Steering Multirobot Behavior via Closed-Loop Affine Activation Editing

Anonymous Authors

***Abstract*— Real-world robots need to adapt their behavior beyond the envelope of their pre-trained policy. Policy finetuning or retraining are options, but they risk catastrophic forgetting, degrading the pretrained policy's base performance. To combat this, we introduce CLAE: *Closed-Loop Affine Activation Editing*, an inference-time framework for steering the behavior of a frozen policy by editing intermediate activations while keeping the base policy weights and downstream action head untouched. CLAE approaches behavior steering as a closed-loop problem whose outputs edit policy activations that adapt online to the robot state, environment, target behavior, and multi-robot context. It trains a sparse autoencoder over frozen-policy activations, selects behavior-relevant latent features via post-hoc probing, and learns a lightweight RL-based steering policy that applies state-dependent affine edits to selected latents during inference. We validate CLAE on a frozen multi-quadrotor navigation policy trained to perform a single task: navigating robots to a set of goal locations while avoiding obstacles. Through extensive simulations and physical tests, we show that while navigating to their goal positions, CLAE can 1. steer individual robot behavior by controlling each robot's velocity profile; 2. coordinate multirobot behavior by preserving a desired formation, and 3. produce entirely new behavior wherein robots are required to reduce their exposure to surveillance cameras in the environment.**

I. Introduction

Learned robot policies are commonly obtained via imitation learning on robot demonstrations, reinforcement learning from environment interactions, or hybrid pipelines that combine large-scale robot data, pretrained vision-language models, and task-specific supervision. When deployed, a trained policy expresses a set of behaviors (a "behavioral envelope") from its training data, objective, conditioning interface, and deployment assumptions. In practice, robots need to modify or extend their behavioral envelope when deployed: a mobile robot or manipulator trained for aggressive navigation or goal reaching may need to behave more cautiously around people, a legged robot trained for energy-efficient navigation may need to trade efficiency for speed, and a team of flying robots trained to navigate to a goal may need to track a new velocity profile, maintain a formation, or avoid certain regions.

Fine-tuning or retraining may adapt a policy to such requirements, but it changes the base policy weights, risking catastrophic forgetting and necessitates revalidation of the policy. These concerns become more pronounced as the community moves towards generalist pretrained models, where maintaining separate fine-tuned variants for every behavioral preference adds deployment and revalidation cost. The multirobot setting amplifies this challenge: the correct adaptation for each robot depends not only on its own state but also on teammates' states. These considerations motivate our central question: can we modify/expand the inference-time behavioral envelope of a trained robot policy while keeping the base policy fixed?

We study this problem as *post-training behavior steering*: modifying the closed-loop behavior of a trained policy at inference time while aiming to retain its base competence. Recent inference-time steering work [1] makes a similar point for pretrained robot policies under spatial or semantic shifts: failures may reflect the absence of a mechanism for selectively adapting existing skills at test time, rather than the absence of the underlying motor skill itself. Existing post-training steering methods [1]–[6] typically act around the policy output, by modifying prompts, command interfaces, sampling trajectories, latent noise, candidate actions, or action selection. In contrast, activation steering modifies behavior by editing intermediate representations within the frozen policy. Prior work in language models [7]–[10] shows that internal activations can be modified to steer outputs. Recent robotics work [11]–[13] extends this idea to robot policies, demonstrating scalar, direction-based, or safety-specific interventions. However, these methods do not formulate activation editing as closed-loop behavior steering, in which edit parameters vary online with the robot's state, environment, target behavior, and other robots.

We propose *closed-loop affine activation editing* (CLAE), a framework for steering a frozen robot policy by editing intermediate activations rather than updating weights or adding an action-residual controller. At each policy step, we encode an intermediate activation block into a sparse autoencoder (SAE) basis, apply affine edits to a small set of behavior-relevant SAE latents, and insert the decoded edited activation back into the frozen policy. A lightweight steering policy predicts the affine edit parameters online; the base policy's frozen downstream layers produce the robot action.

We evaluate CLAE on a frozen multi-quadrotor navigation policy across three behavior-steering tasks: velocity-profile tracking, multi-robot formation control, and camera-aware stealth navigation. Together, these test whether CLAE can steer robot-level motion, coordinate team behavior, and inject deployment-time behavior without making any modifications to the underlying policy. CLAE is the first framework for inference-time, closed-loop behavior steering of a multirobot policy through learned affine activation editing. Although we validate the framework on a multirobot policy learned using reinforcement learning, CLAE only requires white-box (knowledge of architecture and weights) access to intermediate activations; therefore it is possible to steer arbitrary architectures, behaviors, and robot morphologies. Our hope is that CLAE can then be applied to behaviorally steer policies outside of our provided experiments.

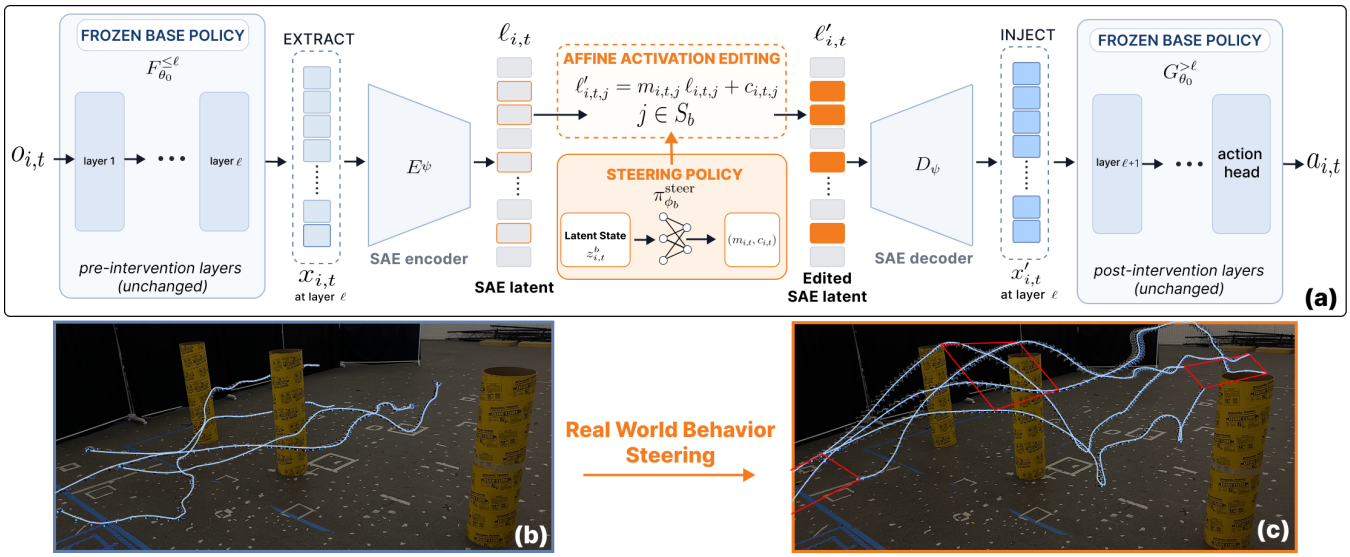

Fig. 1: **Closed-Loop Affine Activation Editing (CLAE). (a)** An intermediate activation $x_{i,t}$ from the frozen base policy is encoded into SAE latents $\ell_{i,t}$. The steering policy $\pi^{\text{steer}}_{\phi_b}$ outputs $(m_{i,t}, c_{i,t})$, which applies the affine edit $\ell'_{i,t,j} = m_{i,t,j}\ell_{i,t,j} + c_{i,t,j}$ on the select latent set $\mathcal{S}_b$, while unselected latents pass through unchanged. The edited latents are decoded and inserted back into the frozen policy. Bottom panels show real-world deployment of four quadrotors on the multi-robot formation task. In **(b)**, the base policy navigates independently, dispersing around obstacles. In **(c)**, CLAE edits activations to steer the quadrotors toward the formation geometry (red overlays), balancing the effort to stay in formation with the core collision-avoidance and goal-reaching behavior of the underlying policy.

Our contributions are:

1) We introduce *closed-loop affine activation editing* (CLAE), a post-training behavior-steering framework that modifies intermediate activations at inference time while preserving the base policy weights.
2) We build a sparse activation-editing interface by training an SAE on frozen-policy activations and using post-hoc behavior probes to select a compact set of behavior-relevant SAE latents. A lightweight RL steering policy then outputs state-dependent affine edits (multiplicative gains and additive offsets) over those latents at each policy step.
3) We evaluate CLAE on a frozen multi-quadrotor navigation policy across three behaviors spanning robot-level motion, coordinated multi-robot behavior, and deployment-time behavior injection: velocity-profile tracking, multi-robot formation control, and stealth navigation, with validation in large-scale simulation and hardware experiments.

II. Related Work

Post-training steering of frozen robot policies. A growing body of work studies deployment-time adaptation of frozen robot policies. Value-guided policy steering reranks candidate actions using an offline value function [2]; inference-time policy steering biases generative-policy sampling with human interaction signals [3]; and diffusion steering optimizes the latent-noise space of frozen diffusion policies with reinforcement learning [4]. Recent methods also guide frozen diffusion or flow policies with VLM-derived rewards [1], select candidate plans using predicted outcomes and VLM reasoning [5], or improve controllability by training VLAs to follow richer command abstractions [6]. A separate line of work leaves the policy weights untouched and instead learns a corrective term in action space [14, 15]. These methods demonstrate the importance of deployment-time steering, but primarily act through prompts, commands, sampling, latent noise, candidate actions, action selection, or action residuals. In contrast, we steer behavior by editing intermediate activations inside the frozen policy.

Activation and representation steering. Activation steering provides a complementary mechanism for modifying model behavior without weight updates. In language models, activation engineering modifies intermediate activations at inference time [7], while representation engineering studies internal representations as objects that can be monitored and manipulated [8]. Sparse autoencoders provide sparse feature bases for analyzing and intervening on dense activations [9, 10], and representation surgery studies feature transformations beyond additive directions [16]. Our work brings these representation-intervention ideas to robot policies, where activation edits affect not only the current action but also the future observations the edited policy induces.

Activation editing for robot behavior steering. Recent robotics work shows that internal activations of robot policies can be causally linked to behavior. Mechanistic interpretability methods identify semantic directions such as speed and direction in VLA models and steer them with activation interventions [11]; SAE-based VLA analyses show that sparse features can influence closed-loop robot behavior through decoder-direction interventions [12]; feature-controllability methods use linear probes and minimal raw-representation interventions to steer VLA outputs toward desired feature values [17]; and Latent Activation Editing modifies activations of a frozen multirobot navigation policy to improve safety [13]. While these methods enable inference-

time activation intervention, in some cases adaptively, none learns an activation-editing policy from behavior-specific return. CLAE instead learns a closed-loop editing policy that produces state-dependent affine edits conditioned on robot state, environment, neighboring agents, and target behavior, while keeping the base policy weights fixed.

III. Method

We propose *closed-loop affine activation editing* (CLAE), an inference-time mechanism for steering a frozen multirobot policy π_{θ_0}(Figure 1a). At each policy step, we intercept an intermediate activation of the frozen base policy π_{θ_0} and encode it into a sparse autoencoder (SAE) basis, apply an affine edit to a small target-behavior-specific subset of SAE latents, decode the edited latent, and insert it back into the frozen policy. The base policy parameters θ_0 remain fixed throughout: we do not fine-tune the policy, modify its architecture, or train an action-residual controller. For each target behavior b, rollout-derived metrics select a behavior-relevant SAE latent set $\mathscr{S}_b$, and a behavior-specific reward trains a lightweight steering policy $\pi^{\text{steer}}_{\phi_b}$ that chooses affine edits online. The base policy, SAE, and steering policy are decentralized and shared across all robots.

A. Frozen-Policy Activation Interface

For a given intervention layer ℓ, we view the frozen base policy as a composition of the computation before ℓ ($F^{\le\ell}_{\theta_0}$) and after ℓ ($G^{>\ell}_{\theta_0}$). Therefore, the base policy can be written as: $\pi_{\theta_0} = G^{>\ell}_{\theta_0} \circ F^{\le\ell}_{\theta_0}$. Given the robot observation $o_{i,t}$, the frozen computation up to layer ℓ produces the intervention activation $x_{i,t}$, and the remaining frozen layers map this activation to the robot action:

$$x_{i,t} = F^{\le\ell}_{\theta_0}(o_{i,t}), \qquad u_{i,t} = G^{>\ell}_{\theta_0}(x_{i,t}). \tag{1}$$

We replace $x_{i,t}$ with edited activation $x'_{i,t}$ before the frozen downstream policy is evaluated: $u_{i,t} = G^{>\ell}_{\theta_0}(x'_{i,t})$. The steering policy does not output a motor command or an action residual $\Delta u_{i,t}$; it acts only by modifying selected SAE latents inside the frozen base policy.

B. Sparse Activation Basis and Behavior Latent Selection

Editing the dense activation $x_{i,t} \in \mathbb{R}^d$ directly would require the steering policy to jointly control all d of its dimensions (which could be large). Instead, we move to a sparse SAE basis where a few behavior-relevant latents can be edited in isolation while the rest pass through unchanged. We train an SAE on activations $x_{i,t}$ collected from rollouts of the frozen base policy. The encoder E_ψ maps each activation to a sparse latent vector, and the linear decoder D_ψ reconstructs it back:

$$\ell_{i,t} = E_\psi(x_{i,t}), \qquad \hat{x}_{i,t} = D_\psi(\ell_{i,t}) = W_D \ell_{i,t} + b_D, \qquad \ell_{i,t} \in \mathbb{R}^{K_{\text{SAE}}}. \tag{2}$$

The SAE is trained post-hoc using the standard reconstruction-plus-sparsity objective used for sparse feature decomposition of neural activations [9, 18]:

$$\mathscr{L}_{\text{SAE}} = \mathbb{E}_x \left[\|x - D_\psi(E_\psi(x))\|_2^2 + \lambda_{\text{sparse}} \|E_\psi(x)\|_1 \right]. \tag{3}$$

After training the SAE, we use post-hoc probing to select a compact set of behavior-relevant latents for the steering policy to edit. This builds on standard representation probing [19]–[21] and SAE feature decomposition [9, 10]. For each target behavior b, rollout-derived metrics serve as probes. We score latents using a split-stable absolute Pearson score $S^{(b)}_j = \text{mean}_s |\rho^{(b)}_{j,s}| - \text{std}_s |\rho^{(b)}_{j,s}|$, where $\rho^{(b)}_{j,s}$ is the correlation between latent j and the metric on trajectory split s. For multiple metrics, $S^{(b)}_j$ combines these scores. We select the editable latent set as $\mathscr{S}_b = \text{TopK}_j S^{(b)}_j$. The SAE basis is behavior-agnostic, while $\mathscr{S}_b$ is behavior-specific: velocity probes use the velocity components, formation probes use relational square-geometry quality, and stealth probes use trajectory-modulation metrics such as lateral motion, velocity, and clearance, since camera avoidance is not directly represented in the frozen policy's activations.

C. Closed-Loop Affine Activation Editing

Given the selected latent set $\mathscr{S}_b$, we learn a behavior-specific steering policy $\pi^{\text{steer}}_{\phi_b}$. At each time step, the steering policy provides an affine edit, $\xi_{i,t} = (m_{i,t}, c_{i,t}) \in \mathbb{R}^{2|\mathscr{S}_b|}$, where $m_{i,t}, c_{i,t} \in \mathbb{R}^{|\mathscr{S}_b|}$ are its multiplicative and additive components respectively. This edit is applied directly within the SAE latent basis before decoding:

$$\ell'_{i,t,j} = \begin{cases} m_{i,t,j}\ell_{i,t,j} + c_{i,t,j}, & j \in \mathscr{S}_b, \\ \ell_{i,t,j}, & j \notin \mathscr{S}_b. \end{cases} \tag{4}$$

Thus, only the behavior-selected SAE latents are modified; all unselected SAE latents are copied unchanged. The full edited SAE latent is decoded to obtain the edited activation, $x'_{i,t} = D_\psi(\ell'_{i,t})$, which is then inserted into the frozen policy activation. When $m_{i,t} = \mathbf{1}$ and $c_{i,t} = \mathbf{0}$, the edited latent reduces to its SAE reconstruction $D_\psi(E_\psi(x_{i,t}))$.

D. Learning Closed-Loop Affine Activation Edits

The affine edit $\xi_{i,t}$ is not observed as a supervised target: its effect is only revealed after the edited activation is decoded, passed through the frozen policy, executed as a robot action, and propagated through the multi-agent simulator. Because useful edits depend on the current robot, environment, and team state, we learn them with closed-loop reinforcement learning. The affine edit $\xi_{i,t} = (m_{i,t}, c_{i,t})$ is the action of the steering policy. This induces a closed-loop transition over the simulator state through the fixed base policy and fixed SAE, $s_{t+1} \sim P_{\theta_0,\psi}(\cdot \mid s_t, \xi_{1:N,t})$. Here, s_t is the simulator state and $\xi_{1:N,t}$ are the affine edits for all robots. The transition kernel $P_{\theta_0,\psi}$ denotes the closed-loop rollout dynamics induced by the simulator, the frozen base policy π_{θ_0}, the frozen SAE (E_ψ, D_ψ), and the activation-editing interface from Sec. III-C. We do not learn this transition model; the steering policy interacts with it through rollouts. Only the steering-policy parameters ϕ_b are optimized. The steering policy maximizes the expected discounted return

$$J_b(\phi_b) = \mathbb{E}_{\tau \sim (\pi^{\text{steer}}_{\phi_b}, P_{\theta_0,\psi})} \left[\sum_{t=0}^{T} \gamma^t \frac{1}{N} \sum_{i=1}^{N} r^{(b)}_{i,t} \right], \qquad \theta_0, \psi \text{ fixed}. \tag{5}$$

The expectation is over closed-loop rollouts of the steering policy. The base policy parameters θ_0 and SAE parameters ψ remain fixed during both steering-policy training and evaluation. The steering observation contains a task-specific head and a common latent-control tail containing the selected latents and the previous activation edit. All target behaviors use the same activation-editing mechanism but different steering observations and rewards. We use the common reward form

$$r_{i,t}^{(b)} = r_{\text{task},i,t}^{(b)} + w_{\text{stab}}^{(b)} r_{i,t}^{\text{stab}} - \lambda_{\text{crash}}^{(b)} C_{i,t} \tag{6}$$

Here, $r_{\text{task},i,t}^{(b)}$ is the behavior-specific steering objective, $C_{i,t}$ is a weighted crash/contact score, and $r_{i,t}^{\text{stab}} = -1[C_{i,t} > 0]$ penalizes unstable or failed states. We optimize Eq. (5) with PPO, using a Beta-parameterized policy [22] that produces bounded affine edits.

IV. Experimental Results

A. Experimental Setup

Simulator & Base Policy: We use the QuadSwarm [23] simulator with 8 quadrotors navigating a $10 \times 10 \times 10$ m room with static cylindrical obstacles. We adopt the end-to-end, decentralized RL policy [24] as the base controller. Each robot observes its state and goal, the relative states of its two nearest neighbors, and a 3×3 signed-distance field of nearby obstacles. Three separate MLPs encode these observations. Neighbor and obstacle embeddings are fused via multi-head attention, concatenated with the self embedding, and passed through downstream MLPs to an action head outputting four normalized rotor thrusts.

Intervention point: CLAE requires only white-box intermediate activation access; the SAE trains on the intercepted block, and probing selects behavior-relevant latents. We intercept the first fused activation $x_{i,t}$ post-attention, as it is the earliest representation combining self, neighbor, and obstacle information. All activation-editing baselines and ablations use this same activation, so observed differences reflect the editing mechanism rather than layer choice.

B. Behaviors

We steer the pretrained multi-robot navigation policy towards the following behaviors: **Velocity-profile tracking.** Each robot is assigned a reference velocity profile $v_{i,t}^{\star}$ and must track it while still reaching its goal and avoiding obstacles. The reference is generated by a minimum snap planner [25], independent of the learned policy and made feasible in the obstacle-rich environment using a CBF-based [26] filter. **Multi-robot formation control.** The robots are partitioned into two four-robot groups, and each group is assigned a target formation. We use a square with desired side length $s^{\star} = 0.5$ m; other formations can be specified by changing the reference distance pattern. **Camera-aware stealth navigation.** We introduce surveillance cameras that are *absent* from the base policy's original training. Each environment contains three cameras with surveillance radius 2.5 m, placed to cover large portions of the free space. The steering policy receives camera-risk features and must reduce exposure while preserving goal reaching and collision avoidance.

Method	SAE	RL-optimized intervention	State-conditioned edit	Intervention rule
Base Policy	—	—	—	$x'_t = x_t$
DSRL (distilled diffusion policy) [4]	—	✓	✓	$a_k = D(s_k, w_k)$, $w_k \sim \pi^{\text{noise}}(\cdot \mid s_k)$
Residual Action RL [14]	—	✓	✓	$a_t = u_t + \Delta a_t(s_t)$
Fixed Scalar SAE Steering [12]	✓	—	—	$x'_t = x_t + \alpha d_j$
Linear Probe Intervention [17]	—	—	✓	$x'_t = x_t + \text{clip}_\eta(u_t)$
Closed-Loop Multiplicative Editing	✓	✓	✓	$\ell'_{S,t} = m_t \odot \ell_{S,t}$
CLAE (ours)	✓	✓	✓	$\ell'_{S,t} = m_t \odot \ell_{S,t} + c_t$

TABLE I: **Compared methods.** All steering methods keep their respective base policies fixed and differ in where and how the intervention is produced.

All behaviors use the reward form in Eq. (6), with only the task reward changing across behaviors:

$$\begin{aligned} r_{\text{task},i,t}^{\text{vel}} &= w_{\text{track}} r_{i,t}^{\text{track}}, \qquad r_{\text{task},i,t}^{\text{form}} = w_{\text{shape}} r_{i,t}^{\text{shape}} + w_{\text{goal}} r_{i,t}^{\text{goal}} \\ r_{\text{task},i,t}^{\text{stealth}} &= w_{\text{zone}} r_{i,t}^{\text{zone}} + w_{\text{margin}} r_{i,t}^{\text{margin}} + w_{\text{goal}}^{\text{stealth}} r_{i,t}^{\text{goal}} \end{aligned} \tag{7}$$

Here $r_{i,t}^{\text{track}}$ is the negative tracking error against the reference velocity, $r_{i,t}^{\text{shape}}$ penalizes deviation from the target square geometry, and $r_{i,t}^{\text{goal}}$ penalizes normalized distance to the assigned goal. The stealth terms combine a binary penalty $r_{i,t}^{\text{zone}}$ for entering a camera's valid surveillance footprint with a soft margin $r_{i,t}^{\text{margin}}$ that grows as a robot approaches a camera center, giving PPO a dense signal around an otherwise sparse objective; exposure remains the primary stealth objective, with $w_{\text{goal}}^{\text{stealth}}$ small and serving only to bias the edited policy toward goal reaching. Each steering policy conditions on the selected SAE latents and its previous activation edit, together with a compact task-specific head: velocity tracking receives reference-tracking information; formation receives teammate-relative information; and stealth receives local surveillance-camera information. All steering inputs are available at inference time from the robot state estimate, the task specification, and the frozen-policy activation stream.

C. Baselines and Evaluation Protocol

Baselines. We compare CLAE against the frozen base policy, two non-activation interventions, two activation-editing baselines, and a multiplicative-only ablation (Table I). All steering methods keep their respective base policies fixed; the activation-editing methods additionally share the same intervention layer and frozen downstream policy. **DSRL** [4] is included as an auxiliary diffusion-policy comparison. Since our base controller has no diffusion-noise interface, we train a diffusion policy [27] by behavior cloning on base policy rollouts, freeze it, and adapt DSRL to steer its initial noise for each target behavior. Because behavior cloning can incur compounding errors under policy-induced distribution shift [28, 29], a relevant consideration in our closed-loop multi-robot setting, we report both the unsteered and steered diffusion policies to distinguish the pre-steering performance gap from steering gains. **Residual Action RL** [14, 15] learns a bounded corrective term on the frozen base policy's output using the same steering observations, reward, RL algorithm, and environment-step budget as CLAE, isolating intervention in action space from intervention in activation space. Since

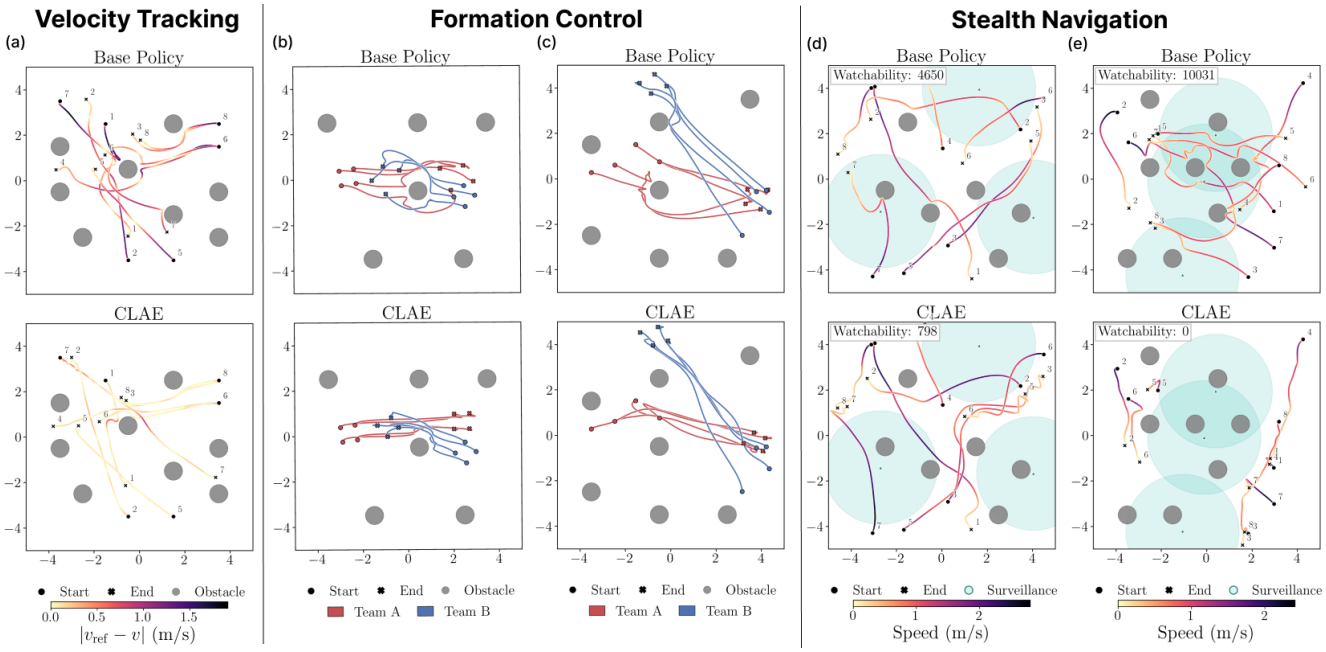

Fig. 2: **Qualitative behavior: CLAE versus the frozen base policy.** Trajectory color denotes velocity-tracking error $|v - v^\star|$ in (a), team membership in (b,c), and speed in (d,e). In (b), both the start and goal configurations are square formations; in (c), robots start from an arbitrary configuration and form the target square. (d,e) Stealth navigation. Teal areas denote surveillance areas that must be avoided.

existing robotic learning activation-steering methods are not designed for closed-loop multi-robot behavior steering, we adapt the closest single-robot VLA-style interventions: **Fixed Scalar SAE Steering** [12], which shifts the activation along one SAE decoder direction, and **Linear Probe Intervention** [17], which fits a probe offline and applies a closed-form perturbation, so its edit varies with state but is not optimized from closed-loop return. We evaluate these on velocity tracking and formation control but not stealth navigation, since the stealth objective is exogenous to the frozen policy's training objective and is not directly represented in its activations. We also evaluate **Closed-Loop Multiplicative Editing**, which removes the additive offset from Eq. (4) and applies only $\ell'_{S,t} = m_t \odot \ell_{S,t}$, isolating the contribution of the affine edit.

Metrics. *Behavior-specific:* **Velocity Error (Vel. Err.)** (m/s) is the mean absolute component-wise error between each robot's velocity and its reference, $|v - v^\star|$, averaged over axes, robots, and timesteps; $|e_x|, |e_y|, |e_z|$ are its per-axis components. **Formation Error (Form. Err.)** measures how closely each four-robot group maintains the target square formation, as the mean relative deviation of the group's six pairwise XY distances from their target values in the square (four sides and two diagonals); 0 corresponds to perfect target geometry. **Watchability** is the total number of robot-camera visibility events per episode, summed over robots, cameras, and timesteps; 0 means no robot was ever visible to any camera. *General:* **Collision-Free Rate (CF)** (%) is the fraction of episodes with no robot-robot or robot-obstacle collisions; **Success Rate (SR)** is the fraction of robots that reach the goal safely, defined as ending within 0.8 m of the assigned goal without being involved in a collision; **Goal Distance (Goal Dist.)** (m) is the mean final distance between each robot and its assigned goal.

D. Results

Table II summarizes the quantitative behavior-steering results and Figure 2 shows the corresponding qualitative behavior changes in representative rollouts. For each behavior, results for all methods are averaged over the same set of 1,000 distinct environment configurations, each generated using a different random seed.

Velocity-profile tracking. CLAE reduces the mean velocity-tracking error from 0.322 m/s (base policy) to 0.085 m/s, a $\sim$3.8$\times$ improvement, with consistent per-axis gains on $|e_x|$, $|e_y|$, and $|e_z|$ (Table IIa). The qualitative result in Figure 2a confirms the effect at the trajectory level: CLAE trajectories, coloured by tracking residual $|v - v^\star|$, remain uniformly pale, whereas the base-policy trajectories show large residuals throughout. Figure 3 shows the per-axis velocity traces for two representative robots: CLAE follows the reference profile closely on v_x, v_y, and v_z, while the base policy produces its own goal-driven velocity profile that diverges sharply from the reference. Residual Action RL, acting on the same frozen base policy, reduces tracking error to 0.269, still far above CLAE's 0.085. The activation-editing baselines improve tracking marginally, while the multiplicative-only variant remains close to the base policy, supporting the benefit of state-dependent affine edits for tracking a time-varying reference.

Multi-robot formation control. CLAE reduces formation error from 0.280 (base policy) to 0.086, a $\sim$3.2$\times$ improvement, while maintaining 98.5% collision-free episodes and 99.4% robot-success (Table IIb). Figure 2b and 2c show two representative configurations. In (b), where the start and goal are both square-structured, the base policy lets the two four-robot groups drift apart near the obstacle, while CLAE preserves a tight square formation geometry during

Method	*Behavior-specific* ↓				*General*		
(a) Velocity-Profile Tracking	**Vel. Err.**	$\lvert e_x \rvert$	$\lvert e_y \rvert$	$\lvert e_z \rvert$	CF (%) ↑	SR ↑	Goal (m) ↓
Base Policy	0.322	0.221	0.212	0.149	99.0	0.888	0.457
Distilled Diffusion Policy (unsteered)	0.340	0.234	0.223	0.156	48.0	0.817	0.496
DSRL (distilled diffusion policy) [4]	0.298	0.218	0.216	0.147	18.7	0.476	0.847
Residual Action RL [14]	0.269	0.172	0.174	0.080	87.7	0.001	3.170
Fixed Scalar SAE Steering [12]	0.305	0.200	0.189	0.134	99.0	0.882	0.559
Linear Probe Intervention [17]	0.294	0.189	0.185	0.139	99.0	0.836	0.593
Closed-Loop Multiplicative Editing	0.318	0.214	0.219	0.134	98.8	0.861	0.501
CLAE (ours)	**0.085**	**0.075**	**0.057**	**0.039**	98.7	0.899	0.456
(b) Multi-Robot Formation Control	**Form. Err.**				CF (%) ↑	SR ↑	Goal (m) ↓
Base Policy	0.280				99.5	0.997	0.271
Distilled Diffusion Policy (unsteered)	0.297				79.6	0.793	0.293
DSRL (distilled diffusion policy) [4]	0.241				70.2	0.682	0.428
Residual Action RL [14]	0.247				94.6	0.944	0.498
Fixed Scalar SAE Steering [12]	0.466				22.3	0.000	3.713
Linear Probe Intervention [17]	0.315				95.8	0.972	0.483
Closed-Loop Multiplicative Editing	0.276				99.2	0.989	0.242
CLAE (ours)	**0.086**				98.5	0.994	0.281
(c) Camera-Aware Stealth Navigation	**Watchability**				CF (%) ↑	SR ↑	Goal (m) ↓
Base Policy	3596.6				99.1	0.924	0.530
Distilled Diffusion Policy (unsteered)	3755.6				27.3	0.245	0.560
DSRL (distilled diffusion policy) [4]	2614.2				12.4	0.101	0.733
Residual Action RL [14]	3636.3				43.7	0.284	0.939
Closed-Loop Multiplicative Editing	3453.2				97.0	0.897	0.604
CLAE (ours)	**1151.3**				96.5	0.912	0.552

TABLE II: **Behavior-steering results across three tasks.** For each behavior, all methods are averaged over the same 1000 distinct environment configurations. Activation-editing and residual-action methods act on the base RL policy, whose weights remain fixed during steering. The unsteered diffusion policy and DSRL rows use a separately distilled diffusion policy.

navigation. In (c), the starting configuration is arbitrary and only the goal forms a square; CLAE pulls the robots into formation *en route* and arrives in the target geometry, whereas the base policy reaches goals without maintaining team structure. Residual Action RL improves formation error to 0.247, compared with 0.086 for CLAE under the same reward and environment-step budget. The activation-editing baselines also struggle on this team-dependent objective, while the multiplicative-only variant remains close to the base policy, supporting the benefit of state-dependent affine edits that vary with teammate configuration.

Camera-aware stealth navigation. CLAE reduces the Watchability score from 3596.6 to 1151.3, a $\sim$3.1$\times$ reduction in cumulative camera exposure, while maintaining a 96.5% collision-free rate and a 91.2% robot-success rate (Table IIc). The multiplicative-only ablation reduces exposure by about 4% despite using the same selected SAE latents and RL training, suggesting that gain modulation alone is insufficient for injecting camera-aware behavior. Residual Action RL does not reduce camera exposure, increasing Watchability above the base policy (3636.3 vs. 3596.6). In Fig. 2d, where low-exposure goal-reaching paths exist, CLAE routes robots around camera footprints or drives them quickly through exposed regions, reducing *cumulative* rather than instantaneous exposure. In Fig. 2e, where camera coverage is denser, CLAE prioritizes exposure reduction over goal reaching, reducing Watchability to 0 while some robots stop short of their goals. Together, these examples show that the same activation-editing interface can discover qualitatively different deployment-time strategies, trading off goal progress and stealth behavior.

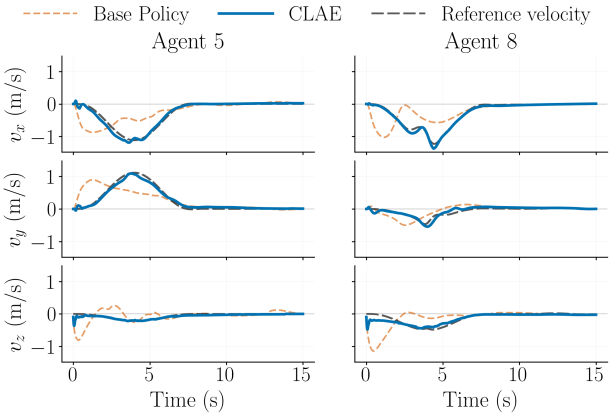

Fig. 3: **Per-axis velocity tracking.** CLAE (blue) follows the reference (dashed grey) on v_x, v_y, v_z for two representative robots; the unedited base policy (dashed orange) diverges on every axis.

Finally, DSRL improves all three behavior-specific metrics relative to the unsteered distilled diffusion policy, but both variants retain collision-free rates well below those of the original base policy across all three tasks. We attribute the navigation degradation primarily to distillation-induced

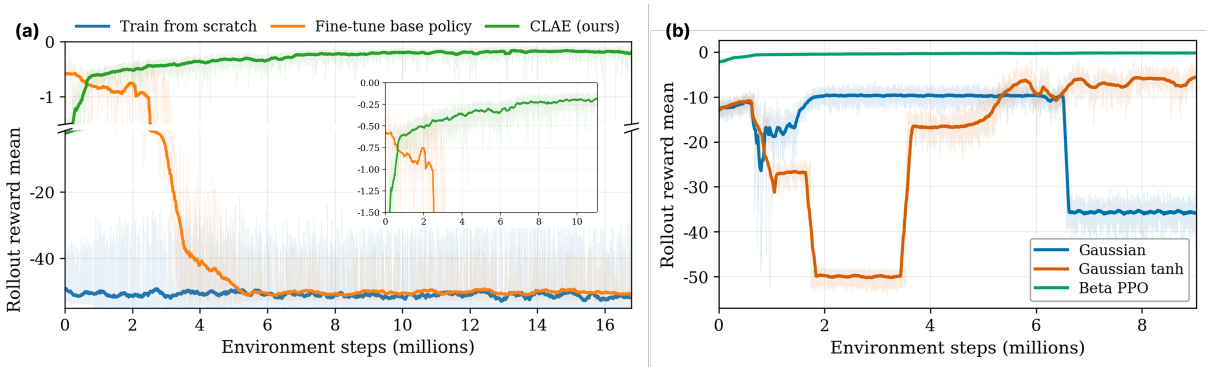

Fig. 4: **Learning curves for design ablations. (a)** CLAE versus fine-tuning and training from scratch on the formation objective. **(b)** Beta, Gaussian, and Gaussian-tanh PPO action heads. Curves are from representative runs and smoothed for visualization; final-checkpoint rollout performance is reported in Table III.

covariate shift [30]. DSRL is unable to recover the lost navigation performance from the distilled policy. We hypothesize this is due to the support of the distilled policy's action distribution being too narrow. Because DSRL operates on a separately distilled diffusion policy, we treat it as an auxiliary comparison rather than a controlled comparison on a shared base policy.

Summary. Across all three tasks, CLAE steers the frozen base policy toward behaviors it was never trained to primarily express, per-robot velocity tracking, multi-robot formation control, and camera-aware stealth navigation, while keeping collision avoidance and goal reaching behavior close to the unedited base policy. These results show that small, bounded affine edits at each policy step can compound over closed-loop execution to produce substantial behavior-level steering.

E. Design Ablations

We isolate three design choices on the formation-control task: activation editing vs direct policy weight optimization, the PPO action head for affine editing, and restricting edits to a compact set of behavior-relevant SAE latents. All learned methods use the same formation reward and environment-step budget, while all activation-editing variants intervene at the same layer. Results appear in Table III; learning curves for the first two ablations are shown in Figure 4.

a) Activation editing mitigates catastrophic forgetting.: We first compare CLAE with directly optimizing policy weights through fine-tuning the pretrained base policy or training a new policy from scratch on the formation reward. Fine-tuning begins competitively, while the pretrained navigation behavior is still largely retained, but its performance deteriorates as optimization proceeds (Fig. 4a). At convergence, its formation error is 0.199, more than twice that of CLAE, while collision-free rate drops to 93.5% and success rate to 0.890. Training from scratch fails to acquire stable navigation within the same environment-step budget. In contrast, CLAE keeps the base policy weights frozen and improves the formation objective by $\sim 3.2\times$ over the base while largely retaining its navigation competence.

b) Beta-PPO stabilizes bounded affine editing.: CLAE uses a Beta-parameterized PPO action head to produce bounded affine edits, with multiplicative gains in $[0.5, 10.0]$ and additive offsets in $[-0.1, 0.1]$. We compare it against an unbounded Gaussian head and a Gaussian-tanh head, which enforces the same bounds through a nonlinear tanh squashing function. Both alternatives lead to unstable closed-loop learning (Fig. 4b): Gaussian PPO drops to 2.6% collision-free, while Gaussian-tanh fails completely (Table III). The failure of the bounded Gaussian–tanh variant shows that constraining the edit range alone is insufficient. The Beta head is naturally bounded with a flexible shape over the interior of the bounds, providing a well-suited parameterization for the small affine edits used by CLAE. This is consistent with the closed-loop behavior in Sec. IV-D: small per-step edits can accumulate through the closed-loop dynamics to produce substantial behavior-level changes.

Method	**Form. Err.** ↓	CF (%) ↑	SR ↑	Goal (m) ↓
Base Policy	0.280	99.5	0.997	0.271
SAE reconstruction (no steering)	0.284	99.1	0.990	0.275
Fine-tune base policy	0.199	93.5	0.890	0.199
Train from scratch	4.715	0.0	0.000	—
CLAE w/ Gaussian PPO	0.761	2.6	0.000	0.761
CLAE w/ Gaussian-tanh PPO	3.276	0.0	0.000	—
CLAE w/o SAE (full-activation)	0.797	66.6	0.127	0.797
CLAE (ours)	**0.086**	98.5	0.994	0.281

TABLE III: **Design ablations of CLAE on the multi-robot formation-control task.** All learned variants share the same intervention point, reward, and environment-step budget; the SAE-reconstruction row involves no steering policy and no training.

c) SAE and behavior probing improve editing performance.: We first verify that the SAE itself is behavior-neutral by setting $m_{i,t} = 1$ and $c_{i,t} = 0$, so the intercepted activation is replaced by its SAE reconstruction with no steering (Table III). Formation error, collision-free rate, and success remain at base-policy levels (0.284, 99.1%, 0.990), confirming that the gains reported for CLAE come from the learned edits rather than from inserting the SAE. We then remove the SAE and behavior-probing stage and allow the steering policy to directly edit every dimension of the intercepted activation. This increases the steering action dimension from $2|S_b|$ to $2d$ and removes the restriction to behavior-relevant features. Even though the intercepted activation is only $d = 30$ dimensional, full-activation editing increases formation error to 0.797, reduces the collision-free

rate to 66.6%, and lowers success to 0.127. These results show the benefit of restricting edits to a compact, probe-selected subset of behavior-relevant SAE latents rather than directly controlling the full activation.

F. Real-World Deployment

We deploy CLAE on Crazyflie 2.1 quadrotors, a severely resource-constrained platform. We focus on the formation control task and deploy the frozen base policy, SAE, and steering policy zero-shot from simulation. The entire CLAE pipeline is implemented in C for real-time execution on the STM32 microcontroller while remaining compatible with the 1 kHz stabilization loop. Each quadrotor performs onboard localization through optical flow, broadcasts its state to neighbors over a radio link, and generates a local SDF (2 m range) onboard from *a priori* obstacle positions. All computation runs entirely onboard at 100 Hz. Figure 1(b,c) shows representative trajectories: the base policy reaches the goals by splitting the formation between the middle obstacle, while CLAE keeps the team flying close together as a group retaining the base navigation behavior.

V. Conclusion

We introduced CLAE, a framework that treats post-training behavior steering of frozen robot policies as a closed-loop problem over SAE latents of an intermediate activation. By learning state-dependent affine edits on a small set of behavior-relevant SAE latents, CLAE steers individual robot behavior, coordinates multirobot behavior, and supports novel behavior injection, all without touching the base policy weights or action head. Because the interface requires only white-box access to an intermediate activation, we see CLAE as a step toward a general recipe for adapting pretrained robot policies to deployment-time behavior objectives that were not the primary objective of, or were absent from, the base policy's training, complementing well-established finetuning approaches. Two limitations point to future work. CLAE relies on rollout-derived metrics to identify behavior-relevant activations, and for behaviors exogenous to the base policy's objective we used trajectory-modulation proxies; principled alternatives for selecting latents in this regime remain open. We also provide no formal safety or stability guarantees on the edited closed-loop system, and extending the interface to large pretrained models such as VLAs is a natural next step we have not yet explored.

References

[1] S. Liu, I. S. Singh, Y. Xu, J. Duan, and R. Krishna, "Vls: Steering pretrained robot policies via vision-language models," *arXiv preprint arXiv:2602.03973*, 2026.

[2] M. Nakamoto, O. Mees, A. Kumar, and S. Levine, "Steering your generalists: Improving robotic foundation models via value guidance," *Conference on Robot Learning (CoRL)*, 2024.

[3] Y. Wang *et al.*, "Inference-time policy steering through human interactions," in *2025 IEEE ICRA*.

[4] A. Wagenmaker, Y. Zhang, M. Nakamoto, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine, "Steering your diffusion policy with latent space reinforcement learning," 2025.

[5] Y. Wu, R. Tian, G. Swamy, and A. Bajcsy, "From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment," *Robotics: Science and Systems (RSS)*, 2025.

[6] W. Chen, J. S. Bhatia, C. Glossop, N. Mathihalli, R. Doshi, A. Tang, D. Driess, K. Pertsch, and S. Levine, "Steerable vision-language-action policies for embodied reasoning and hierarchical control," 2026.

[7] A. M. Turner, L. Thiergart, G. Leech, D. Udell, J. J. Vazquez, U. Mini, and M. MacDiarmid, "Steering language models with activation engineering," 2023.

[8] A. Zou, L. Phan, S. Chen, J. Campbell, P. Guo, R. Ren, A. Pan, X. Yin, M. Mazeika *et al.*, "Representation engineering: A top-down approach to ai transparency," 2023.

[9] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey, "Sparse autoencoders find highly interpretable features in language models," 2023.

[10] A. Templeton, *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic, 2024.

[11] B. Häon, K. Stocking, I. Chuang, and C. Tomlin, "Mechanistic interpretability for steering vision-language-action models," 2025.

[12] A. Swann, L. McGranahan, H. Buurmeijer, M. Kennedy, and M. Schwager, "Sparse autoencoders reveal interpretable and steerable features in vla models," 2026.

[13] S. Das, D. Chiu, Z. Huang, L. Lindemann, and G. S. Sukhatme, "Latent activation editing: Inference-time refinement of learned policies for safer multirobot navigation," 2025.

[14] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," 2018.

[15] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," 2018.

[16] S. Singh, S. Ravfogel, J. Herzig, R. Aharoni, R. Cotterell, and P. Kumaraguru, "Representation surgery: Theory and practice of affine steering," 2024.

[17] H. Buurmeijer, C. A. Alonso, A. Swann, and M. Pavone, "Observing and controlling features in vision-language-action models," *arXiv preprint arXiv:2603.05487*, 2026.

[18] T. Bricken *et al.*, "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread*, 2023.

[19] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," 2016.

[20] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viégas, and R. Sayres, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors," in *International Conference on Machine Learning*, 2018.

[21] Y. Belinkov, "Probing classifiers: Promises, shortcomings, and advances," *Computational Linguistics*, vol. 48, no. 1, pp. 207–219, 2022.

[22] I. G. Petrazzini and E. A. Antonelo, "Proximal policy optimization with continuous bounded action space via the beta distribution," in *symposium series on computational intelligence (SSCI)*. IEEE, 2021.

[23] Z. Huang, S. Batra, T. Chen, R. Krupani, T. Kumar, A. Molchanov, A. Petrenko, J. A. Preiss, Z. Yang, and G. S. Sukhatme, "Quadswarm: A modular multi-quadrotor simulator for deep reinforcement learning with direct thrust control," *arXiv preprint arXiv:2306.09537*, 2023.

[24] Z. Huang, Z. Yang, R. Krupani, B. Şenbaşlar, S. Batra, and G. S. Sukhatme, "Collision avoidance and navigation for a quadrotor swarm using end-to-end deep reinforcement learning," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024.

[25] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. Ieee, 2011, pp. 2520–2525.

[26] L. Wang, A. Ames, and M. Egerstedt, "Safety barrier certificates for heterogeneous multi-robot systems," in *Amer. cont. conf. (ACC)*, 2016.

[27] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[28] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*. PMLR, 2011.

[29] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, 2021.

[30] W. M. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," in *international conference on artificial intelligence and statistics*. PMLR, 2019.