VLNAPO: GRADIENT-GUIDED HARNESS ENGINEERING WITH KEY-TOKEN SUPERVISION FOR FULL-GAME VLNA POLICY OPTIMIZATION IN OPENRA

Anonymous authors
Paper under double-blind review

ABSTRACT

Full-game control in real-time strategy (RTS) games requires agents to transform partial, multimodal observations into long-horizon strategic decisions while satisfying rapidly changing game-state constraints. The central challenge is that strategic actions are combinatorial: an action that is desirable in isolation may be infeasible because of resource limits, technology prerequisites, unit availability or production-queue conflicts. Thus, an agent must jointly determine what should be done from visual, linguistic and numerical context and generate command sequences that are both strategically effective and executable. We propose VLNA, a structured multimodal policy for full-game control in OpenRA. VLNA fuses tactical renderings, language context, and numerical game states to autoregressively generate structured macro-actions, which a deterministic compiler translates into native commands. A cascaded conditional action space uses prefix-conditioned masks to restrict each action while dynamically updating resource and production-queue budgets. Through training gradients, Harness Engineering combines rule-generated local hard labels with the PPO objective, embedding infrastructure and production rules without replacing policy actions during interaction. Across five seeds with 100 evaluation games each on the `singles` map, VLNA achieved a win rate of $77\% \pm 2.00\%$, rising to $90\% \pm 1.55\%$ with persistent Harness supervision. The cascaded action space achieved a joint validity rate of 99.96%. These findings show that structured multimodal reasoning, prefix-conditioned action constraints, and gradient-guided Harness Engineering can jointly improve executability and strategic performance in full-game RTS agents.

1 INTRODUCTION

Intelligent agents operating in dynamic environments must transform heterogeneous observations into coherent long-horizon decisions while respecting constraints that change over time. Real-time strategy (RTS) games provide a demanding testbed for this capability, combining visual perception with resource allocation, construction, production, and combat under partial observability. At each decision point, available actions depend on technological progress, production-queue occupancy, available resources, and choices made earlier within the same action sequence. RTS control therefore involves a large discrete action space in which many combinations are mutually inconsistent, syntactically invalid, or infeasible under current resource constraints. Deep reinforcement learning has enabled full-game play in *StarCraft II* (Vinyals et al., 2017; 2019; Han et al., 2020). However, jointly learning visual representations, environment rules, and long-horizon strategies requires extensive interaction and substantial computational resources. Pretrained vision-language models (VLMs) offer reusable semantic representations, while robotic vision-language-action (VLA) systems show that control actions can be generated as tokens (Zitkovich et al., 2023; Kim et al., 2025). Multimodal game agents have likewise used images and language for instruction following, planning, state prediction, and strategic coordination (SIMA Team, 2024; Li et al., 2025; Shao et al., 2024; Khan & Sukthankar, 2024). Yet their high-level outputs are typically translated into executable actions by external parsers, rule systems, or low-level controllers. Rule enforcement therefore remains largely separated from the learned policy.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

For online RTS control, this separation creates two specific gaps. First, action validity depends jointly on the current observation and the previously generated action prefix. A token that is valid in isolation may become infeasible after an earlier choice consumes resources, occupies a production queue, or changes a technological prerequisite. External post-processing can reject such combinations, but it does not prevent the policy from assigning probability to them during generation. Second, inference-time rule systems can guide individual decisions without necessarily transferring that guidance into policy parameters. Their effects may therefore disappear when the external rule engine is removed. An effective RTS policy consequently requires both an action representation that expresses prefix-dependent feasibility and a training mechanism that converts external rules into persistent policy guidance.

Here, we investigate whether these requirements can be addressed through structured action-token generation and gradient-guided Harness Engineering. We construct a VLM-based policy that jointly encodes tactical renderings, textual tactical context, and structured numerical state. An independent numerical-processing branch is used to preserve quantitative information alongside the visual and linguistic representations. A task-specific decoder generates ten-slot action-token sequences, which a deterministic compiler converts into native OpenRA commands. During autoregressive decoding, prefix-conditioned validity masks filter candidate tokens using the observation, generated prefix, available resources, and production-queue state. We further propose gradient-guided Harness Engineering, in which a rule engine identifies key action slots and constructs rule-derived target labels during reinforcement learning. Joint optimization of a cross-entropy objective for selected key action slots and the reinforcement-learning objective allows these structured priors to influence policy parameters through backpropagation. Using OpenRA for full-game Red Alert control (OpenRA Developers and Contributors, 2010), we test whether the proposed representation generates valid, executable macro-actions and whether rule-guided behavior persists after Harness removal.

We make three contributions. **(1) Executable multimodal macro-actions** map visual, linguistic, and numerical observations to semantic tokens compiled into OpenRA commands. **(2) Prefix-conditioned validity** updates candidate tokens using the observation, action prefix, resources, prerequisites, and queue occupancy. **(3) Gradient-guided Harness Engineering** jointly optimizes rule-derived labels with PPO, allowing rule guidance to persist after Harness removal.

2 Related Work

2.1 Multimodal and Full-Game RTS Control

VLA systems tokenize control actions and learn reusable multimodal representations for robot control (Zitkovich et al., 2023; Kim et al., 2025). In simulated game worlds, SIMA studies instruction following from visual observations (SIMA Team, 2024). Recent language-model-based *StarCraft II* work instead provides an interaction environment, predicts build orders, or coordinates multi-agent play (Li et al., 2025; Khan & Sukthankar, 2024; Shao et al., 2024). Full-game RTS agents and macro-action methods address partial observability, long-horizon credit assignment, and large action spaces (Vinyals et al., 2017; 2019; Han et al., 2020; Wang et al., 2021a; Pang et al., 2019; Xu et al., 2019). Structured policies represent actions sequentially, with parallel branches, or as discrete–continuous parameterizations (Metz et al., 2017; Tavakoli et al., 2018; Xiong et al., 2018). At the multi-agent RTS level, DOP combines policy gradients with value decomposition and reports results on *StarCraft II* micromanagement (Wang et al., 2021b), while Gym-μRTS provides a lightweight testbed (Huang et al., 2021). These studies primarily examine learning architectures, temporal abstraction, or low-level multi-agent control rather than structured multimodal action generation for full-game execution. Our action tokens are explicitly tied to a deterministic compiler and to the resource, queue, and prefix dependencies of a full OpenRA game. OpenRA exposes observations, execution, and trajectory logs while retaining core RTS mechanics (OpenRA Developers and Contributors, 2010), supporting controlled full-game evaluation rather than platform comparison or peak competitive performance.

2.2 Structured Action Spaces

Structured action models reduce complex decision spaces through sequential factorization, parallel action branches, or discrete–continuous parameterization (Metz et al., 2017; Tavakoli et al.,

2018; Xiong et al., 2018). For state-varying discrete action inventories, DOM-Q-NET separates action categories while grounding them in a structured environment representation (Jia et al., 2019). Action-space shaping, option discovery, and causal hierarchical reinforcement learning further restrict decisions or organize them into reusable subgoals (Kanervisto et al., 2020; Srinivas et al., 2016; Khorasani et al., 2025; Li et al., 2021). Although these methods capture inter-dimensional or temporal structure, they do not necessarily model how one generated component changes the feasibility of subsequent components. Invalid-action masking has been studied in RTS policy gradients (Huang & Ontañón, 2022); our autoregressive decoder instead represents RTS macro-actions as semantic slots and updates their candidate sets after each generated token using prefix-conditioned validity masks.

2.3 Dynamic Constraints and Rule-Guided Learning

Dynamic action constraints construct state-dependent feasible action sets, mask invalid logits, and renormalize policy distributions during PPO (Tang et al., 2020); related invalid-action masking methods have also been studied in real-time strategy control (Huang & Ontañón, 2022; Hou et al., 2023) and constrained online three-dimensional bin packing (Zhao et al., 2021). By removing infeasible choices during inference, these methods retain an external environment, rule system, or feasibility checker in the decision loop. (Schmitt et al., 2018) instead uses a pretrained teacher policy to provide a soft target distribution over the complete action space and combines soft cross-entropy supervision with reinforcement learning while the student collects its own experience. Gradient-Guided Harness Engineering differs in that its training targets are generated directly by the rule engine rather than by a pretrained teacher: given an observation o_t, the Harness produces several local action-token proposals for decision-critical positions without directly replacing the action selected by the student, and applies hard-label cross-entropy to the critical actions, with the resulting gradients guiding the PPO update. These online, rule-derived targets therefore differ from imitation-learning targets based on expert corrective labels (Ross et al., 2011); because they selectively supervise key action tokens and shape the policy parameters during training, the learned behavior can persist after the external Harness engine is removed.

3 Method

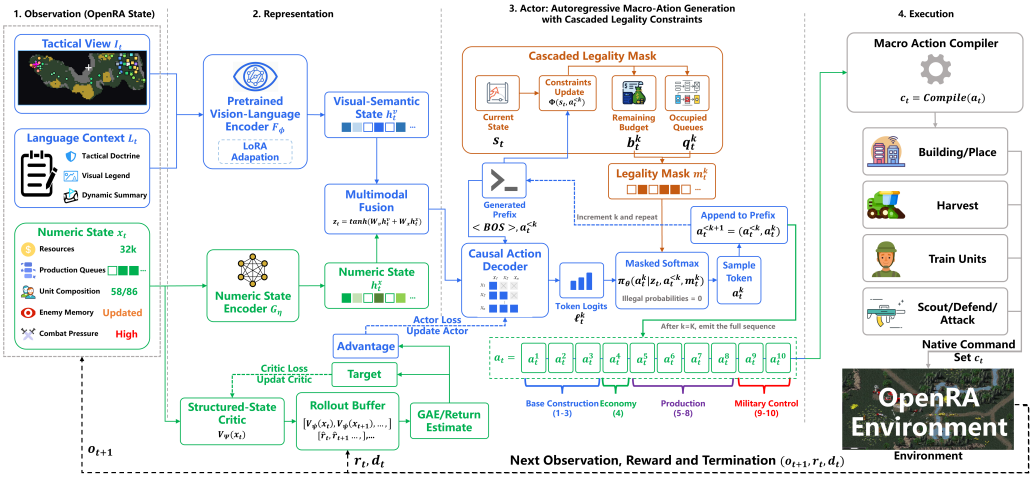


Figure 1: VLNA policy and OpenRA interaction. The actor fuses tactical images, language context, and structured numerical state; a causal decoder generates ten semantic slots with prefix-conditioned masks; a deterministic compiler issues OpenRA commands; and the structured-state critic supplies PPO value estimates.

