interpret fine-grained motion details from a sequence of 16 images, either reconstructed from predicted latent states or recorded from actual execution. GPT-4o runs at the default temperature. As shown by the red text in Fig. 10, GPT-4o still struggles to comprehend subtle motion details, even when given privileged information to guide its focus.

Classifier-Dyn-Latent. For policy steering and monitoring, we adopt a causal transformer-based binary classifier [25] to

You are a helpful assistant that describes robot behaviors.

Task Description: {general task description}

Here are sixteen images(third person view and wrist camera view) of robot execution listed in time order and a task description. Please generate a one-sentence description of the robot's behavior.

Notes:
1. Pay attention to the grasping process as it shows the grip part.
2. Do not hallucinate over the contact position.
3. If the grasp is achieved, the sentence should focus on which part of the object it grasps and makes contact with, e.g. handle, inner surface, etc.
4. 4. If the cup is not grasped in the robot's gripper, the sentence should describe the failure.

<image token> x 16

Fig. 10: **Prompt Template for Behavior Narration for GPT-4o in Cup Task.**

Hyperparameter	Value
Embedding Dimension	64
Number of Head	1
Attention Dropout	0.05
Embedding Dropout	0.05
Block Output Dropout	0.05
Context Length	16
Sinusoidal Embedding	True
Learning Rate	1e-4
Gradient Clip	$(-\infty, 100]$
Epochs	30

TABLE VI: Hyperparameters of **Classifier-Dyn-Latent**.

directly predict success or failure from future latent states. Table VI summarizes its hyperparameters. We manually label each rollout as success or failure under the first task description to train the classifier.

VLM-DynLat-Binary. This end-to-end baseline also predicts a binary success/failure label from future latent states, but employs the same modified VLM architecture as **FOREWARN**, which provides a larger capacity than the transformer-based classifier. It is trained on the same dataset as **Classifier-Dyn-Latent**, with the prompt template shown in Fig. 11.

You are an assistant for failure detection. Analyze the robot behaviors from the provided images presented in time order (combined wrist and third-person view). Compare the observed behaviors with the provided task description and task condition to determine whether the task was successful or a failure.

Respond with only one of the following numbers: 0 or 1. 0 means task failure. 1 means task success.

No additional words, explanations, or clarifications are allowed. Your output must be limited to a single digit (0 or 1) and nothing else.

Task Description: {general task description}

Task Condition:{task context}

<latent token> x 16

Fig. 11: **Prompt Template for Policy Monitoring for VLM-DynLat-Binary.**

VLM-DynLat-Category. Similarly, we develop an end-to-end approach that directly predicts a valid set of action plans for policy steering. The modified VLM is finetuned to output which action-plan indexes are valid under the current task description. Its input includes a text prompt and six sequences of predicted future latent states (Fig. 12), each corresponding to one candidate action plan.

D. Experiment

1) Real Robot Setup:

Fig. 13 demonstrates the setup of our real-world experiments.

You are an assistant for policy steering.

The first behavior mode is observed as the following images: <latent token> x 16.
The second behavior mode is observed as the following images: <latent token> x 16.
The third behavior mode is observed as the following images: <latent token> x 16.
The fourth behavior mode is observed as the following images: <latent token> x 16.
The fifth behavior mode is observed as the following images: <latent token> x 16.
The sixth behavior mode is observed as the following images: <latent token> x 16.

Analyze six sequences of robot behaviors from the provided images presented in time order (combined wrist and third-person view). Compare each sequence with the provided task description and task condition to determine which sequences successfully satisfy the task requirements.

Instructions:
-Respond with a list containing the indices of the valid sequences (e.g., [0,1,2]).
-If none of the sequences satisfy the task requirements, return an empty list: [].
-The indices should be between 0 and 5.
-No additional words, explanations, or clarifications are allowed.
-Your output must be strictly a list containing the indices of the valid sequences or an empty list.

Task Description:{general task description}

Task Condition:{task context}

Fig. 12: **Prompt Template for Policy Steering for VLM-DynLat-Category.**

We employ two cameras, a RealSense D435 camera on the Franka hand and a Zed mini 2i camera placed in front of the robot. In order to increase the contact region and compliancy, we replace the original Franka gripper finger with 3D printed gripper finger from [7].

