Representation-Augmented Reinforcement Learning for Unified Navigation–Combat Agents in ViZDoom

Patrick Chen
Robotics Institute, School of Computer Science
Carnegie Mellon University, USA
bochunc@andrew.cmu.edu

Nicholas Leone
Robotics Institute, School of Computer Science
Carnegie Mellon University, USA
nleone@andrew.cmu.edu

Emanuel Muñoz Panduro
Robotics Institute, School of Computer Science
Carnegie Mellon University, USA
emunoz@andrew.cmu.edu

Abstract: We study representation-augmented reinforcement learning for pixel-based control in ViZDoom [1], focusing on BASIC (reactive combat) and MyWayHome (sparse navigation). Our core question is whether upgrading only the visual representation (frozen pretrained encoders such as DINOv2/v3 [2, 3] or a JEPA-pretrained CNN [4]) and/or adding intrinsic exploration (RND) improves PPO [5] sample efficiency and final performance. We implement a modular PPO agent with a swappable encoder, evaluate transfer protocols (frozen vs. partial unfreezing vs. full fine-tuning), and report both positive and negative results, including when RND helps sparse navigation but tends to slow reactive combat. Code is available at: github.com/16831-RL-Final-Project-PNE/....

Keywords: Reinforcement Learning, Representation Learning, ViZDoom, PPO, JEPA, RND

1 Introduction

Learning control directly from egocentric RGB is a canonical stress test for embodied intelligence: observations are high-dimensional and partially observable, rewards can be sparse, and policies must couple perception with fast closed-loop decision making. These challenges appear in robotics and autonomous systems whenever raw visual sensing is used as the primary input. ViZDoom [1, 6] offers a lightweight but non-trivial proxy: it supports headless simulation at scale while preserving first-person visual complexity and control-relevant dynamics. [1]

1.1 Motivation

A common baseline for discrete control from pixels is Proximal Policy Optimization (PPO) [5], which is stable and easy to implement, yet often spends substantial interaction budget learning low-level visual features. Meanwhile, strong self-supervised visual representations (e.g., the DINO family [2, 3]) and predictive self-supervised objectives (e.g., I-JEPA [4]) produce pretrained backbones

[1] Code: https://github.com/16831-RL-Final-Project-PNE/Representation-Augmented_RL_for_Unified_Navigation-Compat_Agents_in_ViZDoom.

that could reduce the burden of learning perception from scratch. Separately, intrinsic motivation methods such as Random Network Distillation (RND) can improve exploration under sparse reward. This motivates a practical course-relevant question: *if we hold the RL algorithm fixed, how far can we go by upgrading only the visual representation and/or adding a principled exploration bonus?*

1.2 Problem Statement and Prior Work Context

We study on-policy pixel-based RL in ViZDoom [1, 6] and ask: **holding PPO fixed** [5], can we improve (i) sample efficiency and/or (ii) asymptotic performance by swapping only the visual encoder (end-to-end CNN vs. frozen DINOv2/v3 [2, 3] vs. a task-adapted CNN pretrained with a JEPA-style objective inspired by I-JEPA [4]) and optionally adding RND-based intrinsic rewards for exploration? DrQ/DrQ-v2 [7, 8] and world-model methods such as Dreamer [9, 10] are strong alternative paradigms for pixel-based control, but our goal is an ablation-style study that isolates *encoder transfer* and *intrinsic exploration* effects under an unchanged PPO backbone.

1.3 Our Idea (Key Idea, Intuition, Relation to Prior Work)

Key idea. We build a unified PPO agent whose perception module is explicitly swappable and evaluate three representation regimes: (i) an end-to-end 3-layer CNN trained with PPO, (ii) frozen general-purpose DINOv2/v3 encoders [2, 3], and (iii) a CNN pretrained on ViZDoom rollouts via a JEPA-style predictive objective inspired by I-JEPA [4]. Orthogonally, we optionally add RND intrinsic reward to encourage exploration, and (for JEPA transfer) compare frozen vs. partial unfreezing vs. full fine-tuning.

Intuition: why the idea makes sense. In sparse navigation (MyWayHome), learning can be bottlenecked by exploration and unstable early perception; pretrained features can reduce early representation collapse, and novelty-based intrinsic reward can accelerate discovery. In reactive combat (BASIC), rewards are dense and success depends on task-specific, fine-grained cues; frozen features and novelty bonuses can be mismatched and inject variance, potentially slowing convergence.