3.1 Problem Formulation and Method Overview

We formulate full-game real-time strategy (RTS) control as a partially observable sequential decision problem. At decision step t, the simulator has latent state s_t, whereas the policy observes

$$o_t = (I_t, L_t, x_t), \tag{1}$$

where I_t is an online tactical rendering, L_t concatenates fixed tactical guidelines, a visual legend, and a textual summary of the current situation and recent changes, and x_t contains resources, production queues, buildings, unit composition, enemy memory, attack readiness, and defensive pressure. The observation-consistent control state is

$$\bar{s}_t = R(I_t, L_t, x_t), \tag{2}$$

which excludes hidden enemy locations, unrevealed map cells, and other simulator-only fields. Our method (Fig. 1) couples three components: (i) a vision-language-numerical-action (VLNA) policy that fuses the three modalities and generates a ten-slot macro-action; (ii) prefix-conditioned validity masks that restrict each slot using $\bar{s}_t$ and the generated prefix; and (iii) gradient-Guided Harness Engineering, which supplies rule-derived labels during reinforcement learning without replacing policy actions. A deterministic compiler maps the generated sequence to native OpenRA commands.

3.2 VLNA Policy and Macro-Action Representation

Because RTS control requires both semantic context and exact quantities, we augment a pretrained vision-language model with an independent numerical encoder:

$$h_t^v = F_\phi(I_t, L_t), \qquad h_t^x = G_\eta(x_t), \tag{3}$$

where F_ϕ initialized from the pretrained Qwen3.5-0.8B vision-language model and adapted using low-rank adaptation (LoRA), while G_η maps structured state to the policy representation space. We fuse the representations as

$$z_t = \tanh(W_v h_t^v + W_x h_t^x)\,, \tag{4}$$

and condition the actor and causal action decoder on z_t. The critic uses only structured state,

$$\hat{V}_t = V_\psi(x_t), \tag{5}$$

forming an asymmetric actor–critic in which the actor receives multimodal semantics and the critic estimates value from measured game variables.

RTS decisions span construction, economy, production, and military control. To avoid illegal, mutually exclusive, or semantically inconsistent combinations, we decompose each macro-action into K ordered semantic slots. For slot k, $\mathcal{A}_k = \{u_{k,1}, \ldots, u_{k,n_k}\}$ is a closed candidate set, and the task-specific vocabulary is

$$\mathcal{V}_{\text{act}} = \bigcup_{k=1}^{K} \tau_k(\mathcal{A}_k) \cup \mathcal{V}_{\text{sp}}, \tag{6}$$

where τ_k assigns each option a unique token and $\mathcal{V}_{\text{sp}}$ contains decoder-interface special tokens. At step t,

$$a_t = (a_t^1, a_t^2, \ldots, a_t^K), \qquad a_t^k \in \mathcal{A}_k. \tag{7}$$

We set $K = 10$ for construction intent, building category, building item, economic intent, vehicle production, infantry production, support category, support item, military intent, and target intent. The causal decoder factorizes

$$\pi_\theta(a_t \mid o_t) = \prod_{k=1}^{K} \pi_\theta\big(a_t^k \mid z_t, a_t^{<k}, \mu_t^k\big)\,, \tag{8}$$

where $a_t^{<k}$ is the generated prefix and μ_t^k is the validity mask below. After decoding,

$$\tau(a_t) = \big(\tau_1(a_t^1), \ldots, \tau_K(a_t^K)\big) \tag{9}$$

is compiled into native engine commands for the same interval, executing all valid non-null decisions. Tokens have fixed meanings for construction, production, harvesting, rallying, defence, scouting, assault, advance, and focus fire; sequential decoding captures dependencies while compilation executes compatible commands concurrently.

3.3 Prefix-Conditioned Validity

For each slot k, the observable state defines a base feasible set $\mathcal{B}_t^k(\bar{s}_t)$ covering technology, facilities, resources, queue depth, placement, controllable units, target visibility, and enemy memory. The generated prefix defines $\mathcal{C}_t^k(a_t^{<k})$, giving

$$\mathcal{M}_t^k(a_t^{<k}) = \mathcal{B}_t^k(\bar{s}_t) \cap \mathcal{C}_t^k(a_t^{<k}). \tag{10}$$

Thus, no construction forces null building slots, a category restricts the subsequent item, and a military intent restricts compatible targets. We implement this set with the additive logit mask

$$\mu_{t,i}^k = \begin{cases} 0, & u_{k,i} \in \mathcal{M}_t^k(a_t^{<k}), \\ -\infty, & u_{k,i} \notin \mathcal{M}_t^k(a_t^{<k}), \end{cases} \tag{11}$$

and conditional distribution

$$\pi_\theta\big(a_t^k = u_{k,i} \mid z_t, a_t^{<k}, \mu_t^k\big) = \frac{\exp(\ell_{t,i}^k + \mu_{t,i}^k)}{\sum_j \exp(\ell_{t,j}^k + \mu_{t,j}^k)}, \tag{12}$$

where $\ell_{t,i}^k$ is the unmasked decoder logit. Invalid tokens have zero probability, while valid alternatives retain learned preferences. To account for within-macro competition for resources and production capacity, the deterministic function Φ returns

$$(b_t^k, q_t^k) = \Phi(\bar{s}_t, a_t^{<k}), \tag{13}$$

where b_t^k is remaining budget and q_t^k records prefix-reserved queues. A candidate is feasible only when

$$\mathrm{legal}\left(u_{k,i}; \bar{s}_t, b_t^k, q_t^k, a_t^{<k}\right) = 1. \tag{14}$$

Masks therefore encode deterministic semantic and environmental constraints, leaving reinforcement learning to distribute probability within the valid boundary.

3.4 Gradient-Guided Harness Engineering

Validity masks remove illegal actions but do not rank feasible ones. Because sparse rewards weakly supervise useful infrastructure and production, gradient-guided Harness Engineering derives local rule labels from states visited by the current policy.

At each step, Harness receives the policy's (I_t, L_t, x_t) and evaluates rules only on $\bar{s}_t$, excluding hidden enemy locations, unrevealed map cells, and simulator-only fields. A proposal is

$$q = (r, g, p, \mathcal{U}, w), \tag{15}$$

where r, g, and p denote the rule, supervision group, and conflict priority; $\mathcal{U} = \{(k, u_k)\}$ contains recommended slot–option pairs; and $w \geq 0$ is their weight. Infrastructure rules address

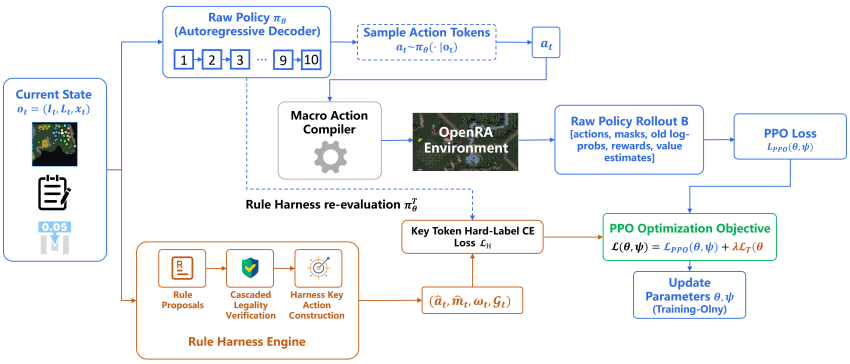


Figure 2: Gradient-guided Harness PPO. The policy executes sampled actions in OpenRA, while Harness derives valid weighted labels from the same observable inputs. The Harness cross-entropy augments PPO with fixed coefficient λ.

completed-building placement, power-shortage repair, core-building completion, and post-conflict resource expansion. Production rules address wartime mobilization, low-resource harvester protection, infantry–tank balancing, and replenishment of basic combat units. Each rule labels only relevant slots.

Proposals are mapped to the closed vocabulary, filtered by the policy's cascaded feasibility checks, and merged by priority, retaining the highest-priority option for each conflicting slot. The target $\hat{a}_t = (\hat{a}_t^1, \ldots, \hat{a}_t^K)$ satisfies

$$\hat{a}_t^k \in \mathcal{M}_t^k(\hat{a}_t^{<k}), \qquad k = 1, \ldots, K. \tag{16}$$

Because later sets depend on earlier targets, an unlabeled prefix slot can invalidate a later proposal. The constructor inserts a feasible zero-weight token and rechecks subsequent labels. For accepted rules covering slot k, $\omega_t^k = w$; otherwise $\omega_t^k = 0$. Each transition stores $\hat{a}_t$, cascaded masks $\hat{\mu}_t$, weights ω_t, and accepted rule identifiers $\mathcal{G}_t$.

Harness targets provide gradients without replacing rollout actions:

$$a_t \sim \pi_\theta(\cdot \mid o_t), \qquad s_{t+1} \sim P(\cdot \mid s_t, a_t). \tag{17}$$

Thus, all PPO likelihoods, transitions, and rewards come from the behavior policy. During updates, the same decoder evaluates the Harness sequence under its target prefix:

$$\pi_\theta^{\mathrm{H}}\left(\hat{a}_t^k \mid z_t, \hat{a}_t^{<k}, \hat{\mu}_t^k\right), \tag{18}$$

where H denotes target-prefix conditioning, not a separate model. The weighted hard cross-entropy is

$$\mathcal{L}_{\mathrm{H}}(\theta) = -\frac{\sum_t \sum_{k=1}^{K} \omega_t^k \log \pi_\theta^{\mathrm{H}}\left(\hat{a}_t^k \mid z_t, \hat{a}_t^{<k}, \hat{\mu}_t^k\right)}{\sum_t \sum_{k=1}^{K} \omega_t^k}. \tag{19}$$

If a batch contains no accepted label, $\mathcal{L}_{\mathrm{H}} = 0$; unlabeled slots and zero-weight repair tokens contribute no Harness gradient.

The policy is trained with the standard clipped PPO objective augmented by the weighted Harness loss:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{PPO}} + \lambda \mathcal{L}_{\mathrm{H}}, \tag{20}$$

where PPO uses the rollout likelihood, value regression, and entropy terms. Harness updates only θ; rollouts, rewards, advantages, and value targets remain those of the behavior policy. At inference, target construction and the Harness engine are removed, leaving the multimodal encoder, autoregressive policy, prefix-conditioned masks, and deterministic compiler active.

Table 1: Training and implementation settings.

Category	Setting
Optimization	AdamW; actor LR 1×10^{-6}; critic LR 1×10^{-4}
Adaptation	LoRA rank $r = 8$; $\alpha = 16$
PPO sampling	Rollout: 512 decisions; minibatch: 8; epochs: 1
PPO coefficients	$\gamma = 0.99$; GAE $\lambda = 0.95$; clip $= 0.20$; entropy $= 0.01$; value $= 0.1$
Budget	$128k$ rollout decisions; two NVIDIA RTX 4090 GPUs

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Environment and Training Protocol. The training and implementation settings are summarized in Table 1. We evaluate VLNA in the full-game OpenRA environment. All policies are trained exclusively on `singles`; `ysmir`, `pitfight`, and `sidestep` are reserved for cross-map evaluation. At the start of each episode, the controlled faction is randomized, the opponent uses the built-in game-engine policy, and game speed is fixed at 25 ticks per second. All learned policies use the PPO objective. The pretrained vision-language encoder is adapted with LoRA, whereas the numerical encoder, fusion layers, autoregressive decoder, and critic are trained jointly. The main configuration applies persistent hard-label Harness supervision to key infrastructure and production slots, with $\lambda = 0.1$. Across controlled comparisons, validity masks, compiler, decision frequency, episode limit, parameter budget, training steps, optimizer schedule, and hardware budget are held fixed. Configurations therefore differ only in the modality, action head, or Harness condition under study. Each configuration is trained on `singles` using five independent random seeds, and each seed-level policy is evaluated over 100 games per setting. We report the mean and standard deviation of the five seed-level win rates. Win-conditioned return is averaged over won games, whereas decision steps are reported separately for wins and losses.

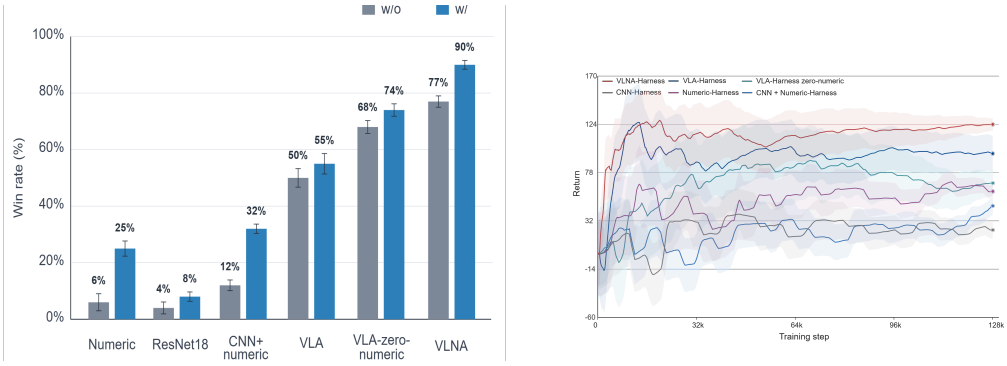
Baselines and Controls. The modality baselines are a numerical-state policy, an image-only ResNet18 policy, their feature-level combination), and a vision-language policy without the numerical branch. The zero-numeric control retains the VLNA numerical encoder and fusion pathway but replaces x_t with zeros. Its comparison with VLNA isolates informative numerical content within the same pathway; comparison with VLA additionally changes the multimodal architecture while preserving the action interface and budgets.

4.2 ARCHITECTURE AND HARNESS ABLATIONS

Fig. 3a compares the architecture variants under matched evaluation conditions. Without Harness supervision, Numeric, ResNet18, CNN+numeric, VLA, VLNA-zero-numeric and VLNA achieved win rates of 6%, 4%, 12%, 50%, 68% and 77%, respectively. Within the matched VLNA pathway, restoring informative numerical state increased the win rate from 68% to 77%.

Harness supervision improved every architecture evaluated in both conditions. The gains were 19, 4, 20, 5, 6, and 13 percentage points for Numeric, ResNet18, CNN+numeric, VLA, VLNA-zero-numeric, and VLNA, respectively. The largest paired gain occurred for CNN+numeric, whereas VLNA achieved the highest absolute win rate of 90%.

Fig. 3b provides complementary optimization evidence. VLNA with Harness maintained the highest late-training return, followed by VLA with Harness. The zero-numeric variant and shallower baselines converged lower. Because the early variability bands overlap broadly, the trajectories provide optimization context but do not replace the final win-rate evaluation.



(a) Final win rates with and without Harness supervision. Error bars show the standard deviation across five seed-level rates.

(b) Training-episode return trajectories over 128k decisions. Lines show five-seed means; shading shows across-seed variability.

Figure 3: Architecture and Harness ablations on `singles`. "w/o" and "w/" denote training without and with Harness supervision, respectively.

Table 2: Harness design ablations on `singles`. "Key" denotes infrastructure and production slots.

Harness variant	Target	Scope	Supervision schedule	Win rate	Win return	Timeout rate
No Harness	–	–	–	77% ± 2.00%	138.50	13%
Triggered Removal	Hard CE	Key	Triggered annealing	80% ± 1.67%	**153.93**	6%
Soft target	Soft CE/KL	Key	Triggered annealing	62% ± 1.79%	153.60	6%
Persistent Harness	Hard CE	Key	Persistent supervision	**90% ± 1.55%**	146.29	**2%**
Expanded scope	Hard CE	All slots	Triggered annealing	12% ± 1.26%	122.82	**2%**

4.3 Harness Target, Scope, and Persistence

As Table 2 shown, under identical scope and capability-triggered removal, hard targets increased the win rate from 62% to 80% compared with soft targets, while yielding similar win-conditioned returns of 153.93 and 153.60, respectively. Persistent supervision further increased the win rate to 90% and reduced the timeout rate from 6% to 2%, indicating the benefit of retaining rule-derived deltas throughout training. In contrast, supervising all slots reduced the win rate to 12%, supporting the restriction of Harness guidance to infrastructure and production decisions.

Table 3: Sensitivity to the Harness coefficient on `singles`.

λ	Win rate (↑)	Win return	Loss steps	Win steps
0.03	64% ± 3.09%	147.42	2262	2081
0.10	**90% ± 1.55%**	146.29	2610	**1366**
0.25	86% ± 2.61%	139.77	2175	1775
0.60	74% ± 2.76%	146.44	2052	1730
0.80	64% ± 1.09%	132.17	3997	3462

4.4 Sensitivity to the Harness Coefficient

As Table 3 shown, the effect of λ was non-monotonic, revealing a trade-off between win rate and win return. Specifically, λ= 0.10 achieved the highest win rate 90% ± 1.55% and the fewest win steps 1,366. Its win return was 146.29. The next-highest win return was 146.44 at λ = 0.60, whereas λ = 0.80 produced the lowest value 132.17. As λ increased from 0.03 to 0.10, the win rate increased from 64% to 90%. Further increases reduced the win rate to 86%, 74%, and 64% at λ = 0.25, 0.60, and 0.80, respectively. This pattern is consistent with the joint objective: moderate Harness supervision complements PPO and improves policy success, whereas stronger supervision introduces a trade-off between win probability and return. We therefore selected λ = 0.10 for persistent Harness.

Table 4: Cross-map evaluation. Bold marks the higher win rate within each map–coefficient pair.

Map	Harness schedule	λ	Win rate (↑)	Win return	Loss steps	Win steps
`ysmir`	Decaying	0.1	$56\% \pm 2.45\%$	147.03	3856	2229
`ysmir`	Persistent	0.1	$\mathbf{62\% \pm 1.79\%}$	149.61	4103	2139
`pitfight`	Decaying	0.1	$96\% \pm 1.26\%$	170.40	1684	1704
`pitfight`	Persistent	0.1	$\mathbf{98\% \pm 0.63\%}$	152.77	5120	1765
`sidestep`	Decaying	0.1	$52\% \pm 2.10\%$	148.43	2228	1819
`sidestep`	Persistent	0.1	$\mathbf{56\% \pm 0.89\%}$	154.15	2372	2080

4.5 Cross-Map Evaluation

As Table 4 shown, at λ=0.1, Persistent Harness improved the win rate on all three unseen maps: from 56% to 62% on ysmir, from 96% to 98% on pitfight, and from 52% to 56% on sidestep. The corresponding gains of 6, 2, and 4 percentage points provide consistent evidence that persistent rule-derived supervision improves cross-map transfer. The stronger-supervision setting, λ=0.8, produced more pronounced improvements; its detailed results are reported in the Appendix A.3. Because win-conditioned return and episode duration did not change consistently across maps, win rate remains the primary transfer metric.

Table 5: Cascaded action-space ablation. Validity is mean±SD; prefix depth spans ten slots.

Variant	Validity	Prefix depth	Native NO-OP	Native command	Win rate
Unconstrained	$0.00\% \pm 0.00\%$	0.41/10	97.28%	2.72%	0%
State only	$0.60\% \pm 1.00\%$	0.06/10	100.00%	0.00%	0%
Semantic prefix	$99.87\% \pm 0.24\%$	9.99/10	72.98%	27.02%	48%
Full cascade	$\mathbf{99.96\% \pm 0.23\%}$	**9.99/10**	**65.63%**	**34.37%**	**77%**

4.6 Cascaded Conditional Action-Space Ablation

As Table 5 shown, unconstrained and state-only generation produced predominantly invalid or no-op outputs and achieved 0% win rate. Semantic-prefix constraints increased validity, native-command execution, and win rate to 99.87%, 27.02%, and 48%, respectively; the full cascade further increased them to 99.96%, 34.37%, and 77%. The improvement from semantic-prefix constraints to the full cascade isolates the contribution of resource- and queue-aware updates, providing evidence that both prefix semantics and within-macro budget consistency jointly improve for executable actions.

5 Conclusion

We introduced VLNA, a structured multimodal policy for full-game RTS control. It multimodal game states to generate semantic macro-actions token, applies prefix-conditioned masks to enforce resource, prerequisite, and production-queue constraints, and compiles valid sequences into OpenRA commands. Gradient-Guided Harness Engineering transfers rule-derived supervision into PPO training, while the Harness engine is removed at inference. The full cascade achieved 99.96% joint validity and a 77% win rate, compared with 99.87% and 48% for semantic-prefix constraints alone. Persistent supervision raised the win rate to 90% ± 1.55% and reduced timeouts from 13% to 2%. These results indicate that structured multimodal representations, constrained decoding, and training-time rule supervision can jointly improve executable and strategic RTS control. However, the evidence is limited to OpenRA Red Alert, evaluation against its built-in opponent policy. Generalization across platforms, game mechanics, opponents, and interfaces therefore remains unestablished. Future work should evaluate VLNA across multiple RTS platforms, maps, and opponent policies, while adapting its action vocabulary, feasibility constraints, and Harness rules to platform-specific mechanics. This would test whether the observed gains reflect reusable principles for multimodal game control or primarily the structure of OpenRA.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish the writing, as described in the paper. We did not use generative AI tools for any other tasks requiring disclosure, and all remaining required-disclosure categories were not applicable to this work. Additionally, we used generative AI tools for literature retrieval and discovery, including identifying related work, as described in the paper. We reviewed all AI-assisted work and revised the generated text for accuracy, clarity, and consistency with the study. We independently verified the relevance and bibliographic accuracy of literature identified with AI assistance by consulting the original sources. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Lei Han, Jiechao Xiong, Peng Sun, Xinghai Sun, Meng Fang, Qingwei Guo, Qiaobo Chen, Tengfei Shi, Hongsheng Yu, Xipeng Wu, and Zhengyou Zhang. TStarBot-X: An open-sourced and comprehensive study for efficient league training in StarCraft II full game. *arXiv preprint arXiv:2011.13729*, 2020. URL https://arxiv.org/abs/2011.13729.

Yueqi Hou, Xiaolong Liang, Jiaqiang Zhang, Qisong Yang, Aiwu Yang, and Ning Wang. Exploring the use of invalid action masking in reinforcement learning: A comparative study of on-policy and off-policy algorithms in real-time strategy games. *Applied Sciences*, 13:8283, 2023. doi: 10.3390/app13148283. URL https://doi.org/10.3390/app13148283.

Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. *The International FLAIRS Conference Proceedings*, 35, 2022. doi: 10.32473/flairs.v35i.130584. URL https://doi.org/10.32473/flairs.v35i.130584.

Shengyi Huang, Santiago Ontañón, Chris Bamford, and Lukasz Grela. Gym-μRTS: Toward affordable full game real-time strategy games research with deep reinforcement learning. In *2021 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE, 2021. doi: 10.1109/COG52621.2021.9619076. URL https://doi.org/10.1109/COG52621.2021.9619076.

Sheng Jia, Jamie Kiros, and Jimmy Ba. DOM-Q-NET: Grounded RL on structured language. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJgd1nAqFX.

Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. Action space shaping in deep reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, pp. 479–486. IEEE, 2020. doi: 10.1109/COG47356.2020.9231687. URL https://doi.org/10.1109/COG47356.2020.9231687.

Muhammad Junaid Khan and Gita Sukthankar. SC-Phi2: A fine-tuned small language model for StarCraft II build order prediction. *AI*, 5(4):2338–2352, 2024. doi: 10.3390/ai5040115. URL https://doi.org/10.3390/ai5040115.

Mohammadsadegh Khorasani, Saber Salehkaleybar, Negar Kiyavash, and Matthias Grossglauser. Hierarchical reinforcement learning with targeted causal interventions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29869–29912. PMLR, 2025. URL https://proceedings.mlr.press/v267/khorasani25a.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Siyuan Li, Lulu Zheng, Jianhao Wang, and Chongjie Zhang. Learning subgoal representations with slow dynamics. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=wxRwhSdORKG.

Zongyuan Li, Yanan Ni, Runnan Qi, Chang Lu, Lumin Jiang, Xiaojie Xu, Xiangbei Liu, Pengfei Li, Yunzheng Guo, Zhe Ma, Huanyu Li, Hui Wu, Xian Guo, Kuihua Huang, and Xuebo Zhang. LLM-PySC2: StarCraft II learning environment for large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/b9523d484af624986c2e0c630ac44ecb-Abstract-Conference.html.

Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete sequential prediction of continuous actions for deep RL. *arXiv preprint arXiv:1705.05035*, 2017. URL https://arxiv.org/abs/1705.05035.

OpenRA Developers and Contributors. OpenRA. Computer software, 2010. URL https://github.com/OpenRA/OpenRA. Continuously developedx.

Zhen-Jia Pang, Ruo-Ze Liu, Zhou-Yu Meng, Yi Zhang, Yang Yu, and Tong Lu. On reinforcement learning for full-length game of StarCraft. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):4691–4698, 2019. doi: 10.1609/aaai.v33i01.33014691. URL https://doi.org/10.1609/aaai.v33i01.33014691.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew 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*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Simon Schmitt, Jonathan J. Hudson, Augustin Žídek, Simon Osindero, Carl Doersch, Wojciech M. Czarnecki, Joel Z. Leibo, Heinrich Küttler, Andrew Zisserman, Karen Simonyan, and S. M. Ali Eslami. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018. URL https://arxiv.org/abs/1803.03835.

Xiao Shao, Weifu Jiang, Fei Zuo, and Mengqing Liu. SwarmBrain: Embodied agent for real-time strategy game StarCraft II via large language models. *arXiv preprint arXiv:2401.17749*, 2024. URL https://arxiv.org/abs/2401.17749.

SIMA Team. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179*, 2024. URL https://arxiv.org/abs/2404.10179.

Aravind Srinivas, Ramnandan Krishnamurthy, Peeyush Kumar, and Balaraman Ravindran. Option discovery in hierarchical reinforcement learning using spatio-temporal clustering. *arXiv preprint arXiv:1605.05359*, 2016. URL https://arxiv.org/abs/1605.05359.

Cheng-Yen Tang, Chien-Hung Liu, Woei-Kae Chen, and Shingchern D. You. Implementing action mask in proximal policy optimization (ppo) algorithm. *ICT Express*, 6:200–203, 2020. doi: 10.1016/j.icte.2020.05.003. URL https://doi.org/10.1016/j.icte.2020.05.003.

Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 4131–4138, 2018. doi: 10.1609/aaai.v32i1.11798. URL https://doi.org/10.1609/aaai.v32i1.11798.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017. URL https://arxiv.org/abs/1708.04782.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P.

Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. doi: 10.1038/s41586-019-1724-z. URL https://doi.org/10.1038/s41586-019-1724-z.

Xiangjun Wang, Junxiao Song, Penghui Qi, Peng Peng, Zhenkun Tang, Wei Zhang, Weimin Li, Xiongjun Pi, Jujie He, Chao Gao, Haitao Long, and Quan Yuan. SCC: An efficient deep reinforcement learning agent mastering the game of StarCraft II. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10905–10915. PMLR, 2021a. URL https://proceedings.mlr.press/v139/wang21v.html.

Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. DOP: Off-policy multi-agent decomposed policy gradients. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=6FqKiVAdI3Y.

Jiechao Xiong, Qing Wang, Zhuoran Yang, Peng Sun, Lei Han, Yang Zheng, Haobo Fu, Tong Zhang, Ji Liu, and Han Liu. Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018. URL https://arxiv.org/abs/1810.06394.

Sijia Xu, Hongyu Kuang, Zhi Zhuang, Renjie Hu, Yang Liu, and Huyang Sun. Macro action selection with deep reinforcement learning in StarCraft. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 15(1):94–99, 2019. doi: 10.1609/aiide.v15i1.5230. URL https://doi.org/10.1609/aiide.v15i1.5230.

Hang Zhao, Qijin She, Chenyang Zhu, Yin Yang, and Kai Xu. Online 3d bin packing with constrained deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 741–749, 2021. doi: 10.1609/aaai.v35i1.16155. URL https://doi.org/10.1609/aaai.v35i1.16155.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning, CoRL 2023*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

APPENDIX

A ADDITIONAL METHOD DETAILS

The main contributions of this work are the VLNA policy optimization scheme, the cascaded prefix-conditioned action space, and the gradient-based harness developed for OpenRA real-time strategy (RTS) scenarios. The experimental system also uses auxiliary reward, memory, and asynchronous sampling mechanisms to mitigate sparse rewards, stabilize decisions under partial observability, and improve the efficiency of online policy optimization. To avoid presenting these implementation choices as algorithmic contributions, this appendix describes them only as components of the training protocol and experimental design.

A.1 Action Vocabulary and Prefix Updates

The decoder emits ten ordered slots: construction intent, building category, building item, economic intent, vehicle production, infantry production, support category, support item, military intent, and target intent. Each slot has a closed candidate set. After decoding, the compiler concurrently executes all valid, non-null commands in the sequence. For slot k, the complete cascade computes the feasible set and prefix state as follows:

$$\mathcal{M}_t^k = \mathcal{B}_t^k(\bar{s}_t) \cap \mathcal{C}_t^k(a_t^{<k}), \qquad (b_t^k, q_t^k) = \Phi(\bar{s}_t, a_t^{<k}), \tag{21}$$

It retains only candidates that pass the deterministic legality check and admit at least one feasible continuation.

A.2 Standard PPO Objective

The PPO likelihood ratio, clipped policy loss, and value loss used in all experiments are

$$\rho_t(\theta) = \frac{\pi_\theta(a_t \mid o_t)}{\pi_{\theta_{\text{old}}}(a_t \mid o_t)}, \tag{22}$$

$$\mathcal{L}_\pi = -\mathbb{E}_t\Big[\min\Big(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big], \tag{23}$$

$$\mathcal{L}_V = \mathbb{E}_t\left[\Big(V_\psi(x_t) - \hat{R}_t\Big)^2\right]. \tag{24}$$

The total objective additionally includes entropy regularization and the local harness-supervision loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_\pi + c_v\mathcal{L}_V - c_e\mathcal{H}(\pi_\theta) + \lambda\mathcal{L}_{\text{H}}. \tag{25}$$

A.3 Cross-Map Evaluation

Table 6: Complete results of the cross-map evaluation.

Map	Harness schedule	λ	Win rate	Win return	Loss/win steps
ysmir	Decaying	0.8	$32\% \pm 1.67\%$	164.09	3699/2511
ysmir	Persistent	0.8	$50\% \pm 1.26\%$	163.84	3807/2650
pitfight	Decaying	0.8	$85\% \pm 1.10\%$	170.01	2438/1622
pitfight	Persistent	0.8	$98\% \pm 0.89\%$	164.28	1949/1678
sidestep	Decaying	0.8	$42\% \pm 2.52\%$	157.43	3294/3040
sidestep	Persistent	0.8	$58\% \pm 1.26\%$	154.10	3410/2663

A.4 Event Rewards for Long-Horizon Credit Assignment

RTS tasks pose a substantial long-horizon credit-assignment problem. Match outcomes are often determined only after thousands of game ticks. The effects of early construction, scouting, production, and defensive decisions propagate through resource accumulation, technology unlocks, unit production, and combat. With terminal outcomes as the only rewards, the policy cannot readily determine which macro-actions improved the game state during early training. We therefore supplement the terminal reward with state-difference event rewards that decompose winning into observable intermediate objectives.

The reward module does not use manually annotated actions. Instead, it extracts interpretable events automatically from consecutive observations o_t and o_{t+1}. Let the feature for event category c be $e_t^c = f_c(o_t)$. Its state difference is

$$\Delta e_t^c = f_c(o_{t+1}) - f_c(o_t). \tag{26}$$

The event reward is

$$r_t^{\text{event}} = \sum_c \alpha_c g_c(\Delta e_t^c), \tag{27}$$

where $g_c(\cdot)$ clips, normalizes, or changes the sign of different event differences, and α_c is the corresponding weight. The immediate reward used by PPO combines the terminal and event terms:

$$r_t = r_t^{\text{terminal}} + r_t^{\text{event}}. \tag{28}$$

The event components follow the main causal chain of an RTS match. Scouting rewards the first enemy sighting, the discovery of new enemy units or structures, and the identification of enemy economic, production, and technology infrastructure. Economic events reward resource growth, increased mining capacity, and the construction of key economic structures such as refineries. Production events reward MCV deployment, the completion of barracks, vehicle factories, and technology structures, and the recovery of a blocked production queue. Combat events reward destroyed enemy units and structures and favorable asset trades, while penalizing losses of friendly units and key structures. Further events characterize technology unlocks, power supply and demand, effective advances, and attack tempo. Because each event is triggered by a change between consecutive states, this mechanism is a state-difference event reward rather than a relabeling of expert actions.

The reward design follows two constraints. First, event rewards provide intermediate guidance but do not replace the match outcome. Terminal wins and losses therefore remain on the dominant reward scale. Wins also receive a time-decayed bonus that favors earlier completion over passive delay. Second, the event reward must remain comparable across decision frequencies. Because the policy acts at macro-action intervals, one decision can span multiple low-level game ticks. We therefore normalize event rewards by the decision interval to prevent systematic scale changes when the decision frequency changes.

This shaping scheme does not prescribe a complete strategy. It provides learnable intermediate gradients for an otherwise sparse match-outcome signal and follows the general principle of dense reward shaping in RTS reinforcement learning.

A.5 Memory Augmentation for Partial Observability

OpenRA is partially observable because the current observation contains only enemy units and structures within the agent's field of view. Previously scouted bases, production facilities, and troop locations can remain strategically relevant after they leave view. RTS actions also have execution delays. The effects of construction or training commands may not appear reliably in the next observation. A policy that relies only on instantaneous observations can therefore forget targets, repeat production commands, or change attack directions too frequently.

To address these issues, the environment maintains a lightweight memory state and exposes it as part of the decision context. Enemy-intelligence memory records the types, locations, quantities, and last-contact times of recently observed enemy units and structures. The policy can thus use historical information for target selection, defensive deployment, and counter-production when the enemy is temporarily hidden. This mechanism approximates the belief state maintained by an RTS player under fog of war and reduces the short-sighted assumption that an unseen entity no longer exists.

Production memory handles command delays and queue uncertainty. After the policy issues a construction or training intent, the system briefly retains that intent. Later observations determine whether it entered the queue, completed, or was blocked by insufficient resources or unmet prerequisites. If the current observation does not yet reflect the command's effect, memory prevents duplicate submissions. If temporary constraints cause a submission to fail, the system can retry after the constraints are resolved. This design reduces interference from low-level interface delays and lets the policy learn the effects of macro-level decisions rather than incidental command-submission failures.

Military decisions additionally use a target-commitment mechanism. Unit movement and combat unfold over time, so repeatedly changing an attack target, rally point, or defensive direction can make units move back and forth without dealing effective damage. Target commitment preserves a military objective for a short interval, reduces high-frequency action oscillation, and gives macro-actions more stable temporal semantics. It does not modify the policy model; instead, it provides temporal consistency that better reflects RTS execution dynamics.

Together, these mechanisms complete the observed state with historical enemy information, pending production intents, and persistent military objectives. They introduce neither additional outcome labels nor replacement decisions. Instead, they reduce the non-Markovian effects of partial observability and action delays.

B HARNESS RULE ENGINE

This appendix section describes the rule engine used to generate auxiliary harness targets during training. The harness does not replace the action sampled by the student policy. Instead, it generates local action proposals that constrain only the action heads associated with currently active rules. The resulting harness targets are used only in the auxiliary harness loss.

B.1 ACTION REPRESENTATION AND LOCAL PROPOSALS

Let the student action at decision step t be represented as a ten-headed action vector:

$$a_t^S = \left(a_{t,0}^S, a_{t,1}^S, \ldots, a_{t,9}^S\right). \tag{29}$$

Conditioned on the current observation o_t, the harness does not directly replace a_t^S. Instead, it generates a set of local proposals. Each proposal is represented as

$$q_j = (r_j, g_j, p_j, \mathcal{U}_j, w_j), \tag{30}$$

where r_j is the rule identifier, g_j is the rule group, p_j is the priority, $\mathcal{U}_j$ is a set of action-head–option assignments, and w_j is the supervision weight, with default value $w_j = 1$.

This representation allows the harness to constrain only the action heads that are relevant to the current state. Action heads associated with rules that are not triggered do not receive artificial pseudo-labels. Consequently, the harness can provide local supervision without imposing a complete action vector at every decision step.

B.2 HARNESS DECISION PROCEDURE

Each soft-stage decision is processed by the harness according to the following procedure.

1. The base action mask is repaired to preserve dependency consistency. In particular, the prefix constraints among building category, building item, and production options are enforced before generating harness proposals.
2. The default rule set is evaluated to generate local proposals. Under the current training configuration, only the `infrastructure` and `production` rule groups enter supervision:
$$\mathcal{G}_{\text{sup}} = \{\text{infrastructure}, \text{production}\}. \tag{31}$$
Proposals from the `economy` and `military` groups are filtered out at this stage.
3. Each proposal is converted into action-head indices. Every proposed assignment is then checked against the current state mask and the conditional prefix constraints. Illegal proposals are discarded and counted in `illegal_count`.
4. The remaining proposals are merged in descending order of priority. If two proposals assign different options to the same action head, the assignment from the higher-priority proposal is retained and one conflict is recorded. If two proposals make the same assignment, the action-head set owned by the accepted rule is extended accordingly.
5. Action heads that are not owned by any accepted rule retain the corresponding student actions as their source. The conditional action mask is then recomputed. If a student option is no longer legal after this repair, it is replaced by `NO_OP` or by the first legal option for that action head. This repair does not generate harness cross-entropy supervision.
6. Under the `key` supervision scope, the harness loss is computed only for action heads explicitly owned by an accepted rule. Under the `full` supervision scope, whenever at least one rule is triggered, a legal canonical action is used to supervise all ten action heads.

The harness action is stored as an auxiliary training target in the rollout buffer and does not modify the action sampled by the student policy.

B.3 Effective Harness Rules

The default rule set contains several rule groups, but only the `infrastructure` and `production` groups are active in the current supervised training configuration. A larger priority value indicates higher precedence during proposal merging. The special option `NO_OP` denotes an explicit request to leave the corresponding action head inactive.

B.3.1 Production Rules

The active production rules are summarized in Table 7.

Table 7: Active production rules and their Harness assignments.

Rule (priority)	Trigger condition	Harness assignment
`combat_mobilization` and subrules Priority 700	An attackable enemy is visible and a recent or persistent threat exists, or enemy units are near the base.	The assignment is determined by the combat-mobilization subrules described below.
`low_funds_harvester` / `low_funds_harvester_reserve` Priority 300	The core chain is released, a refinery is available, funds are below 1400, the harvester count is below its scaled limit, and `harv` is legal.	If affordable: `vehicle_item=harv`; otherwise: `vehicle_item=NO_OP`. In both cases: `structure_intent=NO_OP`; `infantry_item=NO_OP`; `support_production_category=NO_OP`.
`combat_composition` Priority 250	No harvester reservation is active and the planned combat value is below 18000.	Set either `vehicle_item` or `infantry_item` according to the $I_{\text{threshold}} = 5 + 10T$ ratio; set the other production head to `NO_OP`.
`combat_production` Priority 200	No harvester reservation is active and the planned combat value is below 18000.	Set `vehicle_item` to the first legal main battle tank, rocket vehicle, artillery unit, or other vehicle; select `infantry_item` according to the current threat type.

Combat mobilization. Let I denote the planned number of infantry units and let T denote the planned number of main battle tanks. The main battle tank set is

$$\mathcal{T}_{\text{MBT}} = \{4\text{tnk}, 3\text{tnk}, 2\text{tnk}, 1\text{tnk}\}\,. \tag{32}$$

The infantry threshold used by the combat-mobilization rule is

$$I_{\text{threshold}} = 5 + 10T. \tag{33}$$

Once the soft-stage core chain is available, the rule first sets `structure_intent` to `NO_OP`, thereby pausing construction investment during an active combat encounter. If the low-funds harvester rule must preserve the current budget, no additional combat-unit supervision is imposed. Otherwise, the following conditions are applied:

- If $I \geq I_{\text{threshold}}$, the rule selects the first legal main battle tank and sets `infantry_item` to `NO_OP`. If no legal tank is available, both the vehicle and infantry production heads are set to `NO_OP`.
- If $I < I_{\text{threshold}}$, the vehicle production head is set to `NO_OP` and infantry production is preferred. Under enemy vehicle pressure, defined by a remembered vehicle count of at least four or a known enemy war factory, or under air pressure, the infantry candidates are ordered as `e3`, `e1`, `e2`, `e4`. Otherwise, the order is `e1`, `e3`, `e2`, `e4`.

For support production, the harness checks the following fixed priority order:

$$\mathcal{S}_{\text{support}} = \{\text{heli}, \text{hind}, \text{yak}, \text{mig}, \text{ss}, \text{dd}, \text{ca}, \text{pt}, \text{msub}\}\,. \tag{34}$$

If a legal support-production option exists, the first legal item in this order is selected, and action heads 6 and 7 receive additional supervision.

The `combat_composition` rule uses the same $5 + 10T$ infantry–tank ratio but does not impose the construction pause or support-unit constraints used by `combat_mobilization`. The `combat_production` rule only supplements vehicle and infantry production while the planned combat value is below 18000. Once

$$\text{planned_combat_value} \geq 18000,$$

neither rule is triggered.

B.3.2 Infrastructure Rules

The infrastructure rules that contribute positive-weight harness targets are summarized in Table 8.

Table 8: Active infrastructure rules and their Harness assignments.

Rule (priority)	**Trigger condition**	**Harness assignment**
`place_ready` Priority 600	A completed building is ready and can be placed.	`structure_intent=PLACE_READY`; `building_category=NO_OP`; `building_item=NO_OP`.
`power_deficit` Priority 550	The power balance is negative and `powr` is legal.	`structure_intent=BUILD`; `building_category=POWER`; `building_item=powr`.
`core_infrastructure` Priority 500	The core economic or production chain is incomplete or requires recovery, and the target building is legal.	`structure_intent=BUILD`; the category and item are selected from the core infrastructure chain.
`post_attack_economy_expansion` Priority 450	The attack force is ready, the base is safe, funds are at least 2000, and the refinery count is below its target.	`structure_intent=BUILD`; `building_category=ECONOMY`; `building_item=proc`.

The core infrastructure rules follow the order `powr` $\rightarrow$ `proc` $\rightarrow$ `tent`/`barr` $\rightarrow$ `weap` $\rightarrow$ the second `proc`. More precisely, if no power building exists or the current power balance is negative, the target is `powr`. If no refinery exists, the target is `proc`. The rule then fills the barracks, war factory, and second refinery in sequence. If the projected power balance would become negative after constructing the target building, the target is changed to `powr`.

A building that is already in the construction queue is not supervised again. If the core chain makes no progress for 32 consecutive decisions and no recent threat is present, the rules may re-enable construction of a missing core building.

B.4 State Collection and Asynchronous PPO Training

The per-step decision cost increases substantially when a VLNA policy replaces a conventional compact neural policy. Each decision requires structured numerical-state processing, visual-input construction, a vision-language model forward pass, action-token generation, and constrained sampling. Fully serial environment interaction and policy updates would therefore limit training throughput. We use an asynchronous actor-learner training procedure to improve the efficiency of online learning.

During sampling, the actor interacts continuously with the environment using a snapshot of the current policy and collects fixed-length rollout segments. Each segment stores observations, action tokens, action masks, old-policy probabilities, value estimates, rewards, termination indicators, and the required diagnostic information. If an episode remains active at the end of a segment, the current

value function supplies the bootstrap value used to compute advantages and returns. Storing the sampled action tokens, masks, and old-policy probabilities allows the subsequent PPO update to compute policy ratios in the same constrained space used during sampling. This alignment reduces mismatches between the optimization objective and the action-generation process.

The value function is a lightweight critic decoupled from the multimodal actor. For action generation, the actor uses tactical renderings, templated linguistic context, and structured numerical states. The critic receives only normalized structured states and summaries of the environment-side memory. It neither observes images directly nor repeats the vision-language backbone computation.

The critic is separated for two reasons. First, value estimates are queried frequently during rollout collection, bootstrapping, and PPO updates. Repeating image encoding and vision-language inference for each query would substantially increase training cost. Second, PPO value losses can be noisy, and coupling them directly to the vision-language backbone could disrupt action-token generation. The numerical critic instead provides a low-cost, comparatively stable return baseline. Structured states and enemy-intelligence and production memories preserve outcome-relevant information about the economy, production, scouting, and combat. The trade-off is that the value function cannot directly use fine-grained spatial cues from images, so its accuracy depends on the adequacy of the structured-state summary.

During learning, the learner retrieves rollout segments from the buffer and performs PPO updates. It then synchronizes the latest parameters to the actor's policy snapshot. Each segment records the policy version used for sampling to control sample staleness. If a segment is older than the permitted version lag, the learner discards it. This procedure improves sampling throughput while preserving the approximate on-policy assumption required by PPO.

B.5 Complete Training Procedure

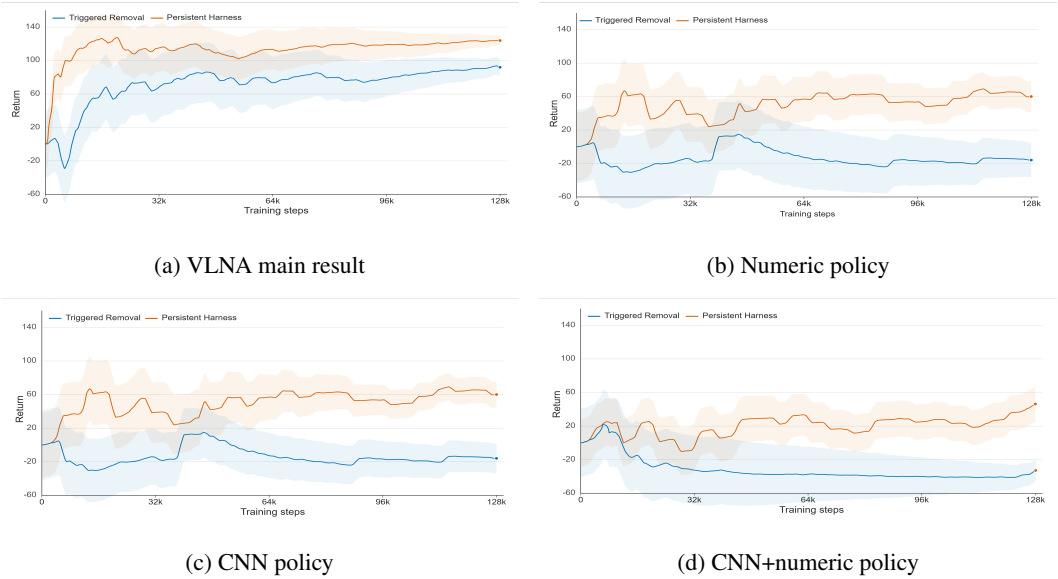
The pseudocode for training Gradient-Based Harness PPO with Persistent Local Rule Supervision is presented below.

Algorithm 1: Gradient-Based Harness PPO with Persistent Local Rule Supervision

Input: Policy model π_θ, value function V_ψ, rule set $\mathcal{R}$, cascaded feasible sets $\mathcal{M}$, and supervision coefficient λ.
Output: Trained policy model π_θ and value function V_ψ.

$u \leftarrow 0$
while training has not terminated **do**
  $B \leftarrow \varnothing$
  for each consecutive environment step t **do**
    Sample $a_t \sim \pi_\theta(\cdot \mid o_t)$
    Query $\mathcal{R}$ with (I_t, L_t, x_t) to obtain proposal set Q_t
    Map proposals to action tokens and reject infeasible proposals
    Merge compatible proposals by priority
    Construct $\hat{a}_t$, $\hat{m}_t$, and ω_t using $\mathcal{M}$
    if the target prefix requires repair **then**
      Insert a feasible token with zero supervision weight
      Recheck all subsequent labels
    end if
    Execute a_t and observe the next state and reward
    Store the policy transition and $(\hat{a}_t, \hat{m}_t, \omega_t, \mathcal{G}_t)$ in B
  end for
  Compute $\mathcal{L}_{\mathrm{PPO}}$ and $\mathcal{L}_{\mathrm{H}}$ from B
  Update θ using $\mathcal{L}_{\mathrm{PPO}} + \lambda \mathcal{L}_{\mathrm{H}}$
  Update ψ through the PPO value term
  $u \leftarrow u + 1$
end while
return (π_θ, V_ψ)

C ADDITIONAL LEARNING-CURVE DIAGNOSTICS



(a) VLNA main result

(b) Numeric policy

(c) CNN policy

(d) CNN+numeric policy

Figure 4: Architecture-level learning-curve diagnostics under Harness supervision. Panel (a) compares Triggered Removal with Persistent Harness for the VLNA main result; panels (b)–(d) compare Persistent Harness with Triggered Removal for the Numerical, CNN, and CNN+numeric policies, respectively. Across architectures, Persistent Harness runs achieve faster recovery from the initial transient, higher sustained returns, and more favorable late-stage trajectories. Solid lines denote the supplied trajectory means, and shaded regions reproduce the supplied uncertainty bands.

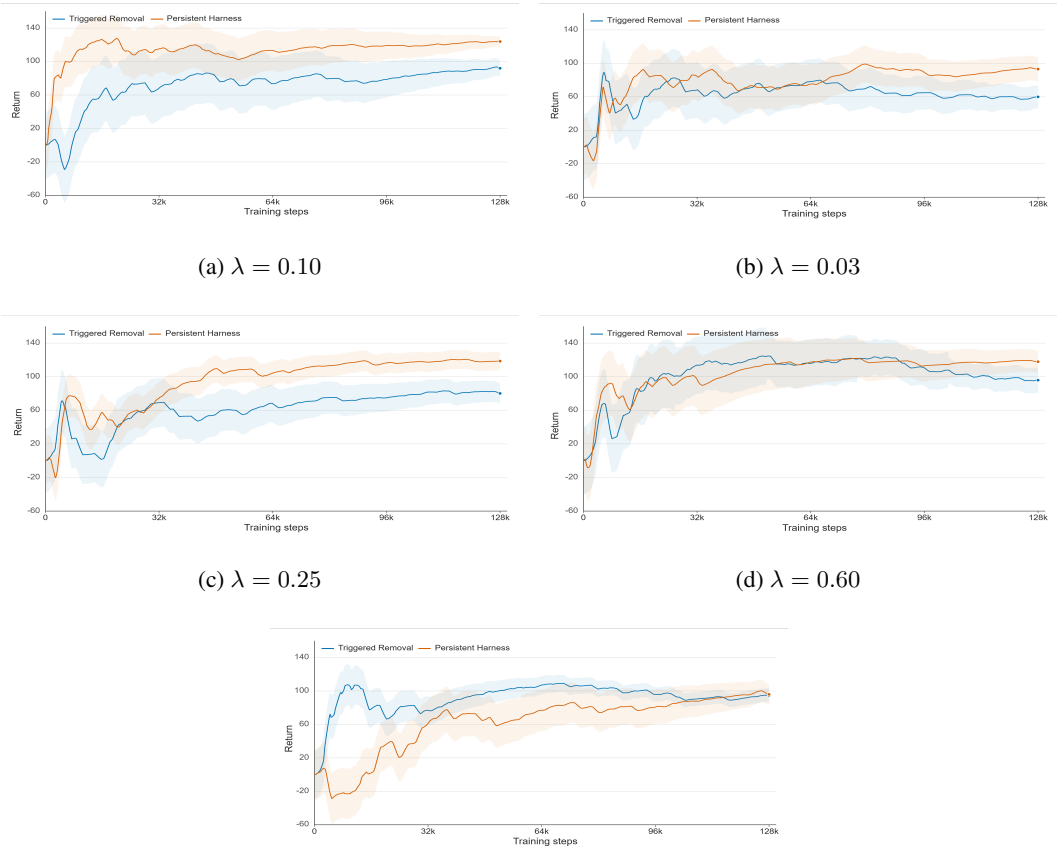
This section reports the learning curves exported from the training runs. Each curve shows the trajectory mean and its accompanying interval over a common 0–128k decision range.

Fig. 4 show that Persistent Harness improves both the return level and late-stage learning behavior across model classes. In the VLNA main-result panel, Persistent Harness rises rapidly and maintains a substantial return advantage over Triggered Removal throughout most of training, indicating that Persistent Harness supervision is particularly effective for the full multimodal policy. The Numerical and CNN panels show a similar pattern: Triggered Removal remains near or below zero for much of training, whereas Persistent Harness maintains substantially higher returns. In the CNN+numeric panel, Persistent Harness also exhibits a clear separation, with a higher mean return and a more favorable late-stage trend than Triggered Removal. Thus, the benefit of Persistent Harness is not restricted to one architecture or representation. It provides a reusable learning signal that accelerates early improvement, prevents prolonged low-return plateaus, and supports more reliable convergence across numerical, visual, and multimodal policies.

C.1 HARNESS-COEFFICIENT SENSITIVITY

Fig. 5 reveal a non-monotonic interaction between the Harness loss weight and its decay schedule. At $\lambda = 0.03$, both schedules recover from the initial instability but converge to moderate returns, indicating that the supervision signal is relatively weak. At $\lambda = 0.10$, the non-decaying schedule achieves a clear and sustained advantage over the decaying schedule, reaching a higher late-stage return. A similar but less pronounced separation is observed at $\lambda = 0.25$, where persistent supervision again provides stronger long-horizon performance. At $\lambda = 0.60$, both schedules converge to relatively high returns and their trajectories become more similar, suggesting that the effect of the decay schedule is reduced at this intermediate coefficient. In contrast, at $\lambda = 0.80$, the decaying schedule provides a stronger early-learning signal, whereas the non-decaying schedule initially performs poorly but gradually catches up. These results show that the Harness coefficient affects not only the final return but also the speed, stability, and timing of learning. Persistent supervision is

most advantageous at $\lambda = 0.10$–0.25, while decay becomes useful for mitigating the optimization burden of strong supervision at $\lambda = 0.80$. The highest coefficient is therefore not uniformly optimal, and coefficient selection should consider both early convergence and late-stage performance.



(a) $\lambda = 0.10$

(b) $\lambda = 0.03$

(c) $\lambda = 0.25$

(d) $\lambda = 0.60$

(e) $\lambda = 0.80$

Figure 5: Coefficient-sensitivity learning curves under decaying and non-decaying Harness supervision. The panels show $\lambda \in \{0.03, 0.10, 0.25, 0.60, 0.80\}$ and illustrate how the supervision coefficient affects both early optimization dynamics and late-stage return.

D Evaluation Trajectories and Exported Data

Fig. 6 summarizes the trimmed mean and supplied spread of army value and total asset value over evaluation decisions for the winning episodes included in the exported summary. These two quantities provide complementary views of in-game development: army value reflects the accumulation of effective combat forces, whereas total asset value captures the broader combination of economic, production, and military resources available to the agent. The policy trained with Gradient-Harness supervision begins to separate from the policy trained without Harness supervision relatively early in the evaluation trajectory. It accumulates army strength and total assets more rapidly, reaches higher intermediate levels during the early-to-mid game, and attains larger peaks in both quantities. The simultaneous separation of the two curves suggests that the advantage is not limited to a short-lived increase in fielded units, but is accompanied by earlier development of the economic and production resources needed to sustain combat operations. In this sense, the trajectories provide a qualitative link between the training signal, earlier force buildup, and the subsequent ability to convert production progress into offensive action. Because Harness supervision is used only during training and the Harness module is removed during both evaluations, the observed difference is produced by the autonomous policies at test time rather than by online teacher intervention. The curves

therefore suggest that the training signal leaves a persistent behavioral effect after the supervision module is removed. Since the summary contains winning episodes and reports trimmed means with the supplied spread, it should be interpreted as a mechanistic and qualitative complement to the multi-episode evaluation statistics, which quantify the overall condition-level effect.

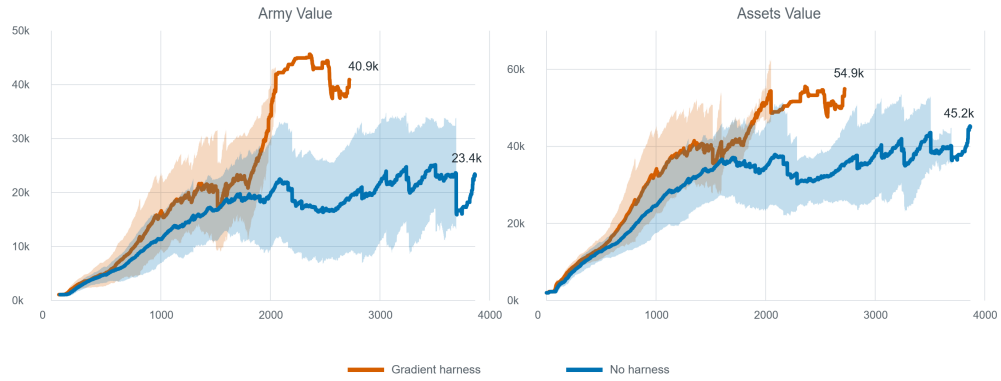
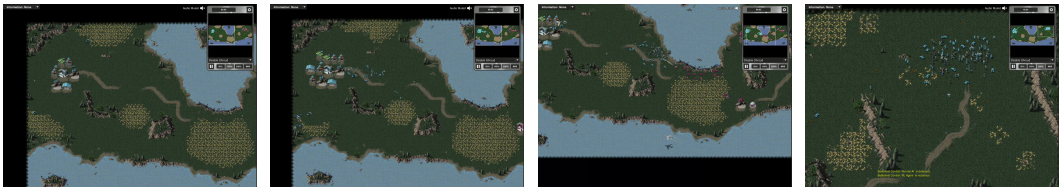


Figure 6: Trimmed mean and supplied spread of army value and total asset value over evaluation decisions. Gradient-Harness and no-Harness trajectories are shown together for the winning episodes included in the exported summary.



(a) Base, 02:56 (b) Force, 06:42 (c) Engagement, 09:05 (d) Victory, 16:24

Harness supervision



(e) Base, 04:03 (f) Assault, 13:19 (g) Engagement, 25:21 (h) Victory, 38:19

no-Harness supervision

Figure 7: Qualitative comparison of recorded OpenRA evaluation replays generated by policies trained with and without Harness supervision. Harness is disabled during both evaluations; the rows differ in the training condition only. Each row shows one episode, and the columns denote phase-based checkpoints: base and economic development, force formation or initial assault, major engagement, and terminal state. The game-clock times are given in the subcaptions. Both terminal panels display the in-game victory message for the RL agent.

To connect the aggregate asset trajectories with realized in-game behavior, Fig. 7 compares two recorded OpenRA test episodes generated after training with the Harness module disabled at inference. The first policy was trained with Harness supervision, whereas the second was trained without it. In the Harness-trained replay, the agent establishes its base at 02:56, expands production and fielded forces by 06:42, reaches a major engagement at 09:05, and wins at 16:24. The policy trained

without Harness supervision also develops a functioning base and eventually wins, but its first clear assault and major engagement occur later, at 13:19 and 25:21, followed by victory at 38:19. Both policies therefore execute without Harness during evaluation, while the selected episodes show a qualitative difference in how quickly economic and production progress is converted into a concentrated offensive force. This behavior is consistent with training-time Harness supervision leaving a persistent effect after the supervision module is removed, although the average condition-level effect is established by the multi-episode evaluation statistics.

E Compositional Macro-Action Vocabulary

The policy uses a task-specific action vocabulary with 90 entries (4 special tokens and 86 action tokens). This vocabulary belongs to the dedicated autoregressive action decoder and is not merged into the VLM natural-language tokenizer. The action entries are partitioned into ten disjoint, head-specific subsets, and each entry uniquely identifies one action-head option. Consequently, a complete macro action is represented by a fixed-length sequence rather than by a single class drawn from the Cartesian product of all head options. Each decoded action contains exactly one token for each of the 10 action heads in Table 9, following the listed order. Action tokens use the form `<ACTION:HEAD:OPTION>`.

Decoding starts from `<BOS>`. At step h, the causally masked Transformer action decoder combines the fused game-state representation, a head-position embedding, and the tokens selected for steps 0 through $h-1$. The output layer scores the shared 90-entry vocabulary, after which a validity mask retains only options that belong to the current head and are compatible with the current game state and previously selected options. This procedure captures dependencies such as selecting a building category before a concrete building item, while preventing invalid cross-head tokens from being emitted. After all ten positions are decoded, the token sequence is mapped deterministically to the structured macro-action tuple.

Every action head includes a neutral alternative, normally `NO_OP`; `KEEP` and `AUTO` serve this role for `economy_intent` and `target_intent`, respectively. These alternatives keep the sequence length fixed when a component is inactive. The remaining special tokens are reserved for sequence handling: `<PAD>` for padding, `<EOS>` for an optional sequence terminator, and `<UNK>` for unknown-token handling; none is an ordinary action-head choice.

Table 9: Fixed decoding order and token allocation for the compositional macro-action vocabulary.

Slot	Action head	Size	Token IDs	Previous slots
0	`structure_intent`	5	4–8	–
1	`building_category`	8	9–16	0
2	`building_item`	22	17–38	0–1
3	`economy_intent`	5	39–43	0–2
4	`vehicle_item`	12	44–55	0–3
5	`infantry_item`	5	56–60	0–4
6	`support_production_category`	4	61–64	0–5
7	`support_production_item`	10	65–74	0–6
8	`military_intent`	7	75–81	0–7
9	`target_intent`	8	82–89	0–8

Table 10: Complete task-specific action vocabulary. Neutral choices are marked in the meaning column.

ID	Token	Meaning
0	`<PAD>`	Reserved padding token
1	`<BOS>`	Autoregressive action-decoder start token
2	`<EOS>`	Reserved end-of-sequence token
3	`<UNK>`	Reserved unknown token
4	`<ACTION:STRUCTURE_INTENT:NO_OP>`	No structure operation (neutral)

Continued on next page

Table 10 – continued from previous page

ID	Token	Meaning
5	<ACTION:STRUCTURE_INTENT:PLACE_READY>	Place a completed structure from the build queue
6	<ACTION:STRUCTURE_INTENT:BUILD>	Start constructing the selected building item
7	<ACTION:STRUCTURE_INTENT:REPAIR>	Repair a damaged friendly structure
8	<ACTION:STRUCTURE_INTENT:DEPLOY_MCV>	Deploy a mobile construction vehicle
9	<ACTION:BUILDING_CATEGORY:NO_OP>	No building category (neutral)
10	<ACTION:BUILDING_CATEGORY:POWER>	Power infrastructure
11	<ACTION:BUILDING_CATEGORY:ECONOMY>	Economic infrastructure
12	<ACTION:BUILDING_CATEGORY:PRODUCTION>	Ground-unit production infrastructure
13	<ACTION:BUILDING_CATEGORY:TECH>	Technology infrastructure
14	<ACTION:BUILDING_CATEGORY:AIR>	Air-production infrastructure
15	<ACTION:BUILDING_CATEGORY:NAVAL>	Naval-production infrastructure
16	<ACTION:BUILDING_CATEGORY:DEFENSE>	Static defense
17	<ACTION:BUILDING_ITEM:NO_OP>	No building item (neutral)
18	<ACTION:BUILDING_ITEM:POWR>	Power Plant
19	<ACTION:BUILDING_ITEM:APWR>	Advanced Power Plant
20	<ACTION:BUILDING_ITEM:PROC>	Ore Refinery
21	<ACTION:BUILDING_ITEM:BARR>	Soviet Barracks
22	<ACTION:BUILDING_ITEM:TENT>	Allied Barracks
23	<ACTION:BUILDING_ITEM:WEAP>	War Factory
24	<ACTION:BUILDING_ITEM:FIX>	Service Depot
25	<ACTION:BUILDING_ITEM:DOME>	Radar Dome
26	<ACTION:BUILDING_ITEM:ATEK>	Allied Tech Center
27	<ACTION:BUILDING_ITEM:STEK>	Soviet Tech Center
28	<ACTION:BUILDING_ITEM:AFLD>	Airfield
29	<ACTION:BUILDING_ITEM:HPAD>	Helipad
30	<ACTION:BUILDING_ITEM:SPEN>	Sub Pen
31	<ACTION:BUILDING_ITEM:SYRD>	Naval Yard
32	<ACTION:BUILDING_ITEM:PBOX>	Pillbox
33	<ACTION:BUILDING_ITEM:HBOX>	Camo Pillbox
34	<ACTION:BUILDING_ITEM:GUN>	Turret
35	<ACTION:BUILDING_ITEM:FTUR>	Flame Tower
36	<ACTION:BUILDING_ITEM:TSLA>	Tesla Coil
37	<ACTION:BUILDING_ITEM:AGUN>	AA Gun
38	<ACTION:BUILDING_ITEM:SAM>	SAM Site
39	<ACTION:ECONOMY_INTENT:KEEP>	Preserve the current harvester assignments (neutral)
40	<ACTION:ECONOMY_INTENT:BALANCE_LOCAL>	Minimize local harvesting cycles
41	<ACTION:ECONOMY_INTENT:BALANCE_GLOBAL>	Balance safe harvesting throughput globally
42	<ACTION:ECONOMY_INTENT:BALANCE_FRONTIER>	Use forward resource fields while accepting additional risk
43	<ACTION:ECONOMY_INTENT:EVACUATE>	Recall threatened harvesters to safe local fields
44	<ACTION:VEHICLE_ITEM:NO_OP>	Do not produce a vehicle (neutral)
45	<ACTION:VEHICLE_ITEM:HARV>	Ore Truck
46	<ACTION:VEHICLE_ITEM:1TNK>	Light Tank
47	<ACTION:VEHICLE_ITEM:2TNK>	Medium Tank
48	<ACTION:VEHICLE_ITEM:3TNK>	Heavy Tank
49	<ACTION:VEHICLE_ITEM:4TNK>	Mammoth Tank
50	<ACTION:VEHICLE_ITEM:JEEP>	Ranger
51	<ACTION:VEHICLE_ITEM:APC>	APC
52	<ACTION:VEHICLE_ITEM:ARTY>	Artillery
53	<ACTION:VEHICLE_ITEM:V2RL>	V2 Rocket Launcher
54	<ACTION:VEHICLE_ITEM:FTRK>	Flak Truck
55	<ACTION:VEHICLE_ITEM:MCV>	MCV
56	<ACTION:INFANTRY_ITEM:NO_OP>	Do not produce infantry (neutral)
57	<ACTION:INFANTRY_ITEM:E1>	Rifle Infantry
58	<ACTION:INFANTRY_ITEM:E2>	Grenadier

Continued on next page

Table 10 – continued from previous page

ID	Token	Meaning
59	<ACTION:INFANTRY_ITEM:E3>	Rocket Soldier
60	<ACTION:INFANTRY_ITEM:E4>	Flamethrower
61	<ACTION:SUPPORT_PRODUCTION_CATEGORY:NO_OP>	No support-production operation (neutral)
62	<ACTION:SUPPORT_PRODUCTION_CATEGORY:AIRCRAFT>	Produce an aircraft
63	<ACTION:SUPPORT_PRODUCTION_CATEGORY:NAVAL>	Produce a naval unit
64	<ACTION:SUPPORT_PRODUCTION_CATEGORY:SET_RALLY_POINT>	Set a production rally point
65	<ACTION:SUPPORT_PRODUCTION_ITEM:NO_OP>	No support-production item (neutral)
66	<ACTION:SUPPORT_PRODUCTION_ITEM:HELI>	Longbow
67	<ACTION:SUPPORT_PRODUCTION_ITEM:HIND>	Hind
68	<ACTION:SUPPORT_PRODUCTION_ITEM:YAK>	Yak
69	<ACTION:SUPPORT_PRODUCTION_ITEM:MIG>	MiG
70	<ACTION:SUPPORT_PRODUCTION_ITEM:SS>	Submarine
71	<ACTION:SUPPORT_PRODUCTION_ITEM:DD>	Destroyer
72	<ACTION:SUPPORT_PRODUCTION_ITEM:CA>	Cruiser
73	<ACTION:SUPPORT_PRODUCTION_ITEM:PT>	Gunboat
74	<ACTION:SUPPORT_PRODUCTION_ITEM:MSUB>	Missile Submarine
75	<ACTION:MILITARY_INTENT:NO_OP>	No military maneuver (neutral)
76	<ACTION:MILITARY_INTENT:SCOUT>	Scout unexplored or strategically useful terrain
77	<ACTION:MILITARY_INTENT:RAID>	Raid a selected enemy objective
78	<ACTION:MILITARY_INTENT:ATTACK_MOVE>	Advance while engaging encountered enemies
79	<ACTION:MILITARY_INTENT:FOCUS_FIRE>	Concentrate attacks on a selected target
80	<ACTION:MILITARY_INTENT:DEFEND_HOME>	Defend the home base
81	<ACTION:MILITARY_INTENT:REGROUP>	Regroup combat units
82	<ACTION:TARGET_INTENT:AUTO>	Let the controller select the default target (neutral)
83	<ACTION:TARGET_INTENT:VISIBLE_THREAT>	Target the currently visible threat
84	<ACTION:TARGET_INTENT:ENEMY_ECONOMY>	Target enemy economic assets
85	<ACTION:TARGET_INTENT:ENEMY_EXPANSION>	Target an enemy expansion
86	<ACTION:TARGET_INTENT:ENEMY_POWER>	Target enemy power infrastructure
87	<ACTION:TARGET_INTENT:ENEMY_PRODUCTION>	Target enemy production infrastructure
88	<ACTION:TARGET_INTENT:ENEMY_BASE>	Target the enemy main base
89	<ACTION:TARGET_INTENT:EXPLORE_ENEMY>	Explore toward the estimated enemy location