Fig. 13: **Real Robot Setup.** The visualization of the real robot environment and the positions of the cameras.

2) Supplementary Experiments & Analyses:

From Action Rollouts to Behavior Narration.

As discussed in Sec. III-B, if we want to use the VLM as an open world verifier $R_\psi^{\text{VLM}}(\cdot;\ell)$, we need to enable the model to understand the underlying textual representation of low-level action outcomes. In this section, we study if our our latent-aligned VLM can accurately describe the outcomes of low-level actions. We also compare our approach with several baselines to investigate the advantages of using an explicit world model for predicting action outcomes and decoding a robot's action plans into behavior narrations.

Baselines. We compare our approach, **FOREWARN**, against four baselines (more implementation details provided in App. C4). (1) **FOREWARN-Oracle**, is an upper-bound on

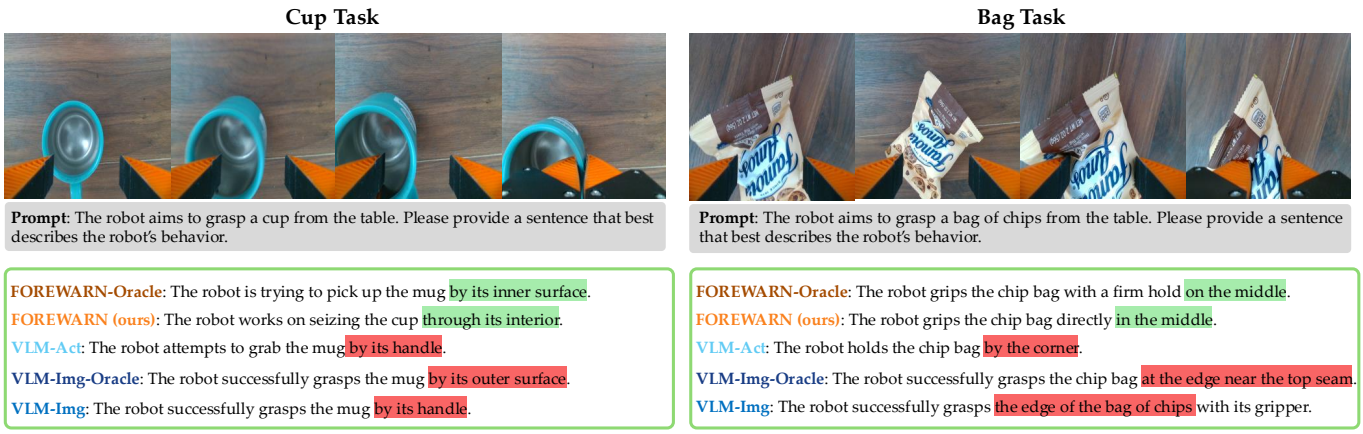

Fig. 14: **Examples of Behavior Narrations Predicted by Each Approach.** The top row displays the ground-truth robot observations and the prompt used for querying VLMs. Only **FOREWARN** and **FOREWARN-Oracle** consistently produce accurate outcome narrations, effectively capturing nuanced motion details. In contrast, the baselines frequently hallucinate or fail to capture critical contact details between the gripper and objects. For instance, in the **Bag** task, **VLM-Act**, **VLM-Img**, and **VLM-Img-Oracle** all hallucinate that the robot is grasping the edge of the bag, whereas it is actually grasping the middle.

our method's performance assuming that we had access to *ground-truth* future observations (instead of relying on the latent dynamics f_ϕ to predict future outcomes). This method uses the encoder $\mathcal{E}_\phi$ on ground-truth future observations to get privileged (posterior) future latent states $z_{t:t+T}$ as input for the VLM. (2) **VLM-Act**, which directly fine-tunes the original Llama-3.2-11B-Vision-Instruct model to generate behavior narrations end-to-end from the current observation o_t and an action plan $a_{t:t+T}$ (represented as text), without explicitly predicting outcomes. (3) **VLM-Img**, which utilizes the *decoded* world model's predictions (i.e., the predicted future visual observations) given a robot's planned actions. We use GPT-4o [29] to process the predicted visual observations and generate behavior narrations in a zero-shot manner. (4) **VLM-Img-Oracle**, which is similar to **VLM-Img** but is an upper bound on this method by using ground-truth visual observations instead of predicted ones.