Relation to prior work. Our DINO transfer follows the plug-and-play frozen encoder paradigm [2, 3]. Our JEPA pretraining follows I-JEPA's stop-gradient + EMA target design [4], and we additionally study **temporal JEPA** by predicting future representations (target at $t + \Delta$), aligning the pretraining signal with control-relevant temporal structure. RND is a standard intrinsic motivation method for sparse-reward exploration; we evaluate its interaction with both end-to-end and frozen representations.

1.4 Contributions

We provide: (i) a unified benchmark over BASIC and MyWayHome with consistent preprocessing and evaluation; (ii) a modular PPO implementation enabling encoder swapping and controlled ablations; (iii) an offline JEPA-style pretraining pipeline on ViZDoom rollouts inspired by I-JEPA [4], including temporal targets and an implementation-matched loss; and (iv) empirical evidence on when frozen transfer improves sample efficiency (MyWayHome), when it slows convergence (BASIC), and how partial unfreezing resolves representation–policy mismatch.

2 Related Work

ViZDoom has been widely used for RL from pixels and navigation–combat tasks [1, 6]. We use PPO [5] as a fixed on-policy backbone. DrQ and DrQ-v2 improve sample efficiency via data augmentation and are strong baselines for pixel RL [7, 8]. For representations, we evaluate frozen DINO-family self-supervised encoders [2, 3]. For predictive self-supervision, our pretraining is inspired by I-JEPA's joint-embedding predictive framework with EMA targets and stop-gradient supervision [4]. World-model approaches such as Dreamer provide an alternative paradigm by learning latent dynamics [9, 10].

3 Problem Setup

3.1 Environments

We study two ViZDoom scenarios with complementary challenges [1, 6]. **BASIC** emphasizes reactive aim-and-shoot behavior with frequent feedback, where success depends on quickly mapping pixels to short-horizon actions. **MyWayHome** emphasizes exploration and long-horizon credit assignment under sparse reward, where a policy must navigate based on partial observations and avoid getting stuck in local loops.

3.2 Observations and Preprocessing

We resize RGB frames to 84×84, normalize to $[0, 1]$, and stack 4 consecutive frames to form a 12-channel observation tensor. We use action repeat (`frame_repeat`$= 4$) and frame stacking (`frame_stack`$= 4$) to reduce control frequency and partially address partial observability.

3.3 Action Space

We implement a minimal Gym-like wrapper with four primitive ViZDoom buttons: `MOVE_FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, and `ATTACK`. For **BASIC**, we include composite actions with shooting (12 actions total). For **MyWayHome**, shooting is removed (6 actions total). Full action enumerations are provided in Appendix A.

3.4 Evaluation Metrics

We report **evaluation average return** (higher is better) and **iterations to converge** (lower is better) under a fixed evaluation protocol. Convergence is defined as the earliest iteration where evaluation return exceeds a threshold and remains above it for m consecutive evaluation checkpoints (Sec. 5).

4 Methods

4.1 Overall Architecture and Ablation Axis

Our pipeline keeps the RL algorithm fixed (PPO) and modifies only two components: (i) the visual representation fed to the policy/value heads, and (ii) an optional intrinsic reward term for exploration. Figure 1 summarizes this end-to-end design. At each timestep, the environment emits $(o_t, r_t^{\text{ext}}, o_{t+1})$; we store transitions in an on-policy rollout buffer, and (optionally) compute an intrinsic novelty reward using RND from the same observations. Crucially, *PPO itself is unchanged* across all variants.

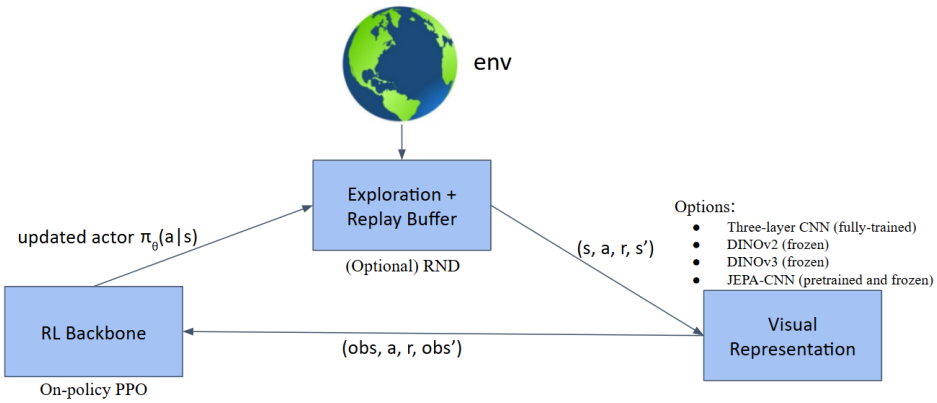

Figure 1: System overview and ablation axis. PPO [5] is fixed as the RL backbone. We vary (i) the visual representation (3-layer CNN trained end-to-end; frozen DINOv2/v3; JEPA-pretrained CNN) and (ii) an optional RND module that shapes rewards via novelty.

