

Article

MiniUAV-VLA: A Compact Vision–Language–Action Model for Cooperative Multi-UAV Search and Elimination via MARL Expert Distillation

Hongwei Han ¹, Guanghong Gong ^{1,2} and Ni Li ^{1,2,*}¹ School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China; hanhongwei@buaa.edu.cn (H.H.); ggh@buaa.edu.cn (G.G.)² State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China

* Correspondence: lini@buaa.edu.cn

Highlights

What are the main findings?

- A MARL-to-VLA expert distillation pipeline converts QMIX expert rollouts into multi-modal demonstrations for centralized cooperative multi-UAV control.
- MiniUAV-VLA reaches $74.4 \pm 4.6\%$ mission success. Across five independent training runs, prompt-end action pooling raises mean success from $40.6 \pm 1.9\%$ with last-token pooling to $76.2 \pm 1.7\%$.

What are the implications of the main findings?

- MARL expert policies can provide an efficient source of training data for compact multi-agent VLA controllers without requiring human-collected demonstrations.
- Prompt-end action pooling and structured multimodal state representations offer practical design guidance for compact VLA systems in simulated cooperative UAV control.

Abstract

Coordinating multiple unmanned aerial vehicles (UAVs) for cooperative missions requires agents that perceive their environment, reason about objectives, and generate joint actions. Vision–language–action (VLA) models unify these capabilities but lack a principled source of multi-agent training data and suffer from a training–inference discrepancy in closed-loop control. We propose MiniUAV-VLA, a compact centralized VLA controller for simulated multi-UAV search-and-elimination based on multi-agent reinforcement learning (MARL) expert distillation. A QMIX expert policy achieving 100% mission success generates multimodal demonstrations pairing rendered tactical map images with structured textual state prompts. A 158 M-parameter VLA model with approximately 65 M trainable parameters in the MiniMind-3V backbone and vision projection is fine-tuned with a multi-agent discrete action head that jointly predicts actions for all UAVs in a single forward pass. We identify a training–inference feature mismatch in behavior cloning and address it via prompt-end action pooling, which extracts action-relevant hidden states at the user–prompt boundary rather than after the generated response. In closed-loop evaluation with four drones and six mobile targets averaged over five evaluation seeds, MiniUAV-VLA reaches $74.4 \pm 4.6\%$ mission success against $9.4 \pm 2.1\%$ for a random policy and $16.2 \pm 3.2\%$ for an observation-limited greedy baseline. Across five independent training runs, prompt-end action pooling improves mean closed-loop success from 40.6% to 76.2% over the last-token alternative. These results support MARL expert distillation as a data-efficient route to compact multi-agent VLA control in this simulated setting.



Academic Editor: Diego González-Aguilera

Received: 30 May 2026

Revised: 21 July 2026

Accepted: 24 July 2026

Published: 27 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: vision–language–action model; multi-UAV coordination; multi-agent reinforcement learning; behavior cloning; cooperative control; expert distillation; search-and-elimination

1. Introduction

Cooperative multi-UAV missions that span area search, target detection, and search-and-elimination are central to a broad class of applications including disaster rescue, border patrol, and urban reconnaissance. These missions require a fleet of agents to perceive rich visual environments, reason about shared objectives, and generate coordinated action sequences under partial observability.

Multi-agent reinforcement learning (MARL) has produced strong results on cooperative control tasks. Value decomposition methods such as QMIX [1] and the Value Decomposition Network (VDN) [2] achieve coordinated behavior through centralized training with decentralized execution (CTDE). Policy gradient approaches such as MAPPO [3] further demonstrate scalability across diverse scenarios. However, MARL policies operate on fixed numeric state representations and produce black-box decisions. A language-compatible interface could eventually support operator task specification; however, the present study evaluates only one fixed structured prompt schema, and does not test unrestricted natural-language commands.

Vision–language–action (VLA) models offer an alternative that unifies perception, language-conditioned reasoning, and action generation within a single neural architecture. RT-2 [4] showed that internet-scale vision–language pretraining transfers to robot action prediction. OpenVLA [5] released an open 7B-parameter realization of this paradigm, and π_0 [6] introduced flow-matching action decoders for continuous control. While the language interface creates a route toward natural task specification, such capabilities must be demonstrated for each control setting. However, many existing VLA models are designed for single-robot manipulation and depend on large-scale human demonstration datasets that often contain millions of transitions; none of the cited systems directly addresses the multi-agent aerial coordination setting studied here.

Applying VLA models to multi-agent UAV control raises three fundamental challenges: (i) there is no standard source of high-quality multimodal multi-agent demonstrations; (ii) the joint action space grows quickly with the number of agents and must be represented compactly; and (iii) standard behavior cloning for VLA architectures introduces training–inference feature misalignment that degrades closed-loop performance.

We address these challenges through **MARL-to-VLA expert distillation**. Rather than collecting real-world demonstrations or running expensive online VLA training, we draw on the complementary strengths of both paradigms: MARL excels at acquiring coordinated control strategies through environment interaction, while VLA provides a flexible multimodal interface for grounding those strategies in visual and language context. A trained MARL expert generates the multimodal supervision signal needed for VLA fine-tuning at negligible additional cost.

Our **MiniUAV-VLA** framework makes the following contributions:

1. **MARL-to-VLA distillation pipeline.** We propose a systematic procedure that transforms closed-loop MARL expert rollouts into multimodal VLA training data, bridging cooperative multi-agent control and vision–language reasoning.
2. **Multi-agent action head with prompt-end pooling.** We design a compact action head outputting joint discrete actions for all UAVs in a single forward pass, and identify the hidden-state pooling position as a primary factor governing closed-loop VLA performance. While pooling at a prompt boundary is itself a familiar idea in vision–language

models, we empirically identify prompt-end pooling as an effective remedy for the training–inference feature mismatch that arises when a discrete action head is attached to a generative VLA backbone for multi-agent control trained via behavior cloning.

3. **Structured multimodal state representation.** We construct a combined visual–textual state encoding that pairs rendered tactical map images with compact structured state prompts to support spatial reasoning about target positions, drone coordination, and action legality. We treat the text as a structured state representation; instruction–paraphrase generalization is not claimed.
4. **Empirical validation.** Closed-loop experiments quantify the training seed stability, language modeling objective, legality filter, specified zero-shot environment changes, goal-directed target behavior, and inference cost. Prompt-end pooling improves all five matched training seed pairs and raises mean success rate from 40.6% to 76.2%.

2. Related Work

2.1. Multi-Agent Reinforcement Learning for UAV Coordination

MARL has been widely applied to cooperative UAV tasks. Value decomposition methods decompose the joint action–value function Q_{tot} into per-agent utilities mixed by a monotonic network: QMIX [1] uses a state-conditioned hypernetwork, while VDN [2] assumes additive decomposability. Multi-Agent PPO (MAPPO) and Independent PPO (IPPO) [3,7] apply proximal policy optimization to cooperative settings and have shown strong performance in benchmark environments [8]. Beyond these representative algorithms, cooperative MARL has also been studied through centralized-critic actor–critic methods and value factorization variants such as MADDPG, COMA, QTRAN, and Weighted QMIX [9–12]. Recent multi-UAV DRL studies have further specialized these algorithms for cooperative search-and-detection of static and dynamic targets [13–15], reconnaissance and path planning under partial observability with centralized training–decentralized execution architectures [16,17], and pursuit–cooperative rounding up in cluttered obstacle environments [18,19], with broader surveys covering scalable cooperative DRL approaches for multi-UAV systems [20]. However, MARL policies operate on fixed numeric observation spaces, lack natural language interfaces, and must be retrained from scratch when mission descriptions change.

2.2. Vision–Language–Action Models

VLA models extend vision–language models (VLMs) by adding action generation capabilities. RT-2 [4] showed that a large VLM fine-tuned on robot demonstrations can produce robot actions as language tokens with emergent cross-domain generalization. OpenVLA [5] scaled this approach to open-source models with 7B parameters, while π_0 [6] introduced a flow-matching policy head for continuous action generation. Octo [21] trained a general transformer policy on large cross-embodiment datasets. Related embodied and robot foundation models include RT-1 [22], PaLM-E [23], VIMA [24], and Open X-Embodiment/RT-X [25], which connect transformer policies, embodied multimodal representations, multimodal prompts, and large-scale robot datasets. Recent 2025 systems have extended this frontier in two complementary directions: $\pi_{0.5}$ [26] studied open-world generalization through heterogeneous co-training, whereas SmolVLA [27] focused on affordable training and efficient deployment. These results sharpen the distinction between broad instruction-conditioned manipulation and the compact fixed-schema multi-UAV controller evaluated here. Despite their success, many of these systems target single-robot manipulation and are trained on large demonstration corpora. The cited systems span a wide range of parameter counts, and none directly implement the four-UAV control interface studied here. In the aerial robotics literature, deep reinforcement learning has been used for vision-based single-

UAV navigation and obstacle avoidance from image observations [28]; recent reviews have also begun to position large language models as components of UAV safety and autonomy stacks [29]. However, neither line of work delivers a vision–language–action controller for cooperative multi-UAV control.

2.3. Compact Language and Robotics Models

Compact language models with small parameter footprints have advanced rapidly. Early sub-3B-parameter models such as TinyLlama [30] and MiniCPM demonstrated near-competitive reasoning at this scale, and more recent releases including Phi-3 [31], Gemma-2 [32], and Qwen2.5 [33] have continued to advance the trade-off between quality and model size in the 0.5B–3.8B range. The MiniMind family [34] sits at the small end of this trend, providing compact language and vision–language backbones with tens of millions of trainable parameters that are practical for single-GPU experimentation. For robotics, ACT [35] and Diffusion Policy [36] demonstrate compact sensorimotor control but lack language grounding. Our work targets the gap between compact architectures and multi-agent VLA control.

2.4. Behavior Cloning and Policy Distillation

Behavior cloning [37] trains a policy to imitate expert actions via supervised learning. DAgger [38] addresses distribution shift through iterative expert querying. Policy distillation [39] transfers knowledge from a teacher policy to a student by training on teacher outputs. In sequence modeling, a related training–inference feature mismatch arises when a model trained with ground-truth right-context is run without that context at inference [40]; this parallels the right-context shortcut we identify in VLA behavior cloning. Our prompt-end pooling directly resolves this mismatch without requiring online interaction or iterative relabeling.

2.5. Research Gap and Positioning

The literature reviewed above advances three largely disjoint threads. Cooperative MARL splits into value-decomposition methods (QMIX [1], VDN [2]) and policy gradient methods (MAPPO [3], IPPO [7]); both deliver coordinated multi-UAV control, but operate on fixed numeric observations and provide no language interface.

Vision–language–action models develop along complementary lines that collectively offer language-grounded behavior: discrete-token policies trained on internet-scale data (RT-2 [4]), open-source 7B-parameter realizations (OpenVLA [5]), flow-matching action decoders for continuous control (π_0 [6]), and cross-embodiment transformer policies (Octo [21]). Prominent generalist VLAs often combine large demonstration corpora with billion-scale backbones, although RT-1, Octo, SmolVLA, and other compact systems show that neither data scale nor parameter count is universal. Compact robotics policies such as ACT [35] and Diffusion Policy [36] achieve small-footprint sensorimotor control but lack language grounding, while behavior cloning treatments of the training–inference mismatch [38,40] target text generation rather than action prediction from a hybrid vision–language context. Recent multi-agent LLM frameworks such as MetaGPT [41] and AgentVerse [42] coordinate multiple language model agents at the dialogue/tool use level, but do not close a perception–action loop in a physical or simulated environment, leaving the multi-agent closed-loop VLA-BC setting we target unaddressed.

Consequently, no prior framework jointly delivers a *compact, structured-text-conditioned, cooperative multi-UAV* controller while controlling for the training–inference feature mismatch that arises when a discrete action head is attached to a generative VLA backbone. MiniUAV-VLA is designed to close exactly this gap in the simulated setting we study,

and the empirical investigation in Section 4 quantifies the contribution of each design choice that the gap demands.

3. Materials and Methods

This study was conducted entirely in simulation and used no chemicals, biological materials, commercial UAV hardware, or physical flight-test instruments. Model training and evaluation were implemented in PyTorch (PyTorch Foundation, San Francisco, CA, USA) using the open-source EPyMARL framework (University of Edinburgh, Edinburgh, UK). The MiniMind-V and SigLIP2 model components are identified by their citations and public release links below; SigLIP2 was released by Google LLC. (Mountain View, CA, USA).

3.1. Task Formulation

We consider a cooperative multi-UAV task in a 2D grid environment, formally modeled as a **Decentralized Partially Observable Markov Decision Process (Dec-POMDP)** [43] of the form given in Equation (1):

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, N \rangle, \quad (1)$$

where $\mathcal{S}$ is the global state space, $\mathcal{O}$ is the joint observation space, $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_N$ is the joint action space for $N = 4$ drones, $\mathcal{T}$ is the stochastic transition function, $\mathcal{R}$ is the shared team reward, and γ is the discount factor.

Environment. The Drones environment is a 30×30 grid with randomly placed walls and trees (obstacle probability $p = 0.01$). Four drones are initialized randomly at episode start. Six targets are placed randomly and follow a stochastic drone-aware movement policy. When a target is outside the view range of every drone, it stays in place with probability 0.2; otherwise, it performs a uniform random move (probability 0.8). When a target is within the view range of at least one drone, it flees from the nearest drone with probability $p_{\text{flee}} = 0.6$, stays in place with probability 0.2, and performs a random move with probability 0.2. Each drone has five discrete actions: $\mathcal{A}_i = \{0:\text{UP}, 1:\text{DOWN}, 2:\text{LEFT}, 3:\text{RIGHT}, 4:\text{STAY}\}$. Episodes terminate on mission completion or after $T_{\text{max}} = 120$ steps.

This environment represents centralized high-level cell-to-cell decision making rather than low-level flight control. It contains no altitude, attitude, continuous velocity, acceleration, or aerodynamic state; consequently, the reported results must not be interpreted as validation of a flight controller or 3D vehicle dynamics.

To test a more purposeful target policy without changing the controller, we also introduce a goal-directed evasive condition for evaluation. At reset, each target receives a unique reachable goal cell that is hidden from both VLA input channels. Before discovery, it follows a shortest legal path toward that goal with probability 0.8, moves randomly with probability 0.1, and stays with probability 0.1. After discovery, it continues toward the goal with probability 0.5, evades the UAVs with probability 0.3, moves randomly with probability 0.1, and stays with probability 0.1. Target goal completion is recorded as an additional metric. The targets do not attack or disable UAVs.

Mission. The environment refers to the benchmark as the “eliminate” task. A target must first enter the nominal view range r_v of a drone, which changes its environmental status to found, and must then enter attack range r_a before it is eliminated. Baseline training and evaluation use $r_v = 6$ and $r_a = 4$.

Reward. The team reward at each step is defined by Equation (2):

$$r_t = r_{\text{found}} \Delta_{\text{found}} + r_{\text{elim}} \Delta_{\text{elim}} + r_{\text{suc}} \mathbb{I}\{\text{success}\} + r_{\text{col}} n_{\text{col}} + r_{\text{ovlp}} n_{\text{ovlp}} + r_{\text{step}}, \quad (2)$$

where Δ_{found} and Δ_{elim} are incremental discovery and elimination counts per step, n_{col} is the number of drone–drone collision events at the current step, and n_{ovlp} is the number of

ordered drone pairs with overlapping view ranges; the view range overlap term discourages redundant area coverage by neighboring drones. Default values: $r_{\text{found}} = 0.5$, $r_{\text{elim}} = 5.0$, $r_{\text{suc}} = 50.0$, $r_{\text{col}} = -0.2$, $r_{\text{ovlp}} = -0.02$, $r_{\text{step}} = -0.01$.

VLA control policy. The Dec-POMDP formalism above describes the underlying environment and the per-agent observation structure used by the QMIX teacher (Section 3.2). MiniUAV-VLA itself is a *centralized team-level policy* distilled from the QMIX expert: at each timestep t , a single centralized VLA policy π_θ receives a tactical image $I_t \in \mathbb{R}^{3 \times 224 \times 224}$ together with a structured text prompt x_t summarizing the state of all four drones, then jointly predicts actions for the fleet in a single forward pass via the mapping in Equation (3). We do not claim or evaluate decentralized onboard execution; extending the framework toward decentralized inference is left to future work.

$$\mathbf{a}_t = (a_t^1, a_t^2, a_t^3, a_t^4) = \pi_\theta(I_t, x_t), \quad a_t^i \in \{0, 1, 2, 3, 4\} \quad (3)$$

3.2. QMIX Expert Training

We train a QMIX policy π_Q as the teacher for VLA distillation. QMIX [1] implements CTDE by decomposing the joint action–value function into a monotonic mixing of per-agent utilities, as in Equation (4):

$$Q_{\text{tot}}(\mathbf{s}, \mathbf{a}) = f_{\text{mix}}(Q_1(o_1, a_1), \dots, Q_N(o_N, a_N); \mathbf{s}), \quad (4)$$

where each Q_i is a Gated Recurrent Unit (GRU)-based recurrent network taking local observation o_i and f_{mix} is a hypernetwork with non-negative weights conditioned on the global state $\mathbf{s}$.

Each agent’s observation includes its own position, the relative positions of teammates and the nearest currently discovered active targets, obstacle features within the view range, shared discovery/elimination status, and a time ratio. Undiscovered target locations are not provided in the teacher observation. The trained QMIX expert achieves 100% mission success, eliminating all six targets in 48.4 ± 0.8 steps on average across the five evaluation seeds.

3.3. VLA Dataset Construction

Given the trained expert π_Q , we collect a multimodal dataset $\mathcal{D} = \{(I_t, x_t, \mathbf{a}_t^{\text{exp}})\}$ by rolling out the expert in closed loop and retaining only transitions from successful episodes.

Tactical map rendering. At each timestep, a 224×224 RGB image encodes spatial information: wall cells (black), tree cells (dark green), empty cells (white), drone positions as color-coded circles with agent IDs (D0–D3 in blue, cyan, purple, and yellow), target positions, view rings (blue), and attack rings (red).

Structured state prompt. The prompt x_t encodes the current state in a compact format:

```
Mode=eliminate;vr=6;ar=4;map=30;t={t}/120;
A=0U1D2L3R4S;N=4;
D={drone positions};
F={found}/6;E={eliminated}/6;
M={legal action mask per drone}; return 4 actions.
```

The legal action mask M is a 5-bit binary string per drone, where bit $i = 1$ indicates that action i avoids obstacles.

Target response. The training target is a structured JSON-like string:

```
{“thinking”: “Follow the QMIX expert policy to find all targets
while avoiding blocked cells.”,
“actions”: [a0, a1, a2, a3]}
```

where the actions are the expert's joint actions at that timestep. An audit of all training records found exactly one unique thinking string, consisting of the fixed rule template quoted above. Zero records contained an explicit direction/action token in this field, and the action array matched the stored expert action in every audited record. QMIX produces the action labels only, and does not produce a language explanation. Because prompt-end pooling extracts a causal hidden state before all assistant tokens, the action head cannot read the thinking text or the textual action labels. Closed-loop inference does not generate or consume an assistant response; the complete response and its language modeling loss are training-only auxiliary supervision. The thinking field and language modeling loss are included by design to encourage the backbone to retain a semantic representation of the multimodal state during training rather than collapsing into a pure action predictor; however, we make no claim that the policy thinks in language at inference or that this field is an interpretable rationale. Section 4.5 directly tests its empirical contribution.

Our main training set contains 5000 transitions; the ablation studies additionally use a 1k subset and a 10k superset drawn from the head of the training split.

3.4. MiniUAV-VLA Architecture

The MiniUAV-VLA model consists of three components: a vision encoder, a language model backbone, and a multi-agent action head.

Vision encoder. We adopt SigLIP2-base-p16-224 [44] for two concrete reasons. First, its sigmoid contrastive objective yields stronger zero-shot dense-feature transfer than the softmax-based Contrastive Language-Image Pretraining (CLIP) family at a comparable parameter budget, which matters because the encoder is frozen throughout training and must supply usable spatial features without any task-specific gradient flow. Second, SigLIP2 augments the contrastive objective with self-distillation and dense local feature prediction during pretraining, which strengthens patch-level spatial representations. This is useful for a frozen encoder that must localize drone and target positions on the rendered tactical map. The encoder emits $14 \times 14 = 196$ patch tokens at 16×16 patches over a 224×224 input, well within the 512-token budget of the MiniMind-3V backbone after accounting for the structured prompt and assistant response. The 196 patch embeddings are projected to the language model's hidden dimension by a two-layer MLP with Gaussian Error Linear Unit (GELU) activation, structured as LayerNorm $\rightarrow$ Linear $\rightarrow$ GELU $\rightarrow$ Linear.

Language model backbone. The backbone is MiniMind-3V [34], an eight-layer causal transformer with hidden dimension $d = 768$, eight attention heads, four key-value heads, and RoPE positional embeddings. The vocabulary size is 6400 and the maximum sequence length is 512 tokens. The trainable backbone comprises approximately 65 M parameters, with about 64 M in the language model and 1 M in the vision projection. The SigLIP2 vision encoder has ≈ 93 M parameters, matching the official SigLIP2-base-patch16-224 release (<https://huggingface.co/google/siglip2-base-patch16-224>, accessed on 26 July 2026), and is frozen throughout training, making for a total inference footprint of ≈ 158 M parameters.

Multi-agent action head. A single linear layer (Equation (5)) maps the pooled hidden state $h_{\text{pool}} \in \mathbb{R}^d$ to joint action logits via a flat output that is subsequently reshaped:

$$\hat{\mathbf{I}}_{\text{flat}} = W_{\text{act}} h_{\text{pool}} + b \in \mathbb{R}^{N|\mathcal{A}|}, \quad \hat{\mathbf{I}} = \text{reshape}(\hat{\mathbf{I}}_{\text{flat}}, N, |\mathcal{A}|) \in \mathbb{R}^{N \times |\mathcal{A}|}, \quad (5)$$

where $N = 4$, $|\mathcal{A}| = 5$, and $W_{\text{act}} \in \mathbb{R}^{(N \times |\mathcal{A}|) \times d}$. The 20 logits are reshaped as four consecutive five-action blocks, one block per drone: logits 0–4 correspond to drone 0, logits 5–9 to drone 1, and so on. The head adds only $768 \times 20 = 15,360$ parameters. The predicted action for drone i at timestep t is $\hat{a}_t^i = \arg \max_j \hat{I}_t^i$.

The overall architecture is illustrated in Figure 1.

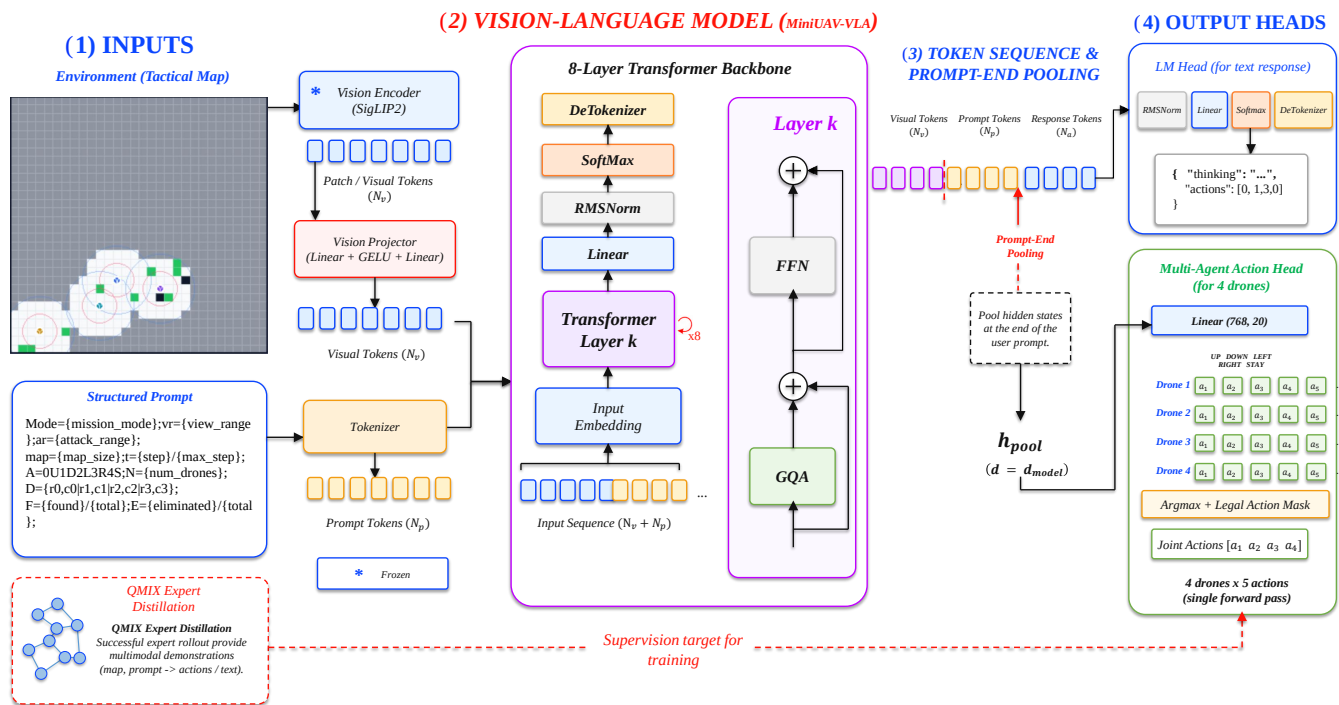


Figure 1. MiniUAV-VLA architecture. The SigLIP2 vision encoder processes the rendered tactical map image, the MiniMind-3V transformer backbone generates hidden states, and the multi-agent action head predicts joint actions for all four drones in a single forward pass via prompt-end hidden state pooling. The asterisk denotes the frozen vision encoder, and ellipses denote omitted intermediate tokens or repeated action entries.

3.5. Prompt-End Action Pooling

A central design choice is *where* in the token sequence to extract the hidden state h_{pool} for action prediction.

Problem. During supervised fine-tuning (SFT), the input sequence is [vision tokens] [prompt tokens] [assistant response tokens]. If h_{pool} is extracted at the *end of the assistant response* (last-token pooling), the action head attends to the ground-truth action text during training, which provides a right-context shortcut. At inference time, however, no response tokens are present, so the model must predict from the prompt alone. This training–inference feature mismatch makes the action head learn representations that are valid only when ground-truth responses are visible, which severely degrades closed-loop performance.

Solution. We extract h_{pool} as the hidden state at the *last prompt token position* (just before the assistant response begins), as in Equation (6):

$$h_{\text{pool}} = H_{T_v+T_p}, \quad (6)$$

where $H \in \mathbb{R}^{L \times d}$ is the hidden state matrix output by the final transformer layer, T_v is the number of vision tokens, and T_p is the number of prompt tokens, so that $H_{T_v+T_p}$ is the hidden state at the final token of the user prompt. The Last-Token baseline in Ablation A1 uses the same single-token extraction operator but applies it at the end of the assistant response rather than the end of the prompt; therefore, the two configurations differ only in the extraction position, as illustrated in Figure 2.

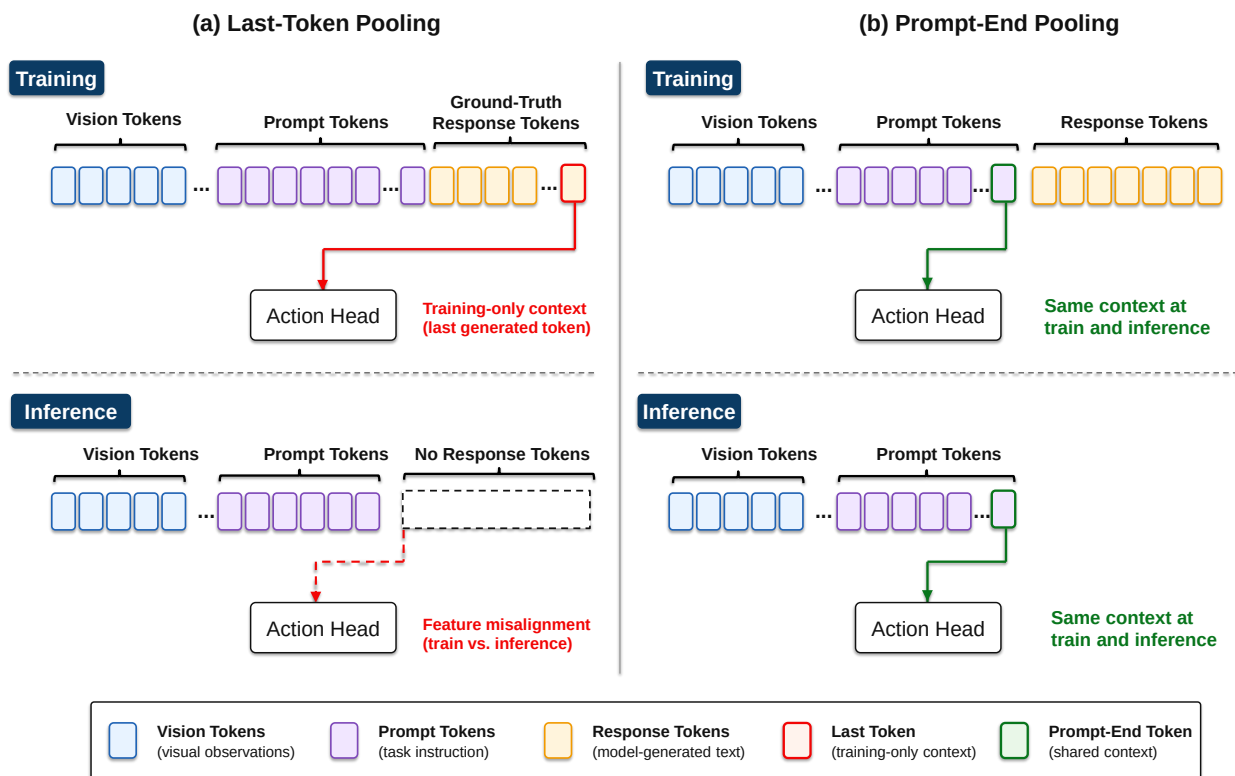


Figure 2. Comparison of action head pooling strategies. **(a)** Last-token pooling attends to the ground-truth response text during training but has no response tokens available at inference, causing a feature mismatch. **(b)** Prompt-end pooling extracts hidden states at the user–prompt boundary, ensuring identical context at training and inference time. Across five matched training seeds, prompt-end pooling raises mean closed-loop mission success from $40.6 \pm 1.9\%$ to $76.2 \pm 1.7\%$, where the variation is the sample standard deviation between independently trained checkpoints. Every matched pair improves, indicating that training–inference feature alignment is an important factor in this VLA behavior cloning setup. Ellipses in the diagram denote omitted intermediate tokens.

3.6. Training Objective

The model is trained with the multi-task loss in Equation (7), which combines language modeling and action prediction:

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda \mathcal{L}_{\text{action}}, \quad (7)$$

where the language modeling loss (Equation (8)) is computed over assistant response tokens only:

$$\mathcal{L}_{\text{LM}} = - \sum_{t \in \mathcal{T}_{\text{resp}}} \log p_{\theta}(y_t | y_{<t}, I, x) \quad (8)$$

and the action prediction loss (Equation (9)) is the mean cross-entropy over all drones, computed per transition $(I_t, x_t, \mathbf{a}_t^{\text{exp}}) \in \mathcal{D}$ and averaged over the minibatch:

$$\mathcal{L}_{\text{action}} = \frac{1}{N} \sum_{i=1}^N \text{CE}(\hat{l}_t^i, a_t^{i,\text{exp}}). \quad (9)$$

We set $\lambda = 1.0$. Training used the AdamW optimizer with a cosine learning rate schedule ($\text{lr}_0 = 10^{-5}$), batch size of 8, gradient clip norm of 1.0, and 10 epochs. All MiniMind-3V transformer blocks, the vision projection MLP, and the multi-agent action head were trained end-to-end (≈ 65 M trainable parameters in total); the small learning rate combined with

the relatively short schedule preserves the pretrained linguistic representations without explicit layer freezing. The SigLIP2 encoder weights were frozen throughout.

For the auxiliary-objective ablation (Section 4.5), the No-thinking variant retains the language modeling loss but its response contains only the action array, while the Action-only variant removes the response supervision and optimizes $\mathcal{L}_{\text{action}}$ alone. The architecture, 5k training set, training seed, optimizer, and schedule are otherwise unchanged.

3.7. Closed-Loop Evaluation Protocol

At each timestep, the evaluation protocol performs the following: (1) render the tactical map image I_t ; (2) construct the prompt x_t ; (3) compute h_{pool} via prompt-end pooling; (4) obtain action logits $\hat{I}_t$ from the action head and take the arg-max action for each drone; (5) if the arg-max action for drone i is illegal under the environment's legality constraints (collision with obstacles or out-of-map), a *safe-actions* post hoc correction selects the legal action with the highest logit, i.e., $\arg \max_{j \in \mathcal{L}_i} \hat{I}_j^i$ where $\mathcal{L}_i$ is the set of legal actions for drone i ; and (6) execute the resulting (always-legal) joint action in the environment. The safe-actions fallback is mathematically equivalent to masking illegal logits to $-\infty$ before the arg-max.

We report five metrics: mission success rate (the fraction of episodes in which all targets are eliminated); found ratio (the average fraction of targets detected per episode); average episode length in steps; average cumulative return; and raw illegal action rate (the fraction of *raw* arg-max predictions that are illegal *before* the safe-actions correction reroutes them to a legal fallback). This last metric directly diagnoses how well the action head has learned to avoid obstacles rather than providing a measure of actions wasted in the environment, since every action executed in the environment is legal by construction.

To isolate this safety mechanism, we additionally evaluate the same checkpoint without post hoc correction. In that condition, the raw arg-max is submitted to the unchanged environment. An obstacle or boundary violation is blocked in place, consumes a timestep, and retains the environment's penalty; "without filter" never means that the UAV can pass through an obstacle. Because the two execution modes subsequently visit different states, their raw-illegal rates need not be identical.

We distinguish two sources of variation. For ordinary single-checkpoint experiments, values are mean $\pm$ standard deviation across five fixed evaluation environment seeds (100 episodes per seed). For pooling stability, each checkpoint is first averaged over those 500 episodes and the reported mean $\pm$ sample standard deviation is then computed across five independently trained checkpoints. The auxiliary-objective variants use one matched training seed, so we report descriptive point differences across the five evaluation-seed blocks and make no training-level significance claim. For the five pooling pairs, an exact two-sided paired sign test gives $p = 0.0625$; this small sample does not support a conventional significance claim, so we report every paired effect rather than adding a significance star.

Inference efficiency was measured at a batch size of 1 on an NVIDIA GeForce RTX 4090 graphics processing unit (NVIDIA Corporation, Santa Clara, CA, USA) over 1000 iterations after 100 warm-up iterations. The timed boundary includes prompt construction and tokenization, 224×224 rendering, SigLIP preprocessing and encoding, the VLM/action head, and CPU legality correction; it excludes environment stepping and disk/video I/O.

4. Results

4.1. Experimental Setup

Unless stated otherwise, all experiments use the Drones environment with 30×30 maps, four drones, six mobile targets, and a 120-step episode limit. The fleet size remains fixed at

four in every experiment. Unless stated otherwise, the “eliminate” task is used ($r_v = 6$ and $r_a = 4$). Each trained checkpoint is evaluated over five random *evaluation* seeds {2026, 3026, 4026, 5026, 6026}, with 100 independent episodes per seed (500 episodes per condition in total). Unless otherwise stated, results are reported as mean $\pm$ standard deviation across these five evaluation seeds; therefore, the reported standard deviation reflects environment and rollout stochasticity, not training-run variance. Complete environment and training hyperparameters are provided in Appendix A, and the full structured prompt format is provided in Appendix B.

Baselines. We compare MiniUAV-VLA against five baselines: two non-learned heuristics, two vision-only learned ablations (one frozen-backbone, one backbone-trainable with a comparable trainable parameter budget to MiniUAV-VLA), one text-only learned ablation, and the QMIX teacher.

- *Random policy*: At each step, each drone samples a uniformly random legal action independently.
- *Greedy heuristic*: Each drone navigates toward the nearest active target within the environment-defined view range. When no target is visible, the drone performs a random legal action. This observation-limited design provides a heuristic reference; we do not claim that it is the strongest possible hand-crafted coordination strategy.
- *Vision-only ResNet-18 (frozen) + MLP head* (vision-only, no language interface): A frozen ImageNet-pretrained ResNet-18 backbone (≈ 11 M parameters, the final fc layer removed) produces a 512-dimensional feature via global average pooling, then a two-layer MLP head (Linear(512, 256) $\rightarrow$ GELU $\rightarrow$ Dropout(0.1) $\rightarrow$ Linear(256, 20)) maps it to joint action logits reshaped to (4, 5). Only the ≈ 136 K MLP head parameters are trainable; the ResNet-18 weights remain frozen throughout. The input is only the 224×224 rendered tactical map; the structured prompt is removed entirely. Training uses the same 5k expert transitions with the same AdamW optimizer, batch size, and 10-epoch schedule; only the action loss is used (no LM loss). Evaluation follows the same protocol as MiniUAV-VLA. This baseline provides a vision-only reference point with a small trainable head on top of strong frozen ImageNet features.
- *Vision-only CNN (ConvNeXt-Small, trainable)* (vision-only, backbone-trainable, no language interface): A fully trainable ConvNeXt-Small backbone (ImageNet-22k pre-trained init, ≈ 50 M parameters) produces a 768-dimensional feature via global average pooling, projected to action logits by a LayerNorm $\rightarrow$ Linear(768, 768) $\rightarrow$ GELU $\rightarrow$ Dropout(0.1) $\rightarrow$ Linear(768, 256) $\rightarrow$ GELU $\rightarrow$ Linear(256, 20) head reshaped to (4, 5). Total trainable capacity is ≈ 51 M, on the same order as the ≈ 65 M trainable parameters of the MiniUAV-VLA backbone. Input, training data, optimizer, schedule, and evaluation are identical to the frozen variant. Together with the frozen ResNet-18, this provides a second vision-only reference point with a comparable trainable parameter budget to MiniUAV-VLA.
- *Text-only LLM policy* (no vision channel): The MiniMind-3V backbone (eight layers, $d = 768$; same no-freeze end-to-end fine-tuning protocol as MiniUAV-VLA) consumes only the structured prompt. The SigLIP2 vision encoder and the vision projection are removed entirely, not merely masked. Prompt-end pooling and the multi-agent action head (Linear(768, 20) reshaped to (4, 5)) are identical to MiniUAV-VLA. The structured prompt is otherwise unchanged. Training uses the same 5k transitions with the same hyperparameters. Because this baseline shares the MiniMind-3V backbone with MiniUAV-VLA, the gap to MiniUAV-VLA is a backbone-matched reference for the contribution of the visual channel, and addresses the natural question of whether the legal-action mask and discovery counts already provided by the prompt are sufficient on their own. All learned baselines use the same training protocol for controlled comparison.

- *QMIX expert*: The teacher policy used to generate training data, evaluated in the same closed-loop setting.

4.2. Main Results

Table 1 compares MiniUAV-VLA against all baselines on 500 eliminate-task episodes per condition (100 episodes for each of five evaluation seeds).

Table 1. Closed-loop performance comparison on the eliminate task (100 episodes per seed across five seeds, $r_v = 6$ and $r_a = 4$, four drones, six targets; values reported as mean $\pm$ standard deviation across seeds). The **Steps** column is the mean episode length across *all* episodes (failed episodes contribute the timeout value $T_{\max} = 120$), so longer steps reflect both slower completion and a higher failure rate.

Method	Inference Params	Success (%)	Found (%)	Steps	Return	Raw Illegal (%)
Random Policy	—	9.4 $\pm$ 2.1	61.0 $\pm$ 3.3	115.9 $\pm$ 1.3	14.13 $\pm$ 2.43	—
Greedy Heuristic	—	16.2 $\pm$ 3.2	70.4 $\pm$ 2.2	113.1 $\pm$ 1.1	17.40 $\pm$ 2.55	—
Vision-only ResNet-18	11 M	14.4 $\pm$ 3.4	76.7 $\pm$ 2.3	112.3 $\pm$ 1.8	23.29 $\pm$ 2.75	17.2 $\pm$ 1.1
Vision-only CNN	51 M	34.4 $\pm$ 5.6	77.3 $\pm$ 2.1	101.3 $\pm$ 2.1	22.33 $\pm$ 2.12	32.2 $\pm$ 0.7
Text-only LLM	64 M	29.6 $\pm$ 5.0	63.4 $\pm$ 0.8	114.5 $\pm$ 1.2	24.46 $\pm$ 1.34	12.4 $\pm$ 1.3
MiniUAV-VLA + filter (Ours)	158 M	74.4 $\pm$ 4.6	94.4 $\pm$ 0.9	67.0 $\pm$ 5.5	51.94 $\pm$ 3.10	15.5 $\pm$ 1.0
QMIX Expert (Teacher)	$\sim$ 1 M	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0	48.4 $\pm$ 0.8	70.13 $\pm$ 0.16	0

Bold type identifies the proposed MiniUAV-VLA configuration and its corresponding results. All rows are five-seed mean $\pm$ std over seeds {2026, 3026, 4026, 5026, 6026} (500 episodes per condition in total). Greedy Heuristic navigates toward targets within the environment-defined view range only; random exploration is used when no target is visible. Vision-only ResNet-18 freezes the ResNet-18 backbone and only trains a $\approx$ 136 K-parameter MLP head; Vision-only CNN trains a ConvNeXt-Small backbone end-to-end ($\approx$ 51M trainable). Both vision-only baselines drop the structured prompt entirely. Text-only LLM shares the MiniUAV-VLA backbone and end-to-end fine-tuning protocol but removes the SigLIP2 vision encoder and the vision projection (no vision tokens at all). All learned methods are evaluated under identical environment reward parameters ($r_{\text{found}} = 0.5$, $r_{\text{elim}} = 5.0$, $r_{\text{suc}} = 50.0$, $r_{\text{col}} = -0.2$, $r_{\text{ovlp}} = -0.02$, $r_{\text{step}} = -0.01$), so the Return column is directly comparable. “Raw Illegal” is the fraction of *raw* arg-max predictions that would collide with obstacles or leave the map; such predictions are rerouted to the highest-logit legal action by the safe-actions post hoc correction (Section 3.7) before being sent to the environment, so every executed action is legal. Reported parameter counts are total inference parameters: MiniUAV-VLA’s 158 M comprises $\approx$ 65 M trainable backbone/projection plus the $\approx$ 93 M frozen SigLIP2 encoder; Text-only LLM’s 64 M is the MiniMind-3V backbone (no vision encoder); Vision-only ResNet-18’s 11 M is $\approx$ 11 M frozen ResNet-18 plus the $\approx$ 136 K trainable MLP head; Vision-only CNN’s 51 M is the trainable ConvNeXt-Small plus its projection head.

MiniUAV-VLA achieves a mission success rate of **74.4 $\pm$ 4.6%** with a found ratio of **94.4 $\pm$ 0.9%**, completing tasks in an average of **67.0 $\pm$ 5.5** steps across the five evaluation seeds. This is a **58.2 percentage point** gap relative to the observation-limited greedy baseline (16.2 $\pm$ 3.2%) and a **65.0 percentage point** gap relative to the random policy (9.4 $\pm$ 2.1%). These are descriptive gaps: each learned baseline represents one trained checkpoint, so they are not interpreted as training-level significance claims.

Modality contributions (vision-only vs. text-only vs. MiniUAV-VLA). The three single-modality ablations isolate the contribution of each input channel under identical training data and evaluation protocol. The Vision-only ResNet-18 (frozen backbone, $\approx$ 136 K trainable MLP head) remains weak at 14.4 $\pm$ 3.4%, below the greedy heuristic, showing that strong frozen ImageNet features without a trainable encoder are not enough to recover competent coordination in this setup. Scaling to the Vision-only CNN (ConvNeXt-Small, $\approx$ 51 M trainable, on the same order as the MiniUAV-VLA backbone) lifts success to 34.4 $\pm$ 5.6%, a **+20.0 pp jump over the frozen variant** that confirms vision benefits from trainable capacity, while a **40.0 pp gap to MiniUAV-VLA remains** despite a comparable trainable parameter budget. This remaining gap suggests that the structured prompt provides useful information complementary to the rendered map, including discovery counts, elimination

counts, and per-drone legal-action masks. The Text-only LLM (MiniMind-3V backbone shared with MiniUAV-VLA, no vision encoder) reaches $29.6 \pm 5.0\%$ despite seeing the full structured prompt, a **44.8 pp gap to MiniUAV-VLA** that gives a backbone-matched reference for the contribution of the visual channel. The text-only found ratio reaches only $63.4 \pm 0.8\%$. Notably, the two backbone-comparable single-modality variants (Vision-only CNN 34.4% , Text-only LLM 29.6%) are within 5 pp of each other, yet MiniUAV-VLA more than doubles either of them, consistent with the vision and structured-text channels providing complementary non-redundant information in this architecture.

The raw illegal action rate of $15.5 \pm 1.0\%$ indicates that the action head's raw arg-max prediction is illegal for roughly one out of every six drone-steps; although the safe-actions post hoc correction (Section 3.7) reroutes these to the highest-logit legal action before they reach the environment, ensuring that the executed action is always legal, this rate still serves as a diagnostic of how well the model has internalized the per-drone legal-action mask carried by the prompt. Section 4.6 quantifies the filter's contribution and separately reports the raw policy. The QMIX expert, with direct access to numeric observation vectors and recurrent state, achieves 100% success in 48.4 ± 0.8 steps with near-zero return variance across seeds. This provides a teacher reference for the distilled student rather than a strict upper bound, since the student operates from rendered imagery and structured text instead of the teacher's numeric observations.

4.3. Ablation Study A1: Action Pooling Position

To assess whether the pooling effect is stable across training randomness, we independently retrain both pooling variants five times as matched pairs. Within each pair, the two variants share the same training split, architecture, initialization, and hyperparameters; only the hidden-state extraction position differs. Each checkpoint is evaluated on the same five evaluation seeds with 100 episodes per seed (500 episodes per checkpoint). Seed 1 is the checkpoint reported in Table 1; we report the per-seed success and the mean across the five independently trained checkpoints, which separates training randomness from the rollout variation reported elsewhere.

Prompt-end pooling improves every one of the five independently trained pairs, by 32.0–40.4 points and increases the five-checkpoint mean success from 40.6% to 76.2% (mean paired gain 35.6 points; Table 2). The direction and magnitude of all five paired effects support the feature mismatch account, while the small number of training runs does not justify a broad claim of optimizer-independent superiority. Averaged over the five checkpoints, prompt-end pooling also raises the found ratio ($95.68 \pm 0.9\%$ versus $86.4 \pm 2.0\%$ for last-token), shortens episodes (65.9 ± 3.0 versus 104.9 ± 1.0 steps), and roughly halves the raw illegal-action rate ($15.8 \pm 1.5\%$ versus $26.3 \pm 17.4\%$), consistent with last-token pooling making weaker use of the prompt-carried legal-action mask.

Table 2. Training seed stability of action-head pooling (mission success, %). Seed 1 is the checkpoint reported in Table 1. Each checkpoint is evaluated over 500 episodes (five fixed evaluation seeds, 100 episodes each); the final column is mean $\pm$ sample standard deviation across the five checkpoints.

Pooling Strategy	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5	Mean $\pm$ SD
Last-Token (answer end)	41.8	39.2	42.8	38.2	41.2	40.6 ± 1.9
Prompt-End (Ours)	74.4	76.8	74.8	78.6	76.6	76.2 ± 1.7
Δ (Prompt-End – Last-Token)	+32.6	+37.6	+32.0	+40.4	+35.4	$+35.6 \pm 3.5$

Bold type identifies the proposed prompt-end pooling configuration and its results. All entries are mission success (percentage points). Seeds are distinct from the fixed evaluation seeds. Prompt-End improves success in every seed; an exact two-sided paired sign test gives $p = 0.0625$, and no significance star is assigned.

4.4. Ablation Study A2: Training Dataset Size

We vary the number of training transitions (1k, 5k, 10k) on the elimination task (Table 3). Subsets are drawn as the leading lines of the full training split; the same evaluation protocol is used across conditions.

Table 3. Ablation on training dataset size (eliminate task, 100 episodes per seed across five seeds; values are mean $\pm$ standard deviation across seeds).

Dataset Size	Success (%)	Found (%)	Steps	Raw Illegal (%)
1 k	42.0 $\pm$ 5.7	84.3 $\pm$ 2.1	96.5 $\pm$ 1.2	26.7 $\pm$ 1.4
5 k (Ours)	74.4 $\pm$ 4.6	94.4 $\pm$ 0.9	67.0 $\pm$ 5.5	15.5 $\pm$ 1.0
10 k	79.8 $\pm$ 4.1	96.4 $\pm$ 1.2	64.5 $\pm$ 4.2	15.3 $\pm$ 0.8

Bold type identifies the 5k baseline configuration used for the proposed model and its results. Seeds {2026, 3026, 4026, 5026, 6026}. All conditions use prompt-end pooling and identical hyperparameters. The 5k $\rightarrow$ 10k difference in success (5.4 pp) is comparable to the cross-seed standard deviations ($\sim$ 4–5 pp), indicating a modest but non-negligible improvement from doubling the data budget under this fixed 10-epoch protocol.

The jump from 1k transitions to 5k yields a **+32.4 pp** improvement in success rate (42.0 $\pm$ 5.7% $\rightarrow$ 74.4 $\pm$ 4.6%), in line with data volume being an important factor in low-data regimes. The further change from 5k to 10k is more modest (+5.4 pp, comparable to the $\sim$ 4–5 pp cross-seed standard deviation), indicating diminishing but non-negligible returns with further data scaling under this fixed 10-epoch protocol. The illegal action rate decreases sharply between 1k and 5k (26.7% $\rightarrow$ 15.5%) and is then nearly unchanged (15.5% vs. 15.3% at 10k, well within cross-seed variation). These results suggest that 5k expert transitions already provide a cost-effective training configuration under this protocol, with the marginal gain from doubling the budget to 10k roughly on the order of cross-seed variability.

4.5. Ablation Study A3: Auxiliary Text and Language-Modeling Objective

Table 4 separates the training-only auxiliary response from the action objective. The reference Full model is compared with the No-thinking and Action-only variants using one matched training seed; mean $\pm$ SD describes the five evaluation seed blocks, not training seed variation.

Table 4. Ablation of the training-only auxiliary response. Values are mean $\pm$ standard deviation across five evaluation seeds.

Variant	LM Loss	Thinking	Success (%)	Found (%)	Steps	Raw Illegal (%)
Full	Yes	Yes	74.4 $\pm$ 4.6	94.4 $\pm$ 0.9	67.0 $\pm$ 5.5	15.5 $\pm$ 1.0
No-thinking	Yes	No	59.8 $\pm$ 2.7	84.0 $\pm$ 0.7	75.5 $\pm$ 2.1	19.8 $\pm$ 0.6
Action-only	No	No	54.0 $\pm$ 2.9	82.9 $\pm$ 0.4	81.1 $\pm$ 3.3	20.6 $\pm$ 1.4

Paired success differences relative to Full: No-thinking, -14.6 points; Action-only, -20.4 points (one training seed, five evaluation seeds).

The Full variant achieves the highest success. Removing the thinking text (No-thinking) lowers success by 14.6 points to 59.8%, and additionally removing the language modeling objective (Action-only) lowers it further to 54.0%, for a 20.4-point drop overall. These results show that both auxiliary components contribute, with the larger single decrement coming from removing the semantic thinking text. Retaining both helps to provide auxiliary supervision, which encourages the backbone to retain a semantic representation of the multimodal state during training rather than collapsing into a pure action predictor. Because prompt-end pooling reads the action-relevant hidden state before any assistant tokens, this is a training-time effect: neither the thinking text nor the language modeling

objective is generated or consumed at inference, provides any memory across timesteps, or constitutes an interpretable reasoning trace.

4.6. Ablation Study A4: Legality Filter

Table 5 evaluates the learned raw policy and the complete system with post hoc legality filtering. Both rows use the same checkpoint and fixed evaluation seeds; their trajectories diverge when correction is first triggered.

Table 5. Ablation of the post hoc legality filter. Values are mean $\pm$ standard deviation across five evaluation seeds.

Evaluated System	Success (%)	Found (%)	Steps	Raw Illegal (%)
MiniUAV-VLA raw policy	72.6 $\pm$ 3.0	92.2 $\pm$ 0.7	70.3 $\pm$ 1.5	18.3 $\pm$ 1.7
MiniUAV-VLA + legality filter	74.4 $\pm$ 4.6	94.4 $\pm$ 0.9	67.0 $\pm$ 5.5	15.5 $\pm$ 1.0

Bold type identifies the results of the complete proposed system with legality filtering.

The filter raises mean success by 1.8 points (from 72.6% to 74.4%) and shortens episodes by 3.3 steps. Therefore, we describe the evaluated system as the learned policy combined with a deterministic legality filter. We do not claim that the network fully learns the legality constraints, only that it captures part of the legality structure; the external correction continues to perform measurable work.

4.7. Zero-Shot Environment and Changes to Target Behavior

In this section, we evaluate one factor at a time without retraining or condition-specific checkpoint selection. Table 6 changes the target count, map size, or obstacle probability while retaining the same four UAVs, mission definition, and 120-step horizon.

Table 6. Zero-shot evaluation under specified 2D environment changes. Values are mean $\pm$ standard deviation across five evaluation seeds.

Changed Factor	Setting	Success (%)	Found (%)	Steps	Raw Illegal (%)
Baseline	6 targets, 30 $\times$ 30, $p = 0.01$	74.4 $\pm$ 4.6	94.4 $\pm$ 0.9	67.0 $\pm$ 5.5	15.5 $\pm$ 1.0
Target count	4	80.8 $\pm$ 1.5	96.5 $\pm$ 0.4	60.2 $\pm$ 3.7	14.2 $\pm$ 1.8
Target count	8	69.0 $\pm$ 2.3	92.9 $\pm$ 0.5	79.2 $\pm$ 1.7	18.0 $\pm$ 0.9
Map size	20 $\times$ 20	95.6 $\pm$ 3.0	99.9 $\pm$ 0.1	44.0 $\pm$ 2.6	13.9 $\pm$ 0.7
Map size	40 $\times$ 40	67.0 $\pm$ 3.1	85.8 $\pm$ 1.5	87.8 $\pm$ 2.3	21.5 $\pm$ 1.0
Obstacle probability	0	79.4 $\pm$ 5.6	96.1 $\pm$ 0.4	59.4 $\pm$ 3.7	10.2 $\pm$ 0.8
Obstacle probability	0.03	68.6 $\pm$ 3.6	89.0 $\pm$ 0.2	74.5 $\pm$ 3.7	18.7 $\pm$ 0.5

Performance changes in the expected direction: larger maps, more targets, and denser obstacles are harder, while the 20 $\times$ 20 map is easiest. The same checkpoint retains 67.0–95.6% success across the tested map sizes and 68.6–79.4% across the tested obstacle probabilities. These are bounded robustness tests, not evidence of generalization to a different fleet size, mission, 3D dynamics, real sensing, or arbitrary language.

Table 7 separately replaces the original random-evasive target policy with the goal-directed evasive process defined in Section 3.

Table 7. Zero-shot evaluation with goal-directed evasive targets. Target goals are hidden from the VLA input. Values are mean $\pm$ standard deviation across five evaluation seeds.

Target Behavior	Defender Success (%)	Found (%)	Steps	Target Goal Completion (%)
Random-evasive (original)	74.4 $\pm$ 4.6	94.4 $\pm$ 0.9	67.0 $\pm$ 5.5	—
Goal-directed evasive	67.2 $\pm$ 4.3	89.0 $\pm$ 0.3	75.9 $\pm$ 3.7	25.3 $\pm$ 2.3

The more purposeful target behavior reduces defender success by 7.2 points, with 25.3% of target goals being completed. This is a harder mission-directed target test, but still does not model weapons, UAV attrition, or a full two-sided engagement.

4.8. Inference Efficiency and Memory Footprint

As shown in Table 8, mean end-to-end decision latency is 28.47 ms. Logical weights occupy 609.06 MiB in FP32 or 304.53 MiB in FP16/BF16; the saved VLA and external frozen vision weights occupy 313.97 MiB. Peak allocated GPU memory is 684.13 MiB, including a measured 42.94 MiB inference increment over the warm resident allocation. These measurements quantify compactness on a desktop GPU. Onboard deployment on a mini-UAV computer is not demonstrated.

Table 8. Batch-1 inference efficiency of MiniUAV-VLA on an NVIDIA GeForce RTX 4090.

Model	Total Params	Trainable	Precision	P50	P95	Decisions /s	Peak Memory
MiniUAV-VLA	~158 M	~65 M	FP16	28.28 ms	28.99 ms	35.13	684.13 MiB

4.9. Qualitative Analysis

Figure 3 shows representative episode trajectories. These trajectories are rendered from ground-truth evaluation logs for visualization. In successful episodes, MiniUAV-VLA exhibits coordinated spatial motion: drones distribute across map quadrants in early steps, sequentially enter the nominal view/attack ranges of targets in distinct regions, and converge on remaining targets in later steps. This behavior is not explicitly programmed but is learned from the QMIX expert demonstrations.

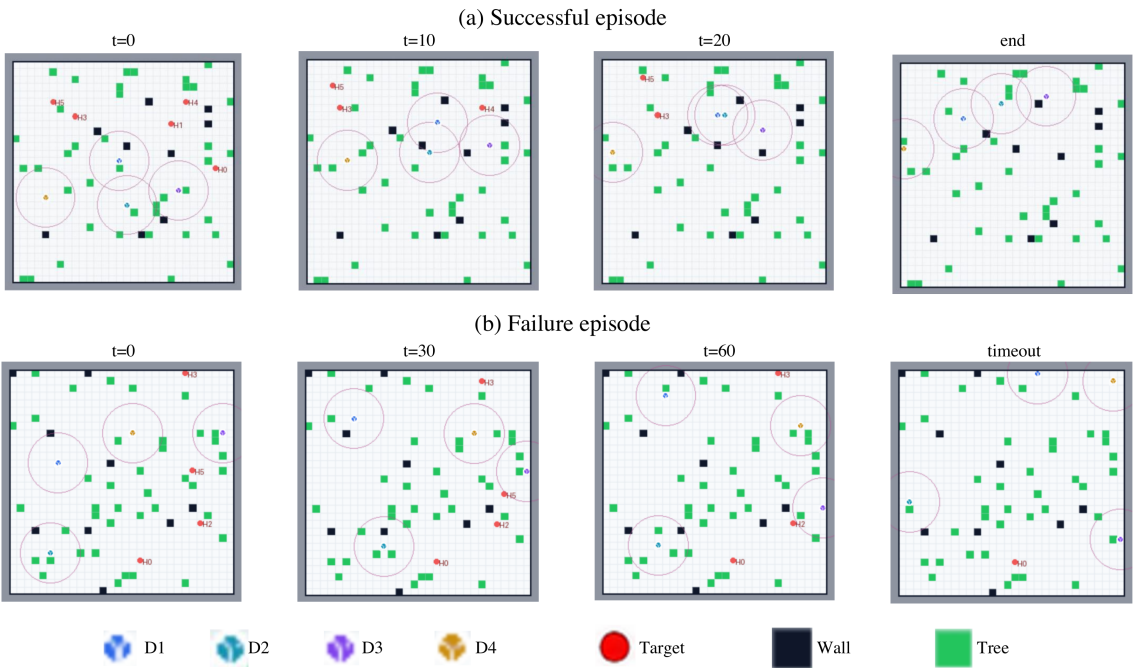


Figure 3. Representative closed-loop episode trajectories of MiniUAV-VLA on the eliminate task, rendered directly from evaluation rollout logs (i.e., real executed trajectories, not schematic reconstructions). Drone and target positions are read off the simulator state at the indicated timesteps. Overlapping transparent rings indicate simultaneous UAV view and attack coverage. They do not represent duplicated trajectories. (a) A successful episode: drones distribute across map quadrants in

the early phase, sequentially trigger the range-based found status in distinct regions, and converge on the remaining targets, eliminating all six within the episode horizon. **(b)** A failure episode (timed out at $t = T_{\max} = 120$): three drones cluster in the upper half of the map after triggering the found status for five targets, while the fleet fails to reach the final lower-right target's view/attack range and the episode terminates with five of six targets eliminated.

In failure cases, a representative failure mode observed in qualitative rollouts is a single target remaining in a peripheral map region after episode timeout. The model tends to concentrate drones in areas of high historical target density, and consequently can fail to bring a UAV within range of a target near a corner or obstacle cluster. This is consistent with the expected limitations of offline imitation learning, since the model has not been trained to recover from configurations where the current greedy strategy fails to bring a drone within the target's range thresholds.

Figures 4 and 5 provide four additional examples from archived rollouts of the MiniUAV-VLA 5k checkpoint. They are included for qualitative coverage only, and are not used to replace any quantitative result. Each row shows the first, middle, and final frame of one episode.

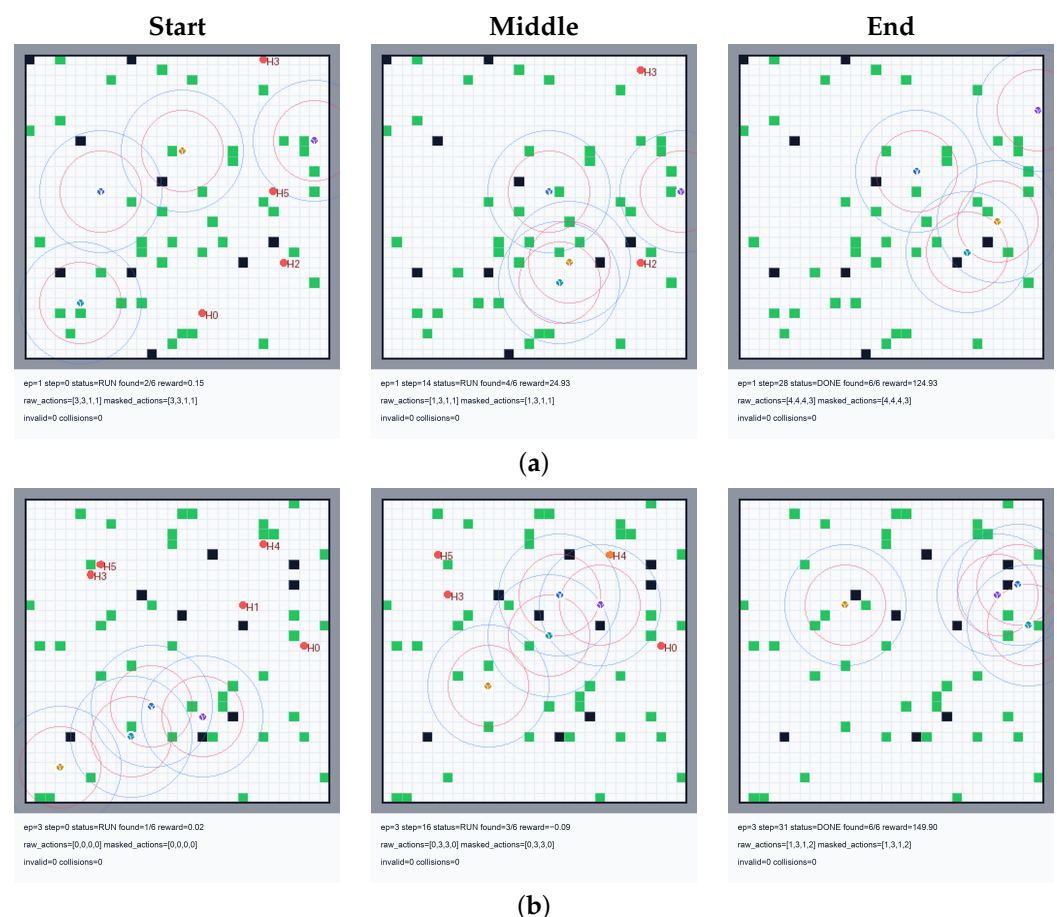


Figure 4. Two additional successful MiniUAV-VLA rollouts rendered from archived logs. Both episodes eliminate all six targets; their raw policies produce eight and seven illegal arg-max actions, respectively, before legality filtering. **(a)** Episode 1: success in 29 steps. **(b)** Episode 3: success in 32 steps.

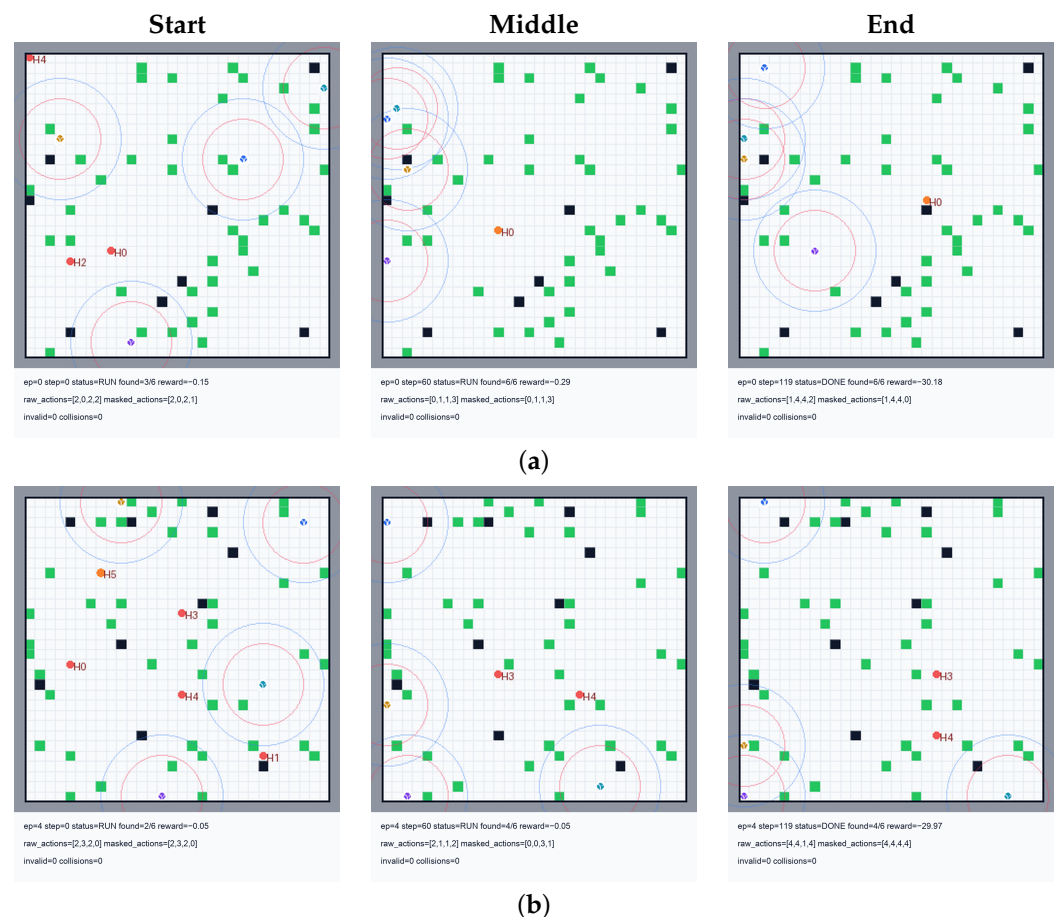


Figure 5. Two additional unsuccessful MiniUAV-VLA rollouts rendered from archived logs. Episode 0 finds all six targets, but fails to eliminate the last one; Episode 4 finds and eliminates only four. Both reach the 120-step limit, illustrating distinct late-stage pursuit and incomplete-coverage failures. (a) Episode 0: timeout with 5/6 targets eliminated. (b) Episode 4: timeout with 4/6 targets eliminated.

5. Discussion

5.1. Evidence for MARL-to-VLA Distillation in the Evaluated Setting

The $74.4 \pm 4.6\%$ closed-loop success of MiniUAV-VLA against $16.2 \pm 3.2\%$ for the observation-limited greedy heuristic and $9.4 \pm 2.1\%$ for the uniformly random policy supports MARL expert distillation as a viable pathway to multi-agent VLA training without real-world data collection in this simulated setting. We do not claim that MiniUAV-VLA dominates handcrafted coordination strategies. It has higher success than the random and observation-limited greedy references used in this study. These comparisons concern the fixed four-UAV, six-target baseline configuration, and should not be extrapolated to arbitrary fleet sizes or missions. Ablation A2 (Table 3) further shows a modest gap between the 5k and 10k conditions: 74.4% vs. 79.8%, a $\Delta = 5.4$ pp difference comparable to the ~ 4 – 5 pp cross-seed standard deviation. Even modest amounts of expert data (on the order of 5000 transitions, collectable in minutes on a single GPU) are sufficient for competent VLA behavior on this task configuration. The QMIX expert acts not only as a label provider but as an implicit coordinator; the VLA policy learns to distribute drones across the map, a cooperative pattern that a purely supervised approach on random trajectories would not exhibit.

Why does the distillation paradigm work this well? We attribute the observed data efficiency and absolute success rate to three concurrent effects. *First*, QMIX rollouts densely cover the coordinated-strategy subspace while ignoring obviously suboptimal joint actions,

so the student VLA never wastes capacity modeling unreachable or self-defeating regions of the action manifold. *Second*, the rendered tactical map compresses the high-dimensional spatial state into a fixed-resolution image, allowing the frozen SigLIP2 encoder to reuse its general visual priors instead of re-learning grid geometry from scratch under a small training budget. *Third*, the structured prompt explicitly carries discovery counts, elimination counts, and per-drone legal-action masks, offloading bookkeeping that would otherwise have to be inferred. Collectively these three effects compress the effective supervision complexity of the task, which matches the diminishing $5k \rightarrow 10k$ gain observed in Ablation A2 under the fixed 10-epoch training schedule.

The 58.2-point gap to the observation-limited greedy baseline and 65.0-point gap to the random policy show that the complete distilled system performs better than these references under the implemented benchmark.

5.2. Modality Contributions: Vision and Structured Text Are Complementary

The three single-modality ablations in Table 1 attribute the headline success rate to its constituent input channels. They comprise one low-capacity vision baseline (Vision-only ResNet-18, frozen), one backbone-comparable vision baseline (Vision-only CNN, trainable ConvNeXt-Small), and one backbone-matched text baseline (Text-only LLM).

Trainable capacity alone is not enough on the vision side. Scaling the vision baseline from a frozen ResNet-18 with a 136k-parameter head ($14.4 \pm 3.4\%$) to a fully trainable ≈ 51 M-parameter ConvNeXt-Small ($34.4 \pm 5.6\%$) yields a **+20.0 pp** improvement. While the trainable capacity is useful, but a 40.0 pp gap to MiniUAV-VLA remains despite a trainable parameter budget on the same order as the MiniUAV-VLA backbone. This remaining gap suggests that the structured prompt provides useful information that is complementary to the rendered map.

The structured text alone is not sufficient. The Text-only LLM baseline retains the full structured prompt, including discovery counts, elimination counts, and per-drone legal-action masks, on the same MiniMind-3V backbone as MiniUAV-VLA, with the SigLIP2 vision encoder and projection removed. If those structured signals were enough to drive coordinated control, this baseline would already be competitive; instead, its found ratio reaches only $63.4 \pm 0.8\%$, 31 pp below MiniUAV-VLA.

The rendered image alone is not sufficient. Even the trainable Vision-only CNN sees the same map at every step; however, it lacks the symbolic constraints carried by the prompt (found/eliminated progress, per-drone legal-action masks, mission mode). Its raw illegal-action rate (32.2%) is more than double MiniUAV-VLA's (15.5%), and its success rate falls 40 pp short. The structured prompt provides an explicit state channel that the dense visual stream does not replicate in this setup. However, it is not evaluated as instruction-following in free-form natural language.

Taken together, the two channels appear complementary in this architecture. This is consistent with the prompt constraining *which* action is admissible and the image informing *where* to move. Removing either channel reduces success by at least 40 points relative to the complete system. The experiment supports structured-text conditioning, but does not demonstrate paraphrase robustness, unseen instruction following, or adaptation to new mission descriptions.

5.3. Prompt-End Pooling and the Behavior Cloning Alignment Problem

Across five independent training runs, prompt-end pooling raises mean success from $40.6 \pm 1.9\%$ to $76.2 \pm 1.7\%$, with improvements of 32.0–40.4 points in every pair (Table 2). The magnitude is comparable to the +32.4-point gain from a fivefold increase in dataset size ($1k \rightarrow 5k$ in Ablation A2, Table 3), indicating that the pooling position is a first-order

design factor rather than a minor tuning detail. This pattern is consistent with the training–inference feature mismatch being an architectural issue in VLA behavior cloning rather than just a data or optimization problem.

Why is the effect so large? The mismatch is a right-context shortcut, related to the same training–inference discrepancies that have long been discussed in sequence modeling [40]. During training, the last-token variant sees the ground-truth action text inside its receptive field, so the action head finds it easier to attend to the just-decoded response tokens than to the more distant prompt context. At inference, those response tokens are absent, and the head is forced to predict from a hidden-state distribution it has never seen during training. This failure mode is also related to shortcut learning [45,46], where models exploit predictive but non-causal cues that are available during training but unreliable at deployment.

The aggregate raw-illegal rates across the five runs ($26.3 \pm 17.4\%$ for last-token versus $15.8 \pm 1.5\%$ for prompt-end) are consistent with this account: the head makes less use of the explicit legal-action mask carried by the prompt, because at training time it could lean on the response tokens instead. Prompt-end pooling removes the shortcut by extracting the hidden state at a position which has identical right-context at both training and inference; thus, the head must ground its decisions in the prompt-derived visual and structured features. The training seed experiment shows that the improvement is not peculiar to one initialization. Therefore, we identify prompt-end pooling (or any equivalent training inference-aligned pooling position) as a useful design consideration for VLA architectures trained via offline behavior cloning.

5.4. Auxiliary Supervision, Safety, Robustness, and Compute

The auxiliary-objective ablation shows a monotonic pattern: the Full variant achieves the highest success (74.4%); removing the thinking text (No-thinking) lowers success by 14.6 points, and additionally removing the language modeling objective (Action-only) lowers it by 20.4 points overall. Both the thinking text and the language modeling loss contribute; therefore, we retain both as auxiliary supervision. This encourages the backbone to keep a semantic representation of the state during training rather than collapsing into a pure action predictor. Because prompt-end pooling reads before the response, this is a training-time effect, and we do not describe the thinking field as online reasoning or explanation.

The legality filter has a measurable but modest effect, with the complete system reaching 74.4% success versus 72.6% for the raw policy. The learned policy captures part but not all of the legality structure; therefore, the filtered results are explicitly attributed to the combination of neural policy with constraint enforcement.

The zero-shot tests show graceful but non-invariant behavior under the selected 2D changes. Success falls on the larger, denser, or more target-rich settings, and the goal-directed targets reduce defender success from 74.4% to 67.2%. These results are evidence of bounded robustness to the factors that were actually varied, not open-world generalization; in particular, all tests retained four UAVs and the same discrete mission/action interface.

Finally, 28.47 ms mean end-to-end latency and 684.13 MiB peak allocation on an RTX 4090 establish a reproducible desktop GPU reference. The measurements are compact relative to billion-parameter VLA systems, but an RTX 4090 is still not equivalent to an onboard mini-UAV processor. Quantization, embedded-GPU benchmarking, power draw, thermal limits, and communication latency still need to be measured before any onboard deployment claim can be made.

5.5. Limitations and Future Directions

This study deliberately isolates the MARL-to-VLA distillation paradigm in a controlled simulated setting, which defines several scope boundaries that we make explicit and that point to clear next steps. First, the controller is centralized and operates on a shared rendered tactical map together with a structured prompt. Decentralized onboard execution and sim-to-real transfer lie outside the scope of this study. The reported RTX 4090 latency and memory are reference measurements, not evidence of an onboard deployment footprint; accordingly, “compact” is defined relative to billion-parameter VLA architectures, not to flight qualification on a mini-UAV computer. A direct next step is benchmarking on embedded hardware and a move towards egocentric onboard sensing. Second, the action head predicts a fixed joint action for $N = 4$ drones, and both the structured prompt template and the $(N \cdot |A|)$ -dimensional output layer assume static fleet membership. We address the joint-action size for the specific $N = 4$ setting via a fixed-size factorized head; promising extensions include larger fleets, mid-mission attrition from communication loss, mechanical failure or planned drop-out, and permutation-invariant or autoregressive action representations. No experiments in this paper changed the number of UAVs, so generalization to larger or smaller fleets remains untested. Third, the distillation dataset retains only transitions from *successful* expert episodes (Section 3.3), which introduces a survivorship bias: the student VLA never sees the recovery behavior that the teacher would produce when an episode is on the verge of timing out; consequently, the student has no in-distribution prior over near-failure states. This bias is a plausible contributor to the step-count gap between MiniUAV-VLA and the QMIX teacher (67.0 vs. 48.4 steps), and is a concrete target for future ablation. The above gap also confirms that the student is not behaviorally equivalent to the teacher; instead, the offline distillation recovers useful coordinated behavior. Fourth, the framework relies on offline SFT alone; online fine-tuning with environment feedback (e.g., REINFORCE or PPO on the VLA policy) is a direct route to further closing the remaining ~ 26 pp gap to the QMIX expert. Finally, the Drones environment is a 2D grid abstraction with no altitude, continuous velocity, acceleration, attitude, aerodynamics, sensor noise, communication loss, energy constraint, or physical collision dynamics. The goal-directed target experiment improves behavioral realism within this abstraction, but does not remove these limitations. We did not perform a real-UAV or high-fidelity simulator test, so the current evidence supports high-level simulated coordination only. Extending the framework to 3D continuous state spaces and physical UAV dynamics is the key step toward real-world deployment. A concrete pathway runs through photorealistic physics-based simulation (e.g., AirSim [47], Flightmare [48], or Isaac Gym [49]) bridged by domain randomization and sim-to-real transfer [50,51]. This would represent a move from a centralized top-down map to onboard egocentric partial perception with decentralized execution, along with staged hardware-in-the-loop and small-scale flight tests; notably, inputs such as the per-drone positions and legal-action masks correspond to signals that are already available onboard (GPS/INS estimates and local obstacle sensing, respectively).

6. Conclusions

We present **MiniUAV-VLA**, a compact centralized VLA controller for simulated cooperative multi-UAV search and elimination with 158 M total inference parameters and ≈ 65 M trainable parameters in the MiniMind-3V backbone and vision projection. This paper delivers the four contributions stated point-for-point in Section 1:

1. *Contribution 1: MARL-to-VLA distillation pipeline.* A QMIX expert achieving 100% mission success generated 5000 closed-loop multimodal transitions, which were used to fine-tune the VLA backbone without any human-collected demonstrations. Ablation A2 (Table 3) shows that 5000 transitions already delivers performance close

to the 10k condition under our fixed 10-epoch training schedule: $74.4 \pm 4.6\%$ vs. $79.8 \pm 4.1\%$ mean $\pm$ std over five seeds. This $\Delta = 5.4$ pp gap is comparable to the cross-seed standard deviation, supporting distillation from a strong MARL teacher as a data-efficient route to multi-agent VLA training in this setting.

2. *Contribution 2: Multi-agent action head with prompt-end pooling.* A single $768 \rightarrow 20$ linear head, reshaped into four per-drone five-action blocks, predicts all four drones' actions jointly. Prompt-end pooling addresses training–inference feature mismatch in the standard last-token configuration. Ablation A1 (Table 2) isolates this design choice, showing a 35.6-point mean improvement across five independent training runs (last-token $40.6 \pm 1.9\%$ versus prompt-end $76.2 \pm 1.7\%$), with the same direction in every pair.
3. *Contribution 3: Structured multimodal state representation.* A rendered tactical map paired with a compact structured prompt encodes spatial layout, discovery and elimination progress, and per-drone legal-action masks. This representation allows the action head to combine spatial reasoning over the image with explicit symbolic constraints from the prompt while remaining within the 512-token budget of the MiniMind-3V backbone, and supports the ≈ 65 M trainable backbone in recovering competitive cooperative behavior. The single-modality baselines (Table 1) indicate that both channels contribute: a trainable vision-only ConvNeXt-Small reaches only $34.4 \pm 5.6\%$ without the structured prompt, the same MiniMind-3V backbone reaches only $29.6 \pm 5.0\%$ without the rendered image, and a frozen ResNet-18 head bottoms out at $14.4 \pm 3.4\%$, each at least 40 pp below MiniUAV-VLA under identical training data conditions, consistent with the two modalities providing complementary non-redundant information in this architecture.
4. *Contribution 4: Empirical validation.* MiniUAV-VLA achieves $74.4 \pm 4.6\%$ success in the baseline configuration, while the raw policy without the legality filter reaches $72.6 \pm 3.0\%$. Additional experiments tested auxiliary objectives, specified zero-shot environment changes, goal-directed target behavior, and inference cost, with success remaining 67.0–95.6% across the tested map sizes and the goal-directed condition reaching $67.2 \pm 4.3\%$. Batch-1 latency was 28.47 ms on an RTX 4090. These results apply only to the evaluated 2D setting with four UAVs.

Taken together, these results are consistent with MARL and VLA being complementary rather than competing. MARL acquires cooperative strategies through environmental interaction, while VLA provides a multimodal interface conditioned on structured text for redeploying those strategies in a compact policy. The scope boundaries of this study and the corresponding future directions are discussed in Section 5.5.

Author Contributions: Conceptualization, H.H.; methodology, H.H.; software, H.H.; validation, H.H.; formal analysis, H.H.; investigation, H.H.; resources, N.L.; data curation, H.H.; writing—original draft preparation, H.H.; writing—review and editing, H.H., G.G. and N.L.; visualization, H.H.; supervision, G.G. and N.L.; project administration, G.G. and N.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The source code, configurations, trained checkpoints, and evaluation scripts are publicly available at <https://github.com/super1603/MiniUAV-VLA> (accessed on 26 July 2026).

Acknowledgments: The authors thank the developers of MiniMind-V [34] and EPyMARL [52] for their open-source implementations.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

VLA	Vision–Language–Action
MARL	Multi-Agent Reinforcement Learning
UAV	Unmanned Aerial Vehicle
QMIX	Q-value Mixing Network
Dec-POMDP	Decentralized Partially Observable Markov Decision Process
CTDE	Centralized Training with Decentralized Execution
SFT	Supervised Fine-Tuning
SigLIP	Sigmoid Loss for Language Image Pretraining
CLIP	Contrastive Language–Image Pretraining
RoPE	Rotary Position Embedding
GRU	Gated Recurrent Unit
MLP	Multilayer Perceptron
GELU	Gaussian Error Linear Unit
LM	Language Model

Appendix A. Environment and Training Hyperparameters

Tables A1 and A2 provide the complete environment and training hyperparameters for reproducibility.

Table A1. Drones environment configuration.

Parameter	Value
Map size	30×30
Number of drones	4
Number of targets	6
View range r_v	6
Attack range r_a	4
Episode limit	120 steps
Target stay probability	0.2
Target random-move probability (outside view range)	0.8
Target flee-from-nearest-drone probability (within view range)	0.6
Goal-directed target probabilities (pre-discovery)	goal 0.8/random 0.1/stay 0.1
Goal-directed target probabilities (post-discovery)	goal 0.5/evade 0.3/random 0.1/stay 0.1
Obstacle probability	0.01
Reward: new detection	0.5
Reward: elimination	5.0
Reward: success	50.0
Reward: collision penalty	−0.2
Reward: view overlap penalty	−0.02
Reward: step penalty	−0.01

Table A2. VLA training hyperparameters.

Hyperparameter	Value
Backbone	MiniMind-3V (8 layers, $d = 768$)
Vision encoder	SigLIP2-base-p16-224 (≈ 93 M, frozen)
Image input/visual tokens	$224 \times 224/196$ patch tokens
Action head	Linear(768, 20) $\rightarrow$ reshape(4, 5)
Trainable backbone/projection parameters	≈ 65 M
Total inference parameters	≈ 158 M
Epochs	10
Batch size	8
Learning rate	10^{-5} (cosine schedule)
Action loss weight λ	1.0
Gradient clip norm	1.0
Optimizer	AdamW
Vision encoder weights	Frozen

Appendix B. Structured Prompt Format

The full prompt template used at each timestep is:

```
Mode={mission_mode};vr={view_range};ar={attack_range};
map={map_size};t={step}/{max_step};
A=0U1D2L3R4S;N={num_drones};
D={r0,c0|r1,c1|r2,c2|r3,c3};
F={found}/{total};E={eliminated}/{total};
M={mask0|mask1|mask2|mask3}; return {num_drones} actions.
```

where M is a 5-bit binary mask per drone (1 = legal action).

References

1. Rashid, T.; Samvelyan, M.; Schroeder de Witt, C.; Farquhar, G.; Foerster, J.; Whiteson, S. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018*; Dy, J., Krause, A., Eds.; Proceedings of Machine Learning Research; MLR Press: Cambridge, MA, USA, 2018; Volume 80, pp. 4295–4304.
2. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-Decomposition Networks for Cooperative Multi-Agent Learning Based on Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems; AAMAS'18*; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2018; pp. 2085–2087.
3. Yu, C.; Velu, A.; Vinitisky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 September–9 December 2022*; Curran Associates, Inc.: Red Hook, NY, USA, 2022; Volume 35, pp. 24611–24624.
4. Zitkovich, B.; Yu, T.; Xu, S.; Xu, P.; Xiao, T.; Xia, F.; Wu, J.; Wohlhart, P.; Welker, S.; Wahid, A.; et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Proceedings of the 7th Conference on Robot Learning, Atlanta, GA, USA, 6–9 November 2023*; Tan, J., Toussaint, M., Darvish, K., Eds.; Proceedings of Machine Learning Research; MLR Press: Cambridge, MA, USA, 2023; Volume 229, pp. 2165–2183.
5. Kim, M.J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; et al. OpenVLA: An Open-Source Vision-Language-Action Model. In *Proceedings of the 8th Conference on Robot Learning, Munich, Germany, 6–9 November 2025*; Agrawal, P., Kroemer, O., Burgard, W., Eds.; Proceedings of Machine Learning Research; MLR Press: Cambridge, MA, USA, 2025; Volume 270, pp. 2679–2713.
6. Black, K.; Brown, N.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; Groom, L.; Hausman, K.; Ichter, B.; et al. π_0 : A Vision-Language-Action Flow Model for General Robot Control. *arXiv* **2024**, arXiv:2410.24164. [[CrossRef](#)]
7. Schroeder de Witt, C.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.S.; Sun, M.; Whiteson, S. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? *arXiv* **2020**, arXiv:2011.09533. [[CrossRef](#)]
8. Samvelyan, M.; Rashid, T.; Schroeder de Witt, C.; Farquhar, G.; Nardelli, N.; Rudner, T.G.J.; Hung, C.M.; Torr, P.H.S.; Foerster, J.; Whiteson, S. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, Montreal, QC, Canada, 13–17 May 2019*; AAMAS'19; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2019; pp. 2186–2188.
9. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017*; Volume 30, pp. 6379–6390.
10. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018*; Volume 32, pp. 2974–2982.
11. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.E.; Yi, Y. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019*; Volume 97, pp. 5887–5896.
12. Rashid, T.; Farquhar, G.; Peng, B.; Whiteson, S. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Virtual, 6–12 December 2020*; Volume 33.
13. Yehoshua, R.; Heredia-Juesas, J.; Wu, Y.; Amato, C.; Martinez-Lorenzo, J. Decentralized Reinforcement Learning for Multi-Target Search and Detection by a Team of Drones. *arXiv* **2021**, arXiv:2103.09520. [[CrossRef](#)]
14. Liu, Y.; Li, X.; Wang, J.; Wei, F.; Yang, J. Reinforcement-Learning-Based Multi-UAV Cooperative Search for Moving Targets in 3D Scenarios. *Drones* **2024**, *8*, 378. [[CrossRef](#)]

15. Wei, D.; Zhang, L.; Liu, Q.; Chen, H.; Huang, J. UAV Swarm Cooperative Dynamic Target Search: A MAPPO-Based Discrete Optimal Control Method. *Drones* **2024**, *8*, 214. [\[CrossRef\]](#)
16. Chen, Y.; Dong, Q.; Shang, X.; Wu, Z.; Wang, J. Multi-UAV Autonomous Path Planning in Reconnaissance Missions Considering Incomplete Information: A Reinforcement Learning Method. *Drones* **2023**, *7*, 10. [\[CrossRef\]](#)
17. Zhao, X.; Yang, R.; Zhong, L.; Hou, Z. Multi-UAV Path Planning and Following Based on Multi-Agent Reinforcement Learning. *Drones* **2024**, *8*, 18. [\[CrossRef\]](#)
18. Kouzeghar, M.; Song, Y.; Meghjani, M.; Bouffanais, R. Multi-Target Pursuit by a Decentralized Heterogeneous UAV Swarm using Deep Multi-Agent Reinforcement Learning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 3289–3295.
19. Zhao, Z.; Wan, Y.; Chen, Y. Deep Reinforcement Learning-Driven Collaborative Rounding-Up for Multiple Unmanned Aerial Vehicles in Obstacle Environments. *Drones* **2024**, *8*, 464. [\[CrossRef\]](#)
20. Frattolillo, F.; Brunori, D.; Iocchi, L. Scalable and Cooperative Deep Reinforcement Learning Approaches for Multi-UAV Systems: A Systematic Review. *Drones* **2023**, *7*, 236. [\[CrossRef\]](#)
21. Team, O.M.; Ghosh, D.; Walke, H.; Pertsch, K.; Black, K.; Mees, O.; Dasari, S.; Hejna, J.; Kreiman, T.; Xu, C.; et al. Octo: An Open-Source Generalist Robot Policy. *arXiv* **2024**, arXiv:2405.12213. [\[CrossRef\]](#)
22. Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Dabis, J.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Hsu, J.; et al. RT-1: Robotics Transformer for Real-World Control at Scale. *arXiv* **2023**, arXiv:2212.06817. [\[CrossRef\]](#)
23. Driess, D.; Xia, F.; Sajjadi, M.S.M.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; et al. PaLM-E: An Embodied Multimodal Language Model. In Proceedings of the International Conference on Machine Learning (ICML), Honolulu, HI, USA, 23–29 July 2023; Volume 202.
24. Jiang, Y.; Gupta, A.; Zhang, Z.; Wang, G.; Dou, Y.; Chen, Y.; Fei-Fei, L.; Anandkumar, A.; Zhu, Y.; Fan, L. VIMA: General Robot Manipulation with Multimodal Prompts. In Proceedings of the International Conference on Machine Learning (ICML), Honolulu, HI, USA, 23–29 July 2023; Volume 202.
25. O'Neill, A.; Rehman, A.; Maddukuri, A.; Gupta, A.; Padalkar, A.; Lee, A.; Pooley, A.; Gupta, A.; Mandlekar, A.; Jain, A.; et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 13–17 May 2024; pp. 6892–6903. [\[CrossRef\]](#)
26. Physical Intelligence; Black, K.; Brown, N.; Darpinian, J.; Dhaliwal, K.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; et al. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv* **2025**, arXiv:2504.16054. [\[CrossRef\]](#)
27. Shukor, M.; Aubakirova, D.; Capuano, F.; Kooijmans, P.; Palma, S.; Zouitine, A.; Aractingi, M.; Pascal, C.; Russi, M.; Marafioti, A.; et al. SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics. *arXiv* **2025**, arXiv:2506.01844. [\[CrossRef\]](#)
28. Kalidas, A.P.; Joshua, C.J.; Md, A.Q.; Basheer, S.; Mohan, S.; Sakri, S. Deep Reinforcement Learning for Vision-Based Navigation of UAVs in Avoiding Stationary and Mobile Obstacles. *Drones* **2023**, *7*, 245. [\[CrossRef\]](#)
29. Yang, Z.; Zhang, Y.; Zeng, J.; Yang, Y.; Jia, Y.; Song, H.; Lv, T.; Sun, Q.; An, J. AI-Driven Safety and Security for UAVs: From Machine Learning to Large Language Models. *Drones* **2025**, *9*, 392. [\[CrossRef\]](#)
30. Zhang, P.; Zeng, G.; Wang, T.; Lu, W. TinyLlama: An Open-Source Small Language Model. *arXiv* **2024**, arXiv:2401.02385. [\[CrossRef\]](#)
31. Abdin, M.; Aneja, J.; Awadalla, H.; Awadallah, A.; Awan, A.A.; Bach, N.; Bahree, A.; Bakhtiari, A.; Bao, J.; Behl, H.; et al. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *arXiv* **2024**, arXiv:2404.14219. [\[CrossRef\]](#)
32. Team, G.; Riviere, M.; Pathak, S.; Sessa, P.G.; Hardin, C.; Bhupatiraju, S.; Hussenot, L.; Mesnard, T.; Shahriari, B.; Ramé, A.; et al. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv* **2024**, arXiv:2408.00118. [\[CrossRef\]](#)
33. Qwen, Y.; Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; et al. Qwen2.5 Technical Report. *arXiv* **2024**, arXiv:2412.15115. [\[CrossRef\]](#)
34. Gong, J. MiniMind-V: Train a 65M-Parameter VLM from Scratch. 2024. Available online: <https://github.com/jingyaogong/minimind-v> (accessed on 29 May 2026).
35. Zhao, T.Z.; Kumar, V.; Levine, S.; Finn, C. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In Proceedings of the Robotics: Science and Systems (RSS), Daegu, Republic of Korea, 10–14 July 2023. [\[CrossRef\]](#)
36. Chi, C.; Feng, S.; Du, Y.; Xu, Z.; Cousineau, E.; Burchfiel, B.C.; Song, S. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In Proceedings of the Robotics: Science and Systems (RSS), Daegu, Republic of Korea, 10–14 July 2023. [\[CrossRef\]](#)
37. Pomerleau, D.A. ALVINN: An Autonomous Land Vehicle in a Neural Network. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 27–30 November 1988; Volume 1, pp. 305–313.
38. Ross, S.; Gordon, G.; Bagnell, D. 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, Fort Lauderdale, FL, USA, 11–13 April 2011; Gordon, G., Dunson, D., Dudík, M., Eds.; Proceedings of Machine Learning Research; MLR Press: Cambridge, MA, USA, 2011; Volume 15, pp. 627–635.

39. Rusu, A.A.; Colmenarejo, S.G.; Gulcehre, C.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; Hadsell, R. Policy Distillation. *arXiv* **2015**, arXiv:1511.06295. [[CrossRef](#)]
40. Bengio, S.; Vinyals, O.; Jaitly, N.; Shazeer, N. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28, pp. 1171–1179.
41. Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Zhang, C.; Wang, J.; Wang, Z.; Yau, S.K.S.; Lin, Z.; et al. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
42. Chen, W.; Su, Y.; Zuo, J.; Yang, C.; Yuan, C.; Chan, C.M.; Yu, H.; Lu, Y.; Hung, Y.H.; Qian, C.; et al. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. *arXiv* **2023**, arXiv:2308.10848. [[CrossRef](#)]
43. Oliehoek, F.A.; Amato, C. *A Concise Introduction to Decentralized POMDPs*; SpringerBriefs in Intelligent Systems; Springer: Cham, Switzerland, 2016; pp. 1–134. [[CrossRef](#)]
44. Zhai, X.; Mustafa, B.; Kolesnikov, A.; Beyer, L. Sigmoid Loss for Language Image Pre-Training. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2–6 October 2023; pp. 11975–11986.
45. Geirhos, R.; Jacobsen, J.H.; Michaelis, C.; Zemel, R.; Brendel, W.; Bethge, M.; Wichmann, F.A. Shortcut learning in deep neural networks. *Nat. Mach. Intell.* **2020**, *2*, 665–673. [[CrossRef](#)]
46. McCoy, R.T.; Pavlick, E.; Linzen, T. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy, 28 July–2 August 2019; pp. 3428–3448.
47. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Proceedings of the Field and Service Robotics: Results of the 11th International Conference*; Springer Proceedings in Advanced Robotics; Springer: Berlin/Heidelberg, Germany, 2017; Volume 5, pp. 621–635.
48. Song, Y.; Naji, S.; Kaufmann, E.; Loquercio, A.; Scaramuzza, D. Flightmare: A Flexible Quadrotor Simulator. In *Proceedings of the Conference on Robot Learning*; PMLR; MLR Press: Cambridge, MA, USA, 2021; pp. 1147–1157.
49. Makoviychuk, V.; Wawrzyniak, L.; Guo, Y.; Lu, M.; Storey, K.; Macklin, M.; Hoeller, D.; Rudin, N.; Allshire, A.; Handa, A.; et al. Isaac Gym: High Performance GPU-Based Physics Simulation for Robot Learning. *arXiv* **2021**, arXiv:2108.10470.
50. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; IEEE: Piscataway, NJ, USA, 2017; pp. 23–30. [[CrossRef](#)]
51. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In *Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI)*; IEEE: Piscataway, NJ, USA, 2020; pp. 737–744. [[CrossRef](#)]
52. Papoudakis, G.; Christianos, F.; Schäfer, L.; Albrecht, S.V. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In Proceedings of the Advances in Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS), Virtual, 6–14 December 2021.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.