Metrics. We adopt the metrics from [10] to evaluate the alignment between predicted behavior narrations and ground-truth narrations: (1) **LLM Score**: A similarity score (ranging from 0 to 1) determined by the GPT-4o model. (2) **GT Accuracy**: A binary score (0 or 1) indicating whether the predictions match the ground-truth narrations, as determined by a human labeler (in this case, the authors). For further details on the motivation behind using these metrics for evaluation, please refer to App. D3.

Results: On the Value of Explicit Action Outcome Prediction. Table VII presents the **GT Accuracy** and **LLM Score** for our approach and each baseline. The results are averaged across 30 test rollouts for each task. The results show that **VLM-Act** performs poorly, achieving less than 50% **GT Accuracy** across all tasks. This underperformance is due to its inability to interpret low-level actions without the grounding provided by a world model's future outcome predictions. In contrast, **FOREWARN**, which leverages an explicit world model, outperforms **VLM-Act** by over 50% on every task, despite both being fine-tuned on the same dataset. These results demonstrate that decoupling the VLM's burden of predicting action outcomes enables the model to produce more accurate outcome narrations than directly training the VLM to both predict outcomes and generate narrations end-to-end.

	GT Accuracy ↑			LLM Score ↑		
	Cup	Bag	Average	Cup	Bag	Average
FOREWARN-Oracle	0.92±0.02	0.77±0.03	0.85±0.03	0.86±0.01	0.76±0.03	0.81±0.02
FOREWARN (Ours)	**0.87±0.02**	**0.75±0.03**	**0.82±0.03**	**0.82±0.02**	**0.72± 0.02**	**0.76±0.02**
VLM-Act	0.37±0.03	0.36±0.06	0.37±0.05	0.52±0.05	0.50±0.07	0.51±0.06
VLM-Img-Oracle	0.61±0.04	0.43±0.03	0.52±0.04	0.65±0.02	0.62±0.02	0.64±0.02
VLM-Img	0.36±0.05	0.33±0.04	0.35±0.05	0.56±0.03	0.60±0.04	0.58±0.04

TABLE VII: **Alignment Between Predicted Behavior Narrations and Ground-Truth Narrations.** **FOREWARN** outperforms all baselines across both tasks and achieves performance comparable to **FOREWARN-Oracle**, which has access to ground-truth action outcomes and represents the upper bound for our approach. We use 50 rollouts to evaluate the performance. For **FOREWARN**, **FOREWARN-Oracle** and **VLM-Act**, the mean and standard deviation are reported by running 3 seeds for the finetuning experiments while **VLM-Img** and **VLM-Img-Oracle**, report 3 queries of GPT-4o.

Results: On the Value of VLM Fine-Tuning. Interestingly, despite being among the most advanced VLMs, both **VLM-Img** and **VLM-Img-Oracle** struggle to accurately interpret robot behaviors directly from visual observations, even with access to ground-truth observations. As shown in Table VII, these methods fall behind **FOREWARN** by at least 30% in **GT Accuracy** and 16% in **LLM Score**. These results show that existing state-of-the-art VLMs struggle to decode fine-grained motion details from video observations, underscoring the importance of fine-tuning for improved performance in such tasks.

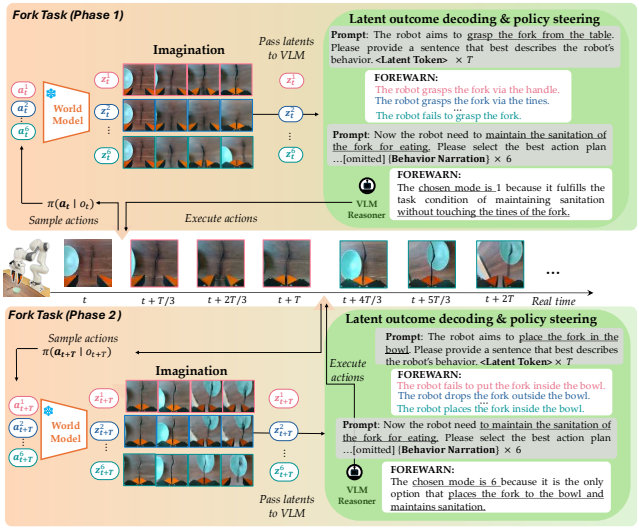

Fig. 15: **Policy Steering: Fork Task.** We visualize the steering process for the **Fork** task including two phases (Pick and Place). For each phase, we visualize the imagined T-step rollouts decoded from the world model for the 3 out of 6 action plans sampled from the base policy on the left. On the right, we show the behavior narrations generated from our finetuned VLM $\mathcal{T}_\psi^{\text{VLM}}$ and the VLM's reasoning $R_\psi^{\text{VLM}}(\cdot;\ell)$ about the outcomes based on the task description ℓ and behavior narrations to select the best action plan to execute. The time axis shows the real-world execution of the selected behavior from the perspective of the wrist-camera.

Results: Qualitative Examples. In Figure 14, we visualize behavior narrations generated by our approach and the baselines. FOREWARN consistently produces more accurate outcome narrations, effectively capturing nuanced motion details. In contrast, the baselines often hallucinate or fail to capture critical contact details between the gripper and objects.

More Qualitative Examples for Policy Steering. We include additional qualitative examples for **Cup** and **Bag** tasks in Fig 16. These examples further demonstrate the effectiveness of our policy steering system for different tasks. The imagined image sequences from the world model show that our system is capable to predict the various outcomes for different action sequences and the VLM can reason about behavior narrations and correctly evaluate the action plans.

FOREWARN as a VLM-in-the-loop Failure Monitoring System. In this section, we present another application of our approach—preemptive failure monitoring—based on the behavior narrations generated in Sec. D2. Similarly, our modified VLM first decodes the predicted latent states from the world model, as behavior narrations $\hat{\ell}_b$. Then it reasons about behavior narrations under task description ℓ and decides whether the future action plan, translated as $\hat{\ell}_b$, is a failure. As both quantitative and qualitative results demonstrate, FOREWARN is a reliable and versatile failure monitoring framework across diverse task.

Baselines. We consider three methods as our baselines, which preemptively predict the outcome of the action plan before the execution. 1) VLM-DynLat-Binary takes the predicted latent states from the world model and task description ℓ as input to the VLM, directly generating binary output to indicate success or failure without the intermediate step of behavior narration; 2) Classifier-Dyn-Latent takes the predicted latent states as input and trains a transformer-based binary classifier to generate binary output with the same dataset as VLM-DynLat-Binary; 3) VLM-Act uses generated behavior narrations in Sec. D2 and queries the VLM again to decide if the behavior is a success or failure within the context of the task description ℓ. These baselines are equivalent to those in Sec. IV-A.

Metrics. To evaluate the overall performance of different methods as preemptive failure monitors, we report the standard detection metrics from prior work [1] including *Accuracy (ACC)*, *True Positive Rate (TPR)*, *True Negative Rate (TNR)*.

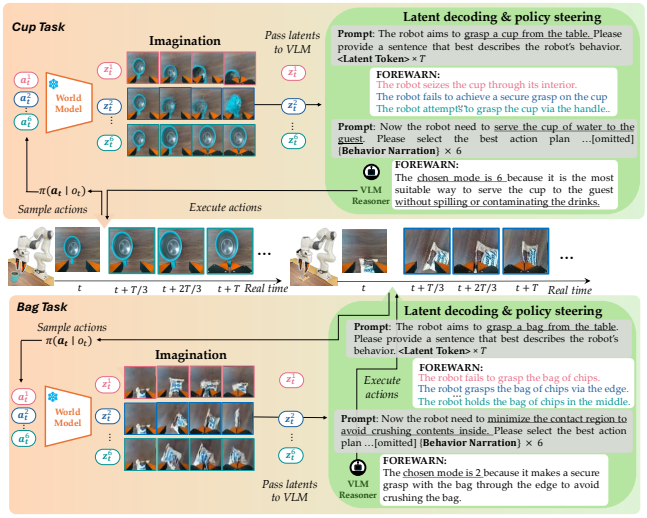

Fig. 16: **Policy Steering for Cup and Bag Task.** We visualize the steering process for the **Cup** task on the top and the **Bag** task on the bottom. For each task, we visualize the imagined T-step rollouts decoded from the world model for the 3 out of 6 action plans sampled from the base policy on the left. On the right, we show the behavior narrations generated from our finetuned VLM $\mathcal{T}_\psi^{\text{VLM}}$ and the VLM's reasoning $R_\psi^{\text{VLM}}(\cdot;\ell)$ about the outcomes based on the task description ℓ and behavior narrations to select the best action plan to execute. The time axis shows the real-world execution of the selected behavior from the perspective of the wrist-camera.

Results: Failure Monitoring Performance. Both VLM-DynLat-Binary and Classifier-Dyn-Latent perform well when evaluation task description is the same as training. However, their performances drop sharply in novel task description, indicating poor generalization of end-to-end model even though they have the same model capacity as our method. In contrast, FOREWARN consistently attains high accuracy,

Method	Cup						Bag						Average					
	Training			Unseen			Training			Unseen			Training			Unseen		
	Acc↑	TPR↑	TNR↑	Acc↑	TPR↑	TNR↑	Acc↑	TPR↑	TNR↑	Acc↑	TPR↑	TNR↑	Acc↑	TPR↑	TNR↑	Acc↑	TPR↑	TNR↑
FOREWARN (Ours)	**0.90**	**0.80**	**1.00**	**0.75**	**0.70**	0.80	0.75	0.71	0.77	**0.75**	**0.75**	0.75	0.83	0.76	0.89	**0.75**	**0.73**	0.78
VLM-DynLat-Binary	0.85	0.77	0.91	0.15	0.11	0.27	0.75	**0.86**	0.69	0.40	0.38	0.42	0.80	**0.82**	0.80	0.28	0.25	0.35
Classifier-Dyn-Latent	**0.90**	**0.80**	**1.00**	0.10	0.10	0.10	**0.80**	0.71	0.85	0.35	0.13	0.50	**0.85**	0.76	**0.93**	0.23	0.12	0.30
VLM-Act	0.65	0.75	0.58	0.50	0.10	**0.90**	0.60	0.00	**0.92**	0.65	0.13	**1.00**	0.63	0.38	0.75	0.58	0.12	**0.95**

TABLE VIII: **Policy Monitoring.** The reported result in the table is averaged over 20 trajectories. FOREWARN, VLM-DynLat-Binary and Classifier-Dyn-Latent perform similarly well in training task description while VLM-Act has poor performance. In unseen task description, FOREWARN is the only method that maintains similar performance as training task description.

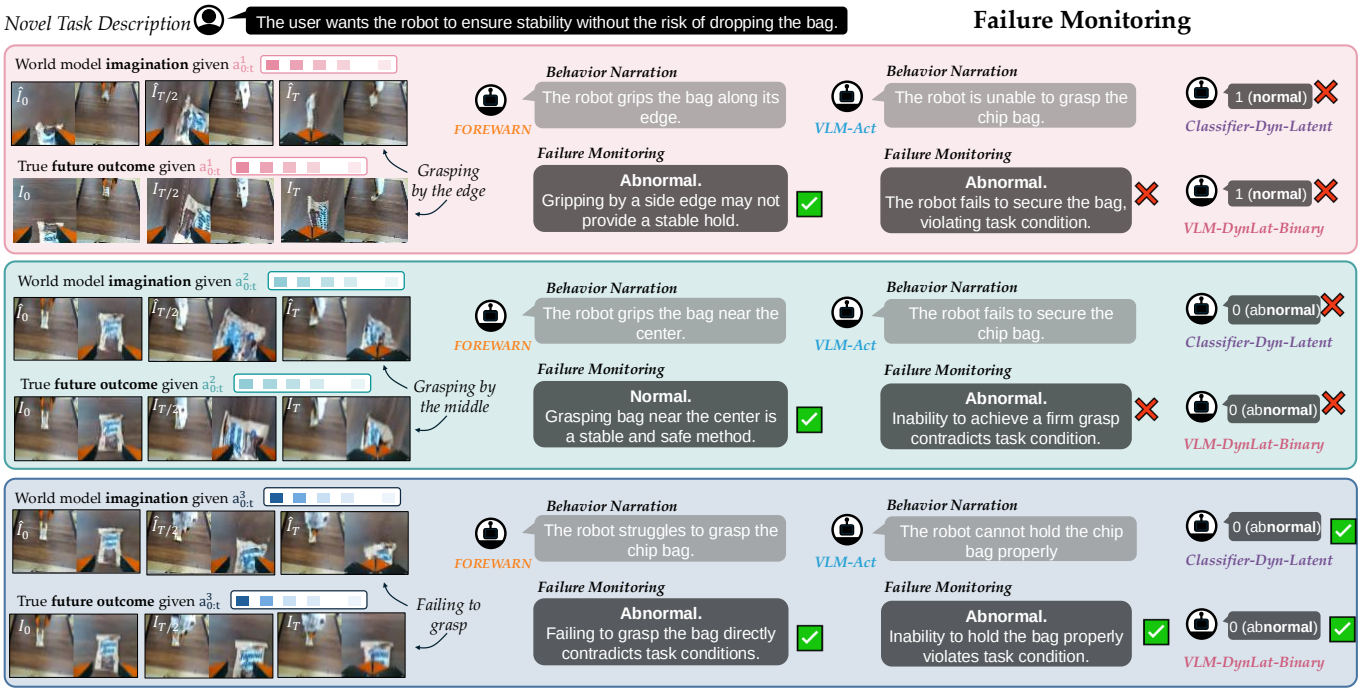

Fig. 17: **Policy Monitoring.** In the top (pink) and medium (green) row, the robot imagines correctly about the robot behaviors but only FOREWARN describes the behavior correctly and generates correct monitoring results with adequate explanations. In the bottom row (blue), all of the methods distinguish failure correctly from success.

Method	Accuracy ↑																	
	Independent						Training Task Description						Novel Task Description					
	World Model			Behavior Narration			Reasoning			Overall System			Reasoning			Overall System		
	Cup	Bag	Average	Cup	Bag	Average	Cup	Bag	Average	Cup	Bag	Average	Cup	Bag	Average	Cup	Bag	Average
FOREWARN				**0.90**	**0.70**	**0.80**	**1.00**	0.85	0.93	**0.90**	0.75	0.83	**0.90**	0.85	0.88	**0.75**	**0.75**	**0.75**
VLM-DynLat-Binary	0.80	0.75	0.78	-	-	-	0.95	0.80	0.88	0.85	0.75	0.80	0.20	0.45	0.33	0.15	0.40	0.28
Classifier-Dyn-Latent				-	-	-	**1.00**	0.90	**0.95**	**0.90**	**0.80**	**0.85**	0.05	0.25	0.15	0.10	0.35	0.23
VLM-Act	-	-	-	0.35	0.35	0.35	0.95	**0.95**	**0.95**	0.65	0.60	0.63	**0.90**	**1.00**	**0.95**	0.50	0.65	0.58

TABLE IX: **Performance Breakdown** for each component in our pipeline across four methods. FOREWARN have high accuracy across all components while VLM-Act has very low accuracy in narration, leading the poor performance of the overall system. both Classifier-Dyn-Latent and VLM-DynLat-Binary performs well in training task description and drop sharply in novel task scenarios.

higher than 75%, with balanced TPR and TNR across all tasks given different task descriptions, demonstrating its reliability and flexibility as a failure monitor.

The generalization capability of our method comes from decoupling the problem of failure monitoring as behavior narration and outcome evaluation.

This is also demonstrated by **VLM-Act**, which has the same intermediate step and shows balanced performance in both task descriptions, but **VLM-Act** often misclassifies behaviors in *Bag Task* as failures, resulting in low TPR and lower overall accuracy than **FOREWARN**.

In Fig. 17, we demonstrate monitoring results for all three different behaviors qualitatively. Across all three modes of behaviors, **FOREWARN** consistently generates accurate descriptions as well as correct monitoring results. **VLM-Act** is biased to generate failure narrations across all three modes, leading to wrong monitoring results. **Classifier-Dyn-Latent** and **VLM-DynLat-Binary** completely do not understand different action plans within different task descriptions. They generate the same monitoring results for totally different descriptions, contradictory to the actual execution.

Component-level Analysis for the System. We analyze each component in our method as well as all the baselines in (Table IX). **FOREWARN** shows high accuracy across all the components while the poor performance of **VLM-Act** is from the low-quality of narration directly generated from low-level action sequences. After finetuning, both methods preserve the strong reasoning capability of the VLM. However, **Classifier-Dyn-Latent** and **VLM-DynLat-Binary** has a huge performance drop in novel task scenarios because they are overfitting to the specific task description in the training.

3) Metric Ablations:

Metrics for Behavior Narration. We investigate four common text-generation metrics proposed in prior work [10]: *Cosine Similarity*, *ROUGE-L*, *LLM Fuzzy Matching*, and *Binary Success Rate*. To assess each metric's correlation with ground-truth labels, we sample 16 narrations for each of three behaviors in the **Cup Task** (grasping by handle, grasping by rim, and grasp failures), yielding 360 intra-category and 768 inter-category comparisons. As shown in Figures. 18,19, and20, it is difficult for Cosine Similarity and ROUGE-L to cleanly distinguish narrations from the same category versus different categories, whereas *LLM Fuzzy Matching* with GPT-4o can easily separate them. This discrepancy arises because the narrations share similar high-level semantics (e.g., "grasping the cup") and differ only in fine-grained details (e.g., grasp location). Consequently, we adopt **LLM Score** and **Ground-Truth Accuracy** (manual matching) as our final metrics for evaluating generated behavior.

Results: Policy Steering Speed. Our system queries the VLM twice to first generate behavior narrations and then select the best action plan. The overall inference time is 3.7 seconds among which the generation of behavior narrations

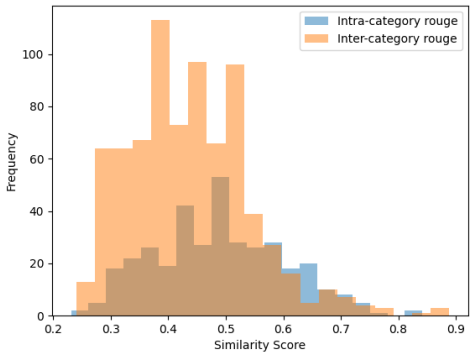

Fig. 18: **Distribution of ROUGE-L Score** shows that intra-catego s.

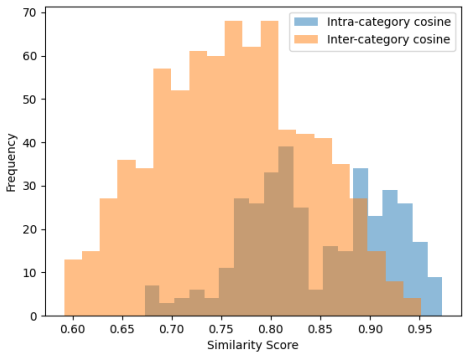

Fig. 19: **Distribution of Cosine Similarity Score** shows that intra-c ones.

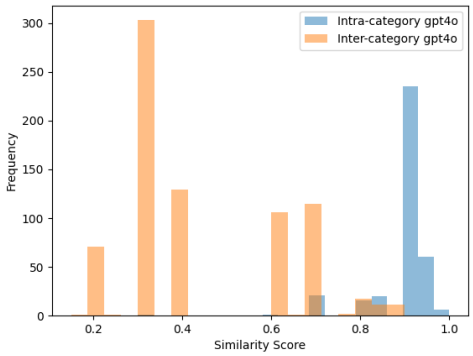

Fig. 20: **Distribution of LLM Score** shows inter-category and inter-category scores can be roughly separated at 0.7.

for 6 candidate action plans takes 1.3 seconds. In comparison, **VLM-Act** takes 22.0 seconds in total for VLM inference and behavior narration runs 19.7 seconds, much slower than **FOREWARN**, partially due to the usage of token per patch in images rather than a single token per state (image) like **FOREWARN**. Additionally, our world model and VLM communicate directly in latent space, avoiding image decoding from world model and encoding from VLM, which could further speeds up inference.

Method	Time (Seconds) ↓			
	World Model Prediction	Behavior Narration	Evaluation	Total
FOREWARN	0.1	1.3	2.3	3.7
VLM-Act	-	19.7		22.0

TABLE X: **Inference time for the Policy Steering System.** Inference time for each component in the system (averaged across 3 runs) shows that **FOREWARN** greatly reduces the time to generate behavior narrations from our modified VLM compared to **VLM-Act**.