4.2 PPO Backbone

We use PPO [5] with a shared encoder ϕ_θ and separate actor/critic heads. Given observation o_t and action a_t, PPO maximizes the clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\big)\Big], \tag{1}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$ and $\hat{A}_t$ is the advantage estimate. We optimize

$$L(\theta) = L^{\text{CLIP}}(\theta) - c_v\,L^{\text{V}}(\theta) + c_e\,\mathbb{E}_t\big[\mathcal{H}(\pi_\theta(\cdot \mid o_t))\big], \tag{2}$$

with value loss $L^{\text{V}}(\theta) = \mathbb{E}_t[(V_\theta(o_t) - \hat{V}_t)^2]$ and entropy regularization. All PPO hyperparameters, rollout length, and evaluation cadence are held constant across ablations.

4.3 Intrinsic Exploration via Random Network Distillation (RND)

RND produces a novelty-based intrinsic reward by measuring prediction error of a trainable predictor network against a fixed random target network. Let f_{tgt} be fixed at random initialization and f_{pred} be trainable. Given an input x_t (we use the stacked observation tensor), the intrinsic reward is

$$r_t^{\text{int}} = \|f_{\text{pred}}(x_t) - f_{\text{tgt}}(x_t)\|_2^2\,. \tag{3}$$

Combining intrinsic and extrinsic rewards. We shape the reward used for PPO updates by adding a scaled intrinsic term to the environment reward:

$$r_t^{\text{train}} = r_t^{\text{ext}} + \eta\,\tilde{r}_t^{\text{int}}, \tag{4}$$

where η is the intrinsic reward coefficient and $\tilde{r}_t^{\text{int}}$ denotes the normalized (and optionally gated) intrinsic reward actually used in training. In our implementation, we apply intrinsic reward only on zero-extrinsic-reward steps:

$$\tilde{r}_t^{\text{int}} = \mathbb{I}\big[r_t^{\text{ext}} = 0\big] \cdot \frac{r_t^{\text{int}}}{\sigma_{\text{EMA}} + \epsilon}, \tag{5}$$

where σ_{EMA} is a running (EMA) standard deviation over recent intrinsic rewards and ϵ is a small constant for numerical stability. For reporting, evaluation return is computed using r_t^{ext} only (no intrinsic reward). Additional implementation details are provided in Appendix B.

4.4 Frozen DINOv2 / DINOv3 Encoders

We evaluate frozen DINO-family encoders as plug-in visual representations [2, 3]. In this regime, the encoder is a *fixed feature extractor*: we compute an embedding for each observation and train only the PPO actor/critic heads on top. Additional DINOv2 vs. DINOv3 analysis and qualitative comparisons are provided in Appendix E.

4.5 JEPA-Style Pretraining (Spatial and Temporal)

Our JEPA-style pretraining follows I-JEPA's joint-embedding predictive design with an online encoder, a predictor, and an EMA target encoder [4]. Figure 2 illustrates the online/target branches and the stop-gradient supervision used to train the predictor. The online encoder ϕ sees a (possibly masked) context input and produces $s_x = \phi(\tilde{x})$; a predictor g outputs $\hat{s}_y = g(s_x)$. The target encoder $\bar{\phi}$ processes the target input y to produce $s_y = \bar{\phi}(y)$, is updated by EMA, and is stop-gradient.

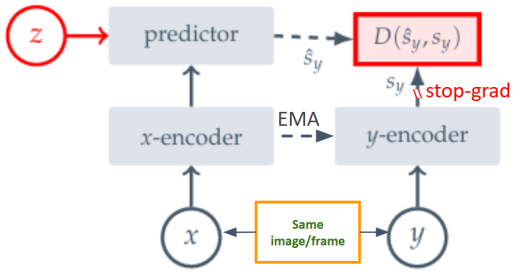

Figure 2: JEPA training schematic (I-JEPA style) [4]. The online x-encoder and predictor learn to match representations produced by an EMA-updated y-encoder under stop-gradient targets.

Masking. We apply patch-wise masking to the *context* input $\tilde{x}$: the online encoder sees partially masked frames, while the target encoder processes unmasked targets. A visual illustration of the masking protocol is provided in Appendix C to save main-page space.

Temporal extension (our addition). Beyond single-frame JEPA ($\Delta = 0$), we train **temporal JEPA** by setting the target to a future frame $y = o_{t+\Delta}$. This encourages the encoder to capture time-persistent, control-relevant structure, improving alignment with downstream RL.

