CoLA: Multi-VLA Coordination via Learned Adapters

Anonymous authors
Paper under double-blind review

Abstract

Vision-Language-Action (VLA) models are typically deployed as isolated single-agent policies. When several such agents share a workspace, they can coordinate implicitly, by observing one another, or explicitly, by exchanging messages, but it remains unclear when explicit communication becomes necessary. We introduce **CoLA** (**Co**ordination via **L**earned **A**dapters), which keeps each agent's VLA backbone frozen and trains a lightweight message channel and a per-arm diffusion action head on top, with fully decentralised execution. Across three tasks that differ in how much each agent can observe, the channel matters only when information is private. On an occluded tray handover, where the box's colour marker is visible to one agent only, CoLA reaches 76.0% success versus 1.3% with messages zeroed throughout training. On two-arm and three-arm handover tasks, removing messages has no measurable effect with a shared overhead camera (90.6% vs. 88.3% at two arms; 63.5% vs. 60.0% at three), but lowers success on the two-arm task's wrist-only variant (74.8% vs. 64.3%). This architecture generalises across different VLA backbones as well. With a frozen $\pi_{0.5}$ backbone on the same occluded tray task, CoLA matches the overall success rate of a LoRA-fine-tuned baseline (37.3% vs. 34.7%) using a fraction of the trainable parameters. When CoLA places the box, it selects the correct tray 71.8% of the time, whereas the independently fine-tuned baseline, whose placing arm never sees the marker, selects it at chance level (34.7% against 33.3%). [1]

1 Introduction

Robot foundation models trained on large cross-embodiment datasets now generalise well in single-arm manipulation (Brohan et al., 2023b;a; Team et al., 2024; Kim et al., 2024). However, many real-world robotic deployments inherently require several arms or robots to act together, ranging from dual-arm collaborative handovers to multi-robot assembly and warehouse staging (Krnjaic et al., 2024; Yifan et al., 2025). In such domains, task execution cannot rely exclusively on individual dexterity; success relies on mutual perception, intention signalling, and closed-loop behavioural synchrony between autonomous partners.

Despite their dexterity in isolation, current VLA models do not coordinate naturally when co-located in a shared workspace (Dey et al., 2026). Each operates under an egocentric assumption: it perceives the workspace through its own sensors and treats partner arms as unmodelled obstacles. Closing this gap faces three practical constraints. Training multi-arm policies end-to-end (Liu et al., 2025b; Zhao et al., 2023; Fu et al., 2024) requires scarce multi-robot data and discards single-agent pretraining. LLM coordinators (Mandi et al., 2024; Liu et al., 2025a) run at 0.5–1 Hz, which is too slow for reactive manipulation. And deployed robots often run fixed on-device backbones whose weights are costly or impossible to modify (we defer further discussion of related works to Section 2).

These constraints jointly define the setting we study, including agents whose backbones stay frozen, coordinating without a centralised arbiter, at real-time control rates. In this setting, agents can coordinate only implicitly, by observing one another, or through an explicit channel added alongside their frozen policies. We therefore ask: *when does explicit inter-agent communication improve coordination over implicit observation alone, and how does this depend on task coupling and partial observability?*

[1] Code: `https://anonymous.4open.science/r/CoLA-F372`

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

To answer this question, we propose **Co**ordination via **L**earned **A**dapters (**CoLA**). CoLA discards each backbone's native action head entirely and in its place trains three modular components on top of each frozen VLA backbone: a latent message encoder, a partner message decoder, and a coordination-conditioned action chunking head. The encoder compresses the agent's readout embedding into a low-dimensional message vector. Each agent sends its encoded message to the other agents via the communication channel. Received messages are then decoded and fused with the agent's self representation, before being mapped to a coordinated action chunk. The encoders, decoders, and action heads are optimised jointly by behavioural cloning on multi-robot demonstrations.