JEPA objective (implementation-matched). Given a context input x and target input y (TD0 or TDΔ), we sample a patch-wise mask M and form $\tilde{x} = x \odot (1 - M)$. The online branch outputs $s_x = \phi(\tilde{x})$ and $\hat{s}_y = g(s_x)$, while the EMA teacher outputs $s_y = \bar{\phi}(y)$ and is stop-gradient. We train with an MSE prediction loss plus variance/covariance regularization (VICReg-style) on predictor outputs:

$$\mathcal{L}_{\text{JEPA}} = \underbrace{\|\hat{s}_y - \text{sg}(s_y)\|_2^2}_{\mathcal{L}_{\text{pred}}} + \lambda_{\text{var}} \underbrace{\frac{1}{D}\sum_{j=1}^{D} \max(0, \sigma^\star - \sigma_j)}_{\mathcal{L}_{\text{var}}} + \lambda_{\text{cov}} \underbrace{\text{mean}\big(\text{offdiag}(C)^2\big)}_{\mathcal{L}_{\text{cov}}}, \tag{6}$$

$$z = \hat{s}_y - \frac{1}{B}\sum_{i=1}^{B} \hat{s}_{y,i}, \quad \sigma_j = \sqrt{\text{Var}(z_{:,j}) + 10^{-4}}, \quad C = \frac{z^\top z}{\max(B-1, 1)}. \tag{7}$$

Here $\text{offdiag}(C)$ zeros the diagonal of C, $\sigma^\star = 1$, and $(\lambda_{\text{var}}, \lambda_{\text{cov}}) = (\texttt{var\_weight}, \texttt{covar\_weight})$.

EMA teacher update. After each online update, the teacher encoder is updated by EMA: $\bar{\theta} \leftarrow m\,\bar{\theta} + (1-m)\,\theta$ with $m = \texttt{momentum}$.

Transfer protocols (JEPA $\rightarrow$ PPO). We compare: (1) frozen encoder, (2) unfreeze last 1 conv block, (3) unfreeze last 2 conv blocks, (4) full fine-tuning from JEPA initialization.

5 Experiments

5.1 Training Details

We train all agents using Proximal Policy Optimization (PPO) [5] for a fixed number of iterations and evaluate periodically over a fixed number of episodes. All variants share identical observation preprocessing, rollout length, optimization hyperparameters, and evaluation cadence to ensure fair comparison across ablations. When Random Network Distillation (RND) is enabled, intrinsic rewards are applied during training as described in Sec. 4.3, while *all reported evaluation metrics are computed using extrinsic reward only*.

To quantify sample efficiency, we define **iterations to converge** as the smallest training iteration index at which the evaluation return exceeds a task-specific threshold τ and remains above this threshold for m consecutive evaluation checkpoints. We set $(\tau, m) = (80, 10)$ for **BASIC** and $(\tau, m) = (0.9, 10)$ for **MyWayHome**. This definition penalizes transient performance spikes and ensures that reported convergence reflects stable task mastery rather than stochastic variation.

5.2 What We Compare

Our experimental design isolates two orthogonal factors (Fig. 1):

1. **Visual representation**: end-to-end CNN baseline, frozen DINOv2/DINOv3 encoders, and JEPA-pretrained CNNs with varying temporal prediction horizons.
2. **Exploration mechanism**: standard PPO vs. PPO augmented with RND-based intrinsic rewards.

6 Results

6.1 Overall Quantitative Results (All Agents)

Table 1 summarizes end-to-end performance under the fixed PPO backbone (Fig. 1). Reported evaluation returns use *extrinsic reward only*. Note the tasks operate on different extrinsic reward scales: BASIC $\in [-410,\ 93]$, MyWayHome $\in [-0.21,\ 1]$.

Table 1: **Overall results across agents.** Higher is better for evaluation average return; lower is better for iterations to converge. Reported evaluation returns use *extrinsic reward only*. The extrinsic return range differs substantially across tasks: BASIC $\in [-410,\ 93]$, MyWayHome $\in [-0.21,\ 1]$.

Agents	Eval Avg Return (BASIC) ↑	Iters to Converge (BASIC) ↓	Eval Avg Return (MyWayHome) ↑	Iters to Converge (MyWayHome) ↓
Random	-156.9	NA	-0.117	NA
PPO+CNN	89.4	67	0.947	241
PPO+DINOv2 (frozen) [2]	85.0	131	0.949	26
PPO+DINOv3 (frozen) [3]	80.7	195	0.941	20
PPO+CNN+RND	84.1	147	0.948	189
PPO+DINOv2+RND [2]	86.4	129	0.952	98
PPO+DINOv3+RND [3]	84.1	155	0.945	36
PPO+JEPA_TD0 [4]	-221.5	NA	-0.24	NA
PPO+JEPA_TD1 [4]	89.3	118	0.950	29
PPO+JEPA_TD2 [4]	89.4	99	0.955	29
PPO+JEPA_TD3 [4]	89.8	81	0.957	26

6.2 Learning Curves (Evaluation Return vs. Iterations)

Figure 3 complements Table 1 by showing the full learning dynamics (extrinsic reward only). In **MyWayHome**, stronger representations (frozen DINOv2/v3 and temporal JEPA TD1–TD3) sharply reduce time-to-solve, reaching near-optimal performance within $\sim$20–29 iterations, compared to

241 iterations for PPO+CNN. In **BASIC**, most successful agents eventually reach similar asymptotic return (≈80–90), but frozen transfer (DINO, JEPA) typically converges more slowly than end-to-end CNN, and RND increases early variance and often delays stable convergence.

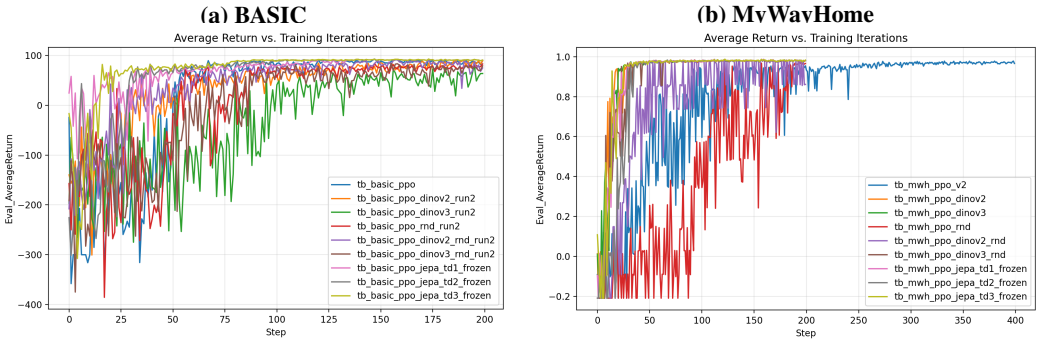

Figure 3: **Evaluation average return vs. training iterations** for all agents (extrinsic reward only). (a) BASIC: most variants reach similar final returns, but frozen transfer and RND can delay stable convergence. (b) MyWayHome: frozen DINO and temporal JEPA rapidly reach high return, substantially improving sample efficiency over PPO+CNN.

6.3 RND Analysis: Why It Helps MyWayHome More Than BASIC

Our RND design normalizes intrinsic rewards and applies them only on zero-extrinsic-reward steps (Sec. 4.3 and Appendix B). This targets sparse navigation, improving PPO+CNN convergence in MyWayHome (241 → 189 iterations in Table 1). However, once strong pretrained representations already solve exploration quickly (e.g., frozen DINOv2/v3 at 26/20 iterations), adding RND can *slow* convergence (DINOv2: 26 → 98; DINOv3: 20 → 36), consistent with novelty bonuses adding variance after reward discovery. In BASIC, where rewards are frequent and task-aligned, RND substantially delays the end-to-end CNN baseline (67 → 147) and does not provide a consistent convergence benefit for frozen encoders (e.g., DINOv2: 131 → 129; DINOv3: 195 → 155), suggesting novelty-seeking is generally misaligned with reactive combat optimization even under conservative gating.

6.4 JEPA Results: Successes and Failures

Temporal JEPA transfers robustly: TD1–TD3 match PPO+CNN asymptotic performance in BASIC (89.3–89.8 vs. 89.4) while improving MyWayHome sample efficiency dramatically (241 → 26–29 iterations) and achieving the best MyWayHome return (0.957 for TD3 in Table 1). In contrast, JEPA_TD0 fails catastrophically under frozen transfer in both tasks, suggesting that the pretrained features are not a good fit for PPO: when the pretraining target does not capture temporal structure and the encoder is kept frozen, the policy cannot effectively use the representation to learn control.

6.5 JEPA Transfer Ablation: Frozen vs. Unfreezing vs. Full Fine-tune

To isolate whether downstream adaptation resolves representation–policy mismatch, we run a transfer-protocol ablation using the **JEPA_TD3** pretrained encoder. All rows in Table 2 share the same JEPA_TD3 initialization and differ only in how many encoder layers are allowed to update during PPO training. Figure 4 shows the corresponding learning curves.

Table 2: **JEPA_TD3 transfer protocols** (same column definitions as Table 1). We compare frozen transfer vs. partial unfreezing vs. full fine-tuning, all initialized from the same JEPA_TD3 pretrained encoder.

Agents	Eval Avg Return (BASIC) ↑	Iters to Converge (BASIC) ↓	Eval Avg Return (MyWayHome) ↑	Iters to Converge (MyWayHome) ↓
PPO+JEPA_TD3 (frozen) [4]	89.8	81	0.957	26
PPO+JEPA_TD3 (unfreeze last 1 layer) [4]	89.4	43	0.967	31
PPO+JEPA_TD3 (unfreeze last 2 layers) [4]	92.6	16	0.952	26
PPO+JEPA_TD3 (fine-tune all layers; JEPA init) [4]	90.2	20	0.959	25

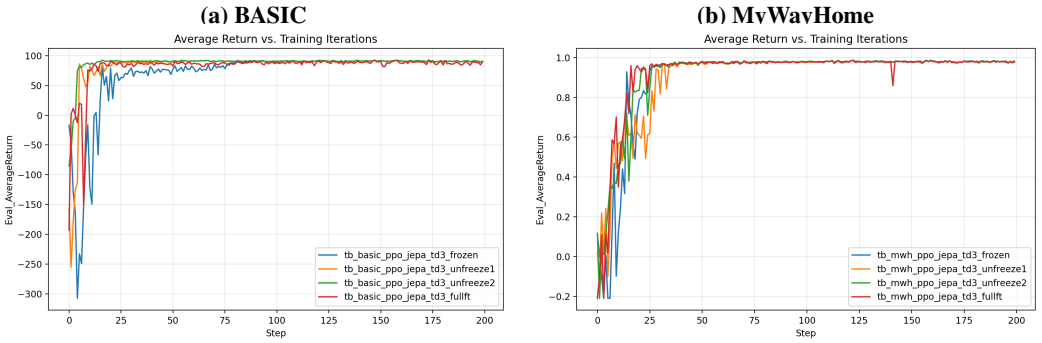

Figure 4: **JEPA_TD3 transfer-protocol learning curves** (extrinsic reward only). In BASIC, allowing limited adaptation substantially accelerates stable learning (81 → 43/16/20 iters for unfreeze-1 / unfreeze-2 / full fine-tune) and unfreezing the last two layers achieves the best return (92.6). In MyWayHome, all protocols converge quickly (25–31 iters) with modest trade-offs between speed and peak return.

In BASIC, allowing limited adaptation yields large gains in convergence speed (81 → 43/16/20 iterations in Table 2), and unfreezing the last two layers achieves both the fastest convergence (16) and the best return (92.6). In MyWayHome, frozen transfer is already strong (26 iterations), and additional adaptation produces smaller changes: full fine-tuning is marginally fastest (25), while unfreezing the last one layer attains the highest return (0.967) but converges more slowly (31).

6.6 Qualitative Rollouts (Supplementary Evidence)

To complement the quantitative trends in Table 1, Fig. 3, Table 2, and Fig. 4, we provide representative rollout GIFs for both tasks in Appendix D. Qualitatively, stronger pretrained representations (DINO, temporal JEPA) reduce early training collapse in MyWayHome (fewer stuck local loops) and produce more consistent navigation progress, while in BASIC end-to-end CNN training and partial/full temporal JEPA fine-tuning quickly learns task-specific cues for aiming/shooting.

7 Discussion

The dominant gains appear in MyWayHome, where sparse reward makes exploration and stable early perception the primary bottlenecks. Frozen DINOv2/v3 and temporal JEPA TD1–TD3 reduce iterations-to-converge from 241 (PPO+CNN) to 20–29 (Table 1), indicating that strong pretrained features substantially reduce early-stage representation learning collapse and accelerate discovery of successful trajectories. RND helps PPO+CNN (241 → 189), but when paired with already-strong encoders it can slow convergence (e.g., DINOv2: 26 → 98; DINOv3: 20 → 36), consistent with intrinsic novelty adding gradient variance after reward discovery.

In BASIC, where rewards are dense and the optimal policy relies on fine-grained, action-conditioned cues, the end-to-end CNN baseline converges faster than frozen transfer variants in the full-agent comparison (67 vs. 81–195 iterations in Table 1), and RND generally delays stable learning (67 → 147 for PPO+CNN+RND). However, the JEPA_TD3 transfer-protocol ablation shows that

representation–policy alignment can be recovered and improved with modest downstream adaptation: unfreezing the last two encoder layers yields rapid convergence (16 iterations) and the highest BASIC return (92.6) in Table 2. Overall, these results support a practical takeaway: strong pretraining is most reliably beneficial in sparse navigation, while reactive combat often requires at least partial encoder adaptation to match task-specific visual cues.

Concrete next steps include analyzing why TD0 fails (e.g., insufficient temporal alignment), tuning intrinsic reward schedules beyond conservative gating, and adding DrQ/DrQ-v2 augmentation baselines [7, 8] for a stronger pixel-RL comparison.

8 Conclusion

We evaluated whether, *holding PPO fixed*, we can obtain gains in pixel-based control by improving visual representation and exploration. In MyWayHome, pretrained representations yield large sample-efficiency gains: frozen DINOv2/v3 reduce convergence from 241 (PPO+CNN) to 26/20 iterations, and temporal JEPA TD1–TD3 achieves near-optimal returns (up to 0.957) within 26–29 iterations (Table 1).

The JEPA_TD3 transfer ablation shows that downstream adaptation can still help even when frozen transfer is already strong. In MyWayHome, **JEPA_TD3 full fine-tuning achieves the fastest convergence** (25 iterations, 0.959 return), while **unfreezing the last 1 layer achieves the best return** (0.967; Table 2, Fig. 4). RND improves the PPO+CNN baseline (241 $\rightarrow$ 189) but can slow convergence when combined with already-strong encoders (e.g., DINOv2: 26 $\rightarrow$ 98; DINOv3: 20 $\rightarrow$ 36).

In BASIC, most successful variants reach similar asymptotic return ($\approx$80–90) in the full-agent comparison (Table 1), but frozen transfer typically converges more slowly and RND delays stable learning for the CNN baseline (67 $\rightarrow$ 147). The JEPA_TD3 transfer ablation highlights the value of modest adaptation in reactive combat: unfreezing the last two encoder layers yields the fastest convergence (16 iterations) and the strongest return (92.6) among JEPA_TD3 protocols (Table 2, Fig. 4), indicating that limited fine-tuning can resolve representation–policy mismatch.

A Action Space Details

We implement a minimal Gym-like wrapper with four primitive ViZDoom buttons: `MOVE_FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, and `ATTACK`. Each discrete action is a 4D multi-hot vector $\mathbf{a} = [a_f, a_l, a_r, a_s] \in \{0, 1\}^4$ in the fixed button order

$$\texttt{BUTTONS} = [\texttt{MOVE\_FORWARD},\ \texttt{TURN\_LEFT},\ \texttt{TURN\_RIGHT},\ \texttt{ATTACK}].$$

BASIC action set (12 actions, includes shooting). For **BASIC**, we use 12 discrete actions (index $\rightarrow$ multi-hot vector $\rightarrow$ semantic meaning):

$$\begin{aligned}\texttt{ACTIONS\_BASIC} = [&[0,0,0,0]\ (0\text{: noop}),\\ &[1,0,0,0]\ (1\text{: forward}),\\ &[0,1,0,0]\ (2\text{: turn\_left}),\\ &[0,0,1,0]\ (3\text{: turn\_right}),\\ &[0,0,0,1]\ (4\text{: shoot}),\\ &[1,1,0,0]\ (5\text{: forward + turn\_left}),\\ &[1,0,1,0]\ (6\text{: forward + turn\_right}),\\ &[1,0,0,1]\ (7\text{: forward + shoot}),\\ &[0,1,0,1]\ (8\text{: turn\_left + shoot}),\\ &[0,0,1,1]\ (9\text{: turn\_right + shoot}),\\ &[1,1,0,1]\ (10\text{: forward + turn\_left + shoot}),\\ &[1,0,1,1]\ (11\text{: forward + turn\_right + shoot})].\end{aligned}$$

MyWayHome action set (6 actions, no shooting). For **MyWayHome**, shooting is unnecessary, so we use a compact 6-action subset without `ATTACK`:

$$\begin{aligned}\texttt{ACTIONS\_NO\_SHOOT} = [&[0,0,0,0]\ (0\text{: noop}),\\ &[1,0,0,0]\ (1\text{: forward}),\\ &[0,1,0,0]\ (2\text{: turn\_left}),\\ &[0,0,1,0]\ (3\text{: turn\_right}),\\ &[1,1,0,0]\ (4\text{: forward + turn\_left}),\\ &[1,0,1,0]\ (5\text{: forward + turn\_right})].\end{aligned}$$

B RND Implementation Details

Implementation-aligned workflow (what we actually run). Our implementation (Fig. 1) follows a strict on-policy workflow: (1) collect one PPO rollout of length `steps_per_iteration` into the rollout buffer; (2) compute per-state intrinsic rewards from the same rollout observations using ConvEncoder-based target/predictor networks (same architecture, different initializations); (3) normalize intrinsic rewards by a running standard deviation (EMA over rollouts) to stabilize scale across training; (4) apply intrinsic reward only when the environment reward is zero (Eq. 5) to target exploration under sparse reward while minimizing interference when extrinsic feedback is present; (5) train the RND predictor for a small number of epochs on the current rollout while keeping the target frozen. PPO computes returns/advantages using r_t^{train} (Eq. 4) for training updates, but evaluation is always reported with extrinsic reward only.

C JEPA Masking Illustration

Figure 5 visualizes the patch-wise masking protocol used during JEPA pretraining: the online encoder sees masked context blocks, while the EMA target encoder processes unmasked targets.

Figure 5: Masking strategy [4]. The online encoder sees masked context blocks; the target encoder processes unmasked targets.

D Qualitative Rollout GIF Links

This appendix (referenced from Sec. 6.6) provides representative rollout GIFs to support qualitative claims about navigation progress and combat behavior across agents.

- **BASIC GIFs:**
  - **Random:** Google Drive Link (BASIC, Random)
  - **PPO+CNN:** Google Drive Link (BASIC, PPO+CNN)
  - **PPO+DINOv3 (frozen):** Google Drive Link (BASIC, DINOv3)
  - **PPO+DINOv3+RND (frozen):** Google Drive Link (BASIC, DINOv3)
  - **PPO+JEPA_TD3 (frozen):** Google Drive Link (BASIC, JEPA_TD3)
  - **PPO+JEPA_TD3 (unfreeze 2):** Google Drive Link (BASIC, JEPA_TD3)
  - **PPO+JEPA_TD3 (full fine-tuning):** Google Drive Link (BASIC, JEPA_TD3)
- **MyWayHome GIFs:**
  - **Random:** Google Drive Link (MWH, Random)
  - **PPO+CNN:** Google Drive Link (MWH, PPO+CNN)
  - **PPO+DINOv3 (frozen):** Google Drive Link (MWH, DINOv3)
  - **PPO+DINOv3+RND (frozen):** Google Drive Link (MWH, DINOv3)
  - **PPO+JEPA_TD3 (frozen):** Google Drive Link (MWH, JEPA_TD3)
  - **PPO+JEPA_TD3 (unfreeze 2):** Google Drive Link (MWH, JEPA_TD3)
  - **PPO+JEPA_TD3 (full fine-tuning):** Google Drive Link (MWH, JEPA_TD3)

E DINOv2 vs. DINOv3 Additional Analysis

DINOv2 [2] and DINOv3 [3] are self-supervised vision transformers trained to produce strong, general-purpose visual descriptors. We use frozen encoders (no gradients) to isolate transfer effects under an unchanged PPO backbone (Sec. 4.4).

DINOv2 (strong general-purpose descriptors). DINOv2 [2] is a self-supervised ViT; in our experiments we use the ViT-g/14 variant. Its main appeal in our setting is early-training representation stability, which is especially useful in sparse-reward navigation.

DINOv3 (scaled model/data, denser features). DINOv3 [3] scales up the DINO family; we use the ViT-7B/16 variant. Figure 6 shows a qualitative example where DINOv3 activations appear denser and more spatially coherent than DINOv2.

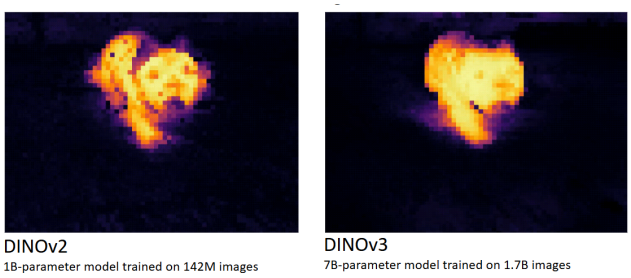

Figure 6: DINOv2 vs. DINOv3 qualitative comparison (frozen). DINOv2 exhibits more fragmented activations with visible holes, while DINOv3 produces denser, more compact, and spatially coherent features.

Acknowledgments

We thank the 16-831 instructors and TAs for feedback and course support.

References

[1] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 2016.

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

[3] O. Siméoni et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[4] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[6] M. Wydmuch, M. Kempka, and W. Jaśkowski. Vizdoom competitions: Playing doom from pixels. *IEEE Transactions on Games*, 11(3):248–259, 2019. arXiv:1809.03470.

[7] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[8] A. I. Kostrikov, D. Yarats, and R. Fergus. Drq-v2: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[9] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[10] D. Hafner, J. Pasukonis, J. Ba, and M. Norouzi. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.