We evaluate CoLA in four settings of increasing shared observability, built on three tasks: an occluded tray handover, where the colour marker is visible only to arm A and arm B must place the box in the matching tray; a two-arm handover with each arm's own wrist camera only; the same two-arm handover with a shared overhead camera; and a three-arm handover chain, which also shares the overhead camera. All settings use a frozen Octo-Base backbone, and the tray task is additionally tested with a frozen $\pi_{0.5}$ backbone. Against the strongest decentralised baseline in each setting, head-only or fully fine-tuned, CoLA achieves between 5.5 and 53.3 percentage points higher success rate, with the largest gain on the occluded tray handover (76.0% vs. 22.7%). To isolate the channel itself, we retrain CoLA with every message fixed to the zero vector during and after training. Removing the channel collapses success on the occluded tray task (76.0% to 1.3%) and lowers it under wrist-only observation (74.8% to 64.3%), but has no measurable effect once the arms share an overhead view (90.6% vs. 88.3% at two arms and 63.5% vs. 60.0% at three, both within seed-to-seed variation). Whether a team of frozen VLAs needs an explicit channel is therefore determined by the task's information structure: the channel is essential when task-relevant state is private to one agent, and redundant when the agents can already observe it. Because the channel is a small module trained on cached features of a frozen backbone, it can be added to already-deployed VLA policies without retraining them, and used only where the information structure calls for it. This makes the choice of when and what to communicate a design question in its own right for multi-robot VLA systems.

2 Related Work

Multi-Agent Reinforcement Learning and Emergent Communication. Learned inter-agent communication is well-studied in cooperative MARL (Albrecht et al., 2024). Architectures like CommNet (Sukhbaatar et al., 2016), DIAL (Foerster et al., 2016), and TarMAC (Das et al., 2019) optimise continuous differentiable communication channels jointly with agent policies. Emergent communication shows that symbolic or continuous communication protocols can spontaneously ground coordination cues without direct linguistic supervision (Lazaridou et al., 2017; Havrylov & Titov, 2017; Baker et al., 2020). Alternatively, communication can also be strictly bounded to the action space. Recent approaches, such as Multiagent Control via Action Suggestions (MCAS), demonstrate that sharing suggested joint actions, instead of full observations, allows agents to effectively infer belief states under decentralised partial observability (Asmar & Kochenderfer, 2024). However, these methods train all agent policies from scratch. In CoLA, by contrast, each agent's feature extractor is a pretrained VLA that stays frozen while the channel is learned.

Multi-Robot and Bimanual Manipulation. Prior multi-robot manipulation architectures typically rely on unified centralised controllers. Centralised imitation models such as ACT (Zhao et al., 2023), Mobile ALOHA (Fu et al., 2024), InterACT (Lee et al., 2024), and RDT-1B (Liu et al., 2025b) model multiple limbs within a single concatenated action space. TwinVLA (Im et al., 2026) coordinates twin single-arm models via dense cross-attention, while AnyBimanual (Lu et al., 2025) transfers unimanual policies to bimanual control; both, however, update backbone weights. Closer to our setting, CHORUS (Doshi et al., 2026) adapts a single VLA backbone so that each robot runs an independent copy conditioned only on its own observations and a robot-identifying prompt, and MIMIC-D (Dong et al., 2025) jointly trains decentralised diffusion policies that coordinate implicitly from local information. Both deliberately avoid communication between agents at inference time; neither asks when such a channel becomes necessary, which is the question we study. LLM-based planners such as RoCo (Mandi et al., 2024) and COHERENT (Liu et al., 2025a) employ multi-agent text dialogues for task assignment, but their multi-second query latencies cannot sustain reactive physical handovers.

Parameter-Efficient Fine-Tuning (PEFT). PEFT methods such as adapters (Houlsby et al., 2019), LoRA (Hu et al., 2022), and prompt tuning (Lester et al., 2021) adapt a single foundation model to a new domain, and in robotics are used to adapt VLAs to new embodiments or sensors (Kim et al., 2024; 2025). CoLA instead repurposes adapters for communication between agents.

3 Problem Formulation & Methodology

3.1 Decentralised Multi-Agent Formalisation

We model multi-robot cooperative manipulation as a decentralised partially observable process with communication, following the Dec-POMDP-Com (Goldman & Zilberstein, 2003) but without a reward function, as is standard in imitation learning (Abbeel & Ng, 2004). The model is defined by the tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}_i\}, \Sigma, \mathcal{P}, \{\mathcal{O}_i\}, \Omega, \rho_0, T \rangle$. Here $\mathcal{I} = \{1, \dots, N\}$ is the set of N robot agents; $\mathcal{S}$ is the global state space; $\mathcal{A}_i$ is the continuous action space of agent i; and $\Sigma = \mathbb{R}^{d_m}$ is the space of messages agents may exchange. The transition distribution $\mathcal{P}(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}_1, \dots, \boldsymbol{a}_N)$ depends on physical actions only, so messages influence the environment solely through the actions they induce. Each agent i receives observations from its local observation space $\mathcal{O}_i$, drawn jointly according to $\Omega(\boldsymbol{o}_1, \dots, \boldsymbol{o}_N \mid \boldsymbol{s})$. Unlike the standard formulation, which conditions observations on the successor state and joint action, we condition on the current state only, since each observation is a camera rendering of that state. Finally, ρ_0 is the initial state distribution and T is the episode horizon.

Under decentralised execution, each agent receives only its local observation $\boldsymbol{o}_i^t \in \mathcal{O}_i$ (an RGB camera stream), without direct access to partner observations $\boldsymbol{o}_{j \neq i}^t$ or the global state $\boldsymbol{s}^t$ (a subset of experiments relaxes this with a shared overhead view; those results are marked explicitly). Agents communicate over a channel graph defined by neighbourhood sets $\mathcal{N}(i) \subseteq \mathcal{I} \setminus \{i\}$: at each timestep t, agent i broadcasts a message $\boldsymbol{m}_i^t \in \Sigma$ and receives $\{\boldsymbol{m}_j^t\}_{j \in \mathcal{N}(i)}$. We restrict attention to *reactive* policies, which condition on the current observation and the partners' current messages rather than on full action–observation histories:

$$\pi_i\big(\boldsymbol{A}_i^t \mid \boldsymbol{o}_i^t, \{\boldsymbol{m}_j^t\}_{j \in \mathcal{N}(i)}\big), \tag{1}$$

where $\boldsymbol{A}_i^t = \big(\boldsymbol{a}_i^t, \boldsymbol{a}_i^{t+1}, \dots, \boldsymbol{a}_i^{t+K-1}\big) \in \mathbb{R}^{K \times D_a}$ denotes an open-loop chunk of K future actions ($D_a = 7$). This restriction suffices when task-relevant state is either currently observable to some agent or recoverable from current proprioception.

Rather than optimising a reward, we are given demonstrations from a scripted expert team policy $\pi^E : \mathcal{S} \to \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$, which maps the global state to a joint action $(\boldsymbol{a}_1^t, \dots, \boldsymbol{a}_N^t)$ for every agent at each timestep, so each demonstration trajectory records all N agents' actions synchronously. The expert acts on the global state $\boldsymbol{s}^t$, so it is not itself a decentralised reactive policy; a learner can reproduce its behaviour only to the extent that the required information is available through the agent's own observation or the messages it receives. Our goal is to learn decentralised reactive policies that imitate π^E, which we do by behavioural cloning (Section 3.5).

3.2 Frozen Foundation Backbones

Each agent i has a pretrained VLA backbone f_θ whose parameters remain frozen throughout training. At each timestep, the backbone encodes each of the agent's V_i camera views $v \in \{1, \dots, V_i\}$, together with the task instruction $\boldsymbol{l}$, into a feature vector, $\boldsymbol{f}_i^{t,v} = f_\theta(\boldsymbol{o}_i^{t,v}, \boldsymbol{l}) \in \mathbb{R}^d$. The agent's proprioceptive state $\boldsymbol{q}_i^t$ is mapped to an embedding $\boldsymbol{p}_i^t = \phi_{\text{prop}}^i(\boldsymbol{q}_i^t)$ by a small network trained jointly with the other modules. Because proprioception is easier to fit than the visual and message channels, a policy can shortcut through it and under-use those harder signals (Lu et al., 2026); to discourage this, $\boldsymbol{q}_i^t$ is zeroed with probability 0.1 during training. The agent's representation concatenates these:

$$\boldsymbol{h}_i^t = \big[\, \boldsymbol{f}_i^{t,v_1} \,\|\, \dots \,\|\, \boldsymbol{f}_i^{t,v_{V_i}} \,\|\, \boldsymbol{p}_i^t \,\big] \in \mathbb{R}^{d_h}. \tag{2}$$

Both the CoLA message encoder and the action head take $\boldsymbol{h}_i^t$ as input, so a message can carry information from any of the agent's views and from its physical state. Because f_θ is frozen, its features can be computed once and cached, and the trainable modules can only select and transmit information that f_θ already encodes.

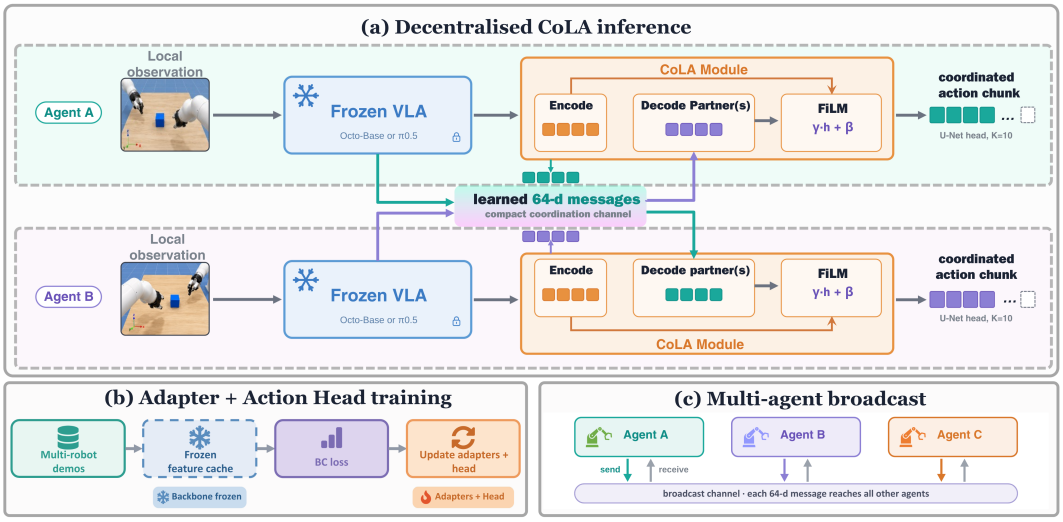

Figure 1: **CoLA framework overview.** (a) *Decentralised inference:* each agent's frozen VLA backbone (Octo-Base or $\pi_{0.5}$) encodes its local observation. Its CoLA module compresses the backbone feature into a compact message ($d_m = 64$), decodes the partner's message, and concatenates the result with its own representation to form $\boldsymbol{z}_i$. This vector conditions a U-Net action head via FiLM, which predicts a coordinated action chunk ($K = 10$). (b) *Adapter and action-head training:* backbone features are cached offline and the backbones stay frozen. Only the CoLA modules are trained with a behavioural cloning loss, with gradients flowing end-to-end through the communication channel. (c) *Multi-agent broadcast:* each agent broadcasts its message to all other agents and receives theirs, so the same design scales to $N \geq 3$ agents.

3.3 CoLA Architectural Design

Each agent i discards its backbone's native action head and instead hosts three trainable modules (Fig. 1):

Latent Message Encoder (ϕ^i_{enc}). A two-layer MLP with ReLU activations that compresses local backbone representations into a coordination message:

$$\boldsymbol{m}_i^t = \phi^i_{\text{enc}}\big(\text{sg}_{\boldsymbol{f}}[\boldsymbol{h}_i^t]\big) \in \mathbb{R}^{d_m}, \quad d_m = 64, \tag{3}$$

where $\text{sg}_{\boldsymbol{f}}[\cdot]$ denotes a stop-gradient applied only to the backbone-feature components of $\boldsymbol{h}_i^t$; the proprioceptive encoder ϕ^i_{prop} receives gradients through the message.

Message Decoder ($\phi^{j\to i}_{\text{dec}}$). One decoder per *ordered pair*, mapping partner j's message into i's feature space,

$$\boldsymbol{c}_{j\to i}^t = \phi^{j\to i}_{\text{dec}}(\boldsymbol{m}_j^t) \in \mathbb{R}^d, \quad \forall j \in \mathcal{N}(i). \tag{4}$$

Decoders are not shared across senders: i learns a distinct mapping for each partner, so that in the three-arm task B reads A's and C's messages through separate weights.

Coordination Action Head. Concatenates the agent's own representation with its decoded partner embeddings,

$$\boldsymbol{z}_i^t = \big[\, \boldsymbol{h}_i^t \,\big\|\, \{\boldsymbol{c}_{j\to i}^t\}_{j\in\mathcal{N}(i)} \,\big] \in \mathbb{R}^{d_h + |\mathcal{N}(i)|\,d}, \tag{5}$$

which conditions the diffusion head of Section 3.5: a 1D temporal U-Net with FiLM conditioning that predicts the noise added to the joint chunk, alongside a logit head for the binary gripper column.

Action Chunking Dynamics. Actions are generated in chunks of $K = 10$ steps executed open-loop at 50 Hz control rate (0.2 s per chunk). The policy is queried once per chunk, giving a 5 Hz inference rate (one query every 200 ms).

3.4 Multi-Agent Topology Routing

CoLA extends to N agents through flexible graph topologies:

Bimanual Teams ($N = 2$): Direct peer-to-peer exchange, $\mathcal{N}(1) = \{2\}$ and $\mathcal{N}(2) = \{1\}$. Each agent's head input is $[\boldsymbol{h}_i \,\|\, \boldsymbol{c}_{j\to i}]$, of dimension $d_h + d$.

Tri-Manual System ($N = 3$): A fully connected topology, $\mathcal{N}(i) = \mathcal{I} \setminus \{i\}$ for every agent, giving six directed channels. Each agent decodes *both* partners' messages through its own per-sender decoder, $\boldsymbol{z}_i = [\boldsymbol{h}_i \,\|\, \boldsymbol{c}_{j\to i} \,\|\, \boldsymbol{c}_{k\to i}]$ of dimension $d_h + 2d$. The architecture is symmetric: no agent acts as a hub, all three heads take identically shaped input, and encoder and decoder architectures are identical across agents.

3.5 End-to-End Imitation Learning Objective

Given a multi-agent demonstration dataset $\mathcal{D} = \{(\boldsymbol{o}^t_{1:N}, \boldsymbol{A}^t_{1:N})\}$ collected from scripted experts, CoLA is trained by behavioural cloning. We split each expert chunk as $\boldsymbol{A}^t_i = [\boldsymbol{Q}^t_i, \tilde{\boldsymbol{g}}^t_i]$, where $\boldsymbol{Q}^t_i \in \mathbb{R}^{K\times 6}$ holds the joint targets and $\tilde{\boldsymbol{g}}^t_i \in \{-0.9, 0.9\}^K$ the normalised gripper channel, from which we derive binary labels $\boldsymbol{y}^t_i = \mathbb{1}[\tilde{\boldsymbol{g}}^t_i > 0] \in \{0,1\}^K$ for the classification loss. Following diffusion policies (Ho et al., 2020; Chi et al., 2025), we sample, independently for each agent, a diffusion step τ_i uniformly from $\{1, \dots, S-1\}$ with $S = 100$, and noise $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We then form the corrupted joint chunk $\tilde{\boldsymbol{Q}}^t_i = \sqrt{\bar{\alpha}_{\tau_i}}\,\boldsymbol{Q}^t_i + \sqrt{1-\bar{\alpha}_{\tau_i}}\,\boldsymbol{\epsilon}_i$ under a cosine schedule (Nichol & Dhariwal, 2021), and train each agent's head ϕ^i_{unet} to predict the added noise conditioned on $\boldsymbol{z}^t_i$:

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{\mathcal{D}}\Big[\sum_{i=1}^{N} \mathbb{E}_{\tau_i,\,\boldsymbol{\epsilon}_i}\big[\mathcal{L}_i\big]\Big], \qquad \mathcal{L}_i = \frac{1}{6K}\big\|\phi^i_{\text{unet}}(\tilde{\boldsymbol{Q}}^t_i, \tau_i, \boldsymbol{z}^t_i) - \boldsymbol{\epsilon}_i\big\|^2_F + \lambda\,\mathrm{BCE}\big(\phi^i_{\text{grip}}(\boldsymbol{z}^t_i),\, \boldsymbol{y}^t_i\big), \tag{6}$$

where ϕ^i_{grip} outputs K gripper logits, the BCE is averaged over the K steps, and $\lambda = 0.25$. At test time, we sample with DDIM (Song et al., 2020) using 10 denoising steps. We predict the gripper with a separate classification head because it is binary: in early development, predicting it as a continuous dimension produced near-constant outputs, which motivated the separate head. The classification head reaches 98.7–100% validation gripper accuracy across all four coordination regimes (Table 14).

Gradients Through the Channel. Because $\boldsymbol{z}^t_i$ depends on partners' messages, gradients flow across the channel. Agent j's message $\boldsymbol{m}^t_j$ enters only its receivers' action heads, so j's encoder is trained entirely by its partners' losses, whereas the decoder producing $\boldsymbol{c}^t_{j\to i}$ belongs to agent i and is trained by $\mathcal{L}_i$ alone. Each encoder is therefore pushed to transmit whatever its partners need to disambiguate their actions.

Offline Feature Caching. Because f_θ is frozen, its features are computed once per dataset and reused by every run and ablation, so training CoLA takes under one GPU-hour, against 2–49 for the fine-tuned baselines at the same optimiser budget (Table 5).

4 Experiments

Our experiments aim to answer the following questions: **(Q1)** Can agents built on frozen VLA backbones coordinate under decentralised execution when only a message channel and a new action head are trained? **(Q2)** How much does the message channel contribute to success? **(Q3)** Does the approach generalise to three agents and to different frozen backbones?

4.1 Tasks

We evaluate on three tasks that differ in how much task-relevant information each agent can observe with its own cameras (Fig. 2).

Occluded tray handover. Arm A grasps a box bearing one of three coloured markers (blue, green, or yellow) on the face pointing toward Arm A, and hands it to Arm B, which must place it on the

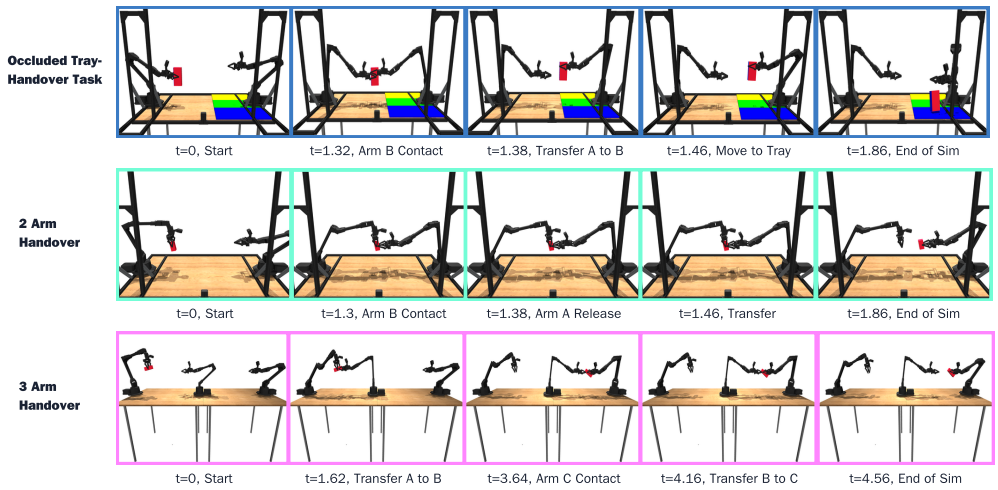

Figure 2: **Multi-Agent Manipulation Progression.** Key milestones across evaluated tasks: (top) Occluded Tray-Handover Task, (middle) 2-arm Reactive Handover, and (bottom) 3-arm Handover Chain.

tray of the matching colour on its own side of the table. Each arm observes only its own wrist camera. The marker appears in Arm A's camera in 100% of control steps and in Arm B's in under 5%. Without communication, Arm B therefore has almost no access to the colour, and among the episodes in which it places the box it can expect to select the correct tray only about one time in three. An episode is successful if the box enters the correct tray.

Two-arm handover. Arm A starts holding the object and presents it at a handoff point; Arm B must approach, grasp, and hold while A releases. A's presentation pose is randomised by ± 10 cm laterally and vertically. The challenge is timing: B must close its gripper before A opens its own, and a premature release drops the object. We evaluate two variants: each arm's wrist camera only, and each arm's wrist camera together with a shared overhead camera. An episode is successful if Arm B grasps the object without dropping it.

Three-arm handover chain. The object passes A $\rightarrow$ B $\rightarrow$ C, with A's presentation randomised as above, and all arms additionally share the overhead view alongside their own wrist camera. B must receive the object and pass it on, C's approach depends on when B is ready, and a failure at either transfer ends the episode. An episode is successful if both transfers complete and Arm C then holds the object for 20 steps.

4.2 Experimental Setup

Simulation Environments. All environments are simulated in MuJoCo (Todorov et al., 2012) with a 2 ms physics timestep (500 Hz) and a control decimation of 10, giving a 50 Hz control rate. We control $N \in \{2, 3\}$ 6-DoF Trossen ViperX 300 arms with parallel-jaw grippers. Each arm's action is 7-dimensional: target positions for its six revolute joints, normalised per dimension to $[-0.9, 0.9]$ using min–max statistics from the training split, plus a gripper command, which CoLA predicts as a binary open/closed decision. Each agent observes a 256×256 RGB image from its own wrist-mounted camera, together with its 7-dimensional proprioceptive state (six joint angles and gripper opening). In the overhead variants of the two- and three-arm handover, every agent additionally observes a shared fixed overhead camera alongside its own wrist camera (Section 3.1), for CoLA and the baselines alike. The occluded tray-handover task uses wrist cameras only.

All three tasks use Octo-Base (93M parameters). On the occluded tray-handover task we additionally evaluate $\pi_{0.5}$ (base checkpoint, $\sim$3.3B parameters (Physical Intelligence et al., 2025)). Both backbones are frozen in their entirety and run once offline, encoding each agent's observation at each timestep into a single cached feature vector: the 768-dimensional readout embedding for Octo-

Base and, because $\pi_{0.5}$ has no readout token, a padding-masked mean over its final-layer PaliGemma prefix tokens (image, prompt, and joint state as 256-bin tokens, matching the LoRA baseline) giving a 2048-dimensional vector. On this backbone, the 7-d proprioceptive state therefore enters CoLA twice: once inside the frozen 2048-D feature (via the prompt tokens) and again through ϕ_{prop} (Section 3.2); the proprio dropout described there regularises only the latter copy. Neither backbone's own action head is used. CoLA's modules (Section 3.3) have the same architecture across backbones; only the layer widths that depend on the backbone feature dimension change (14.5M trainable parameters with Octo-Base, 16.9M with $\pi_{0.5}$, of which the communication channel accounts for 0.33M and 0.83M respectively and the remainder is the per-arm action head and proprioceptive encoder). CoLA uses a single generic instruction, "coordinate with partner", for every arm and task, whereas the baselines receive role-specific instructions (Table 4). No instruction names the marker colour. Pretrained Octo does not complete the two-arm handover zero-shot with its own action head (Table 13).

Evaluation Protocol. For the handover tasks, each policy is evaluated on 200-episode blocks at evaluation-seed offsets drawn from 0, 1000, 2000, 3000 and 4000, and we report the mean $\pm$ standard deviation of success rate across the blocks evaluated. The number of blocks differs by cell: five for every method on the 2-arm overhead task except CoLA with zeroed messages (three); three for every method on the 2-arm wrist-only task (offsets 0, 1000, 2000); and five for the baselines but a single block (offset 0) for both CoLA rows on the three-arm chain. The per-seed blocks behind every cell are given in Tables 6 to 8. The occluded tray-handover task keeps the handover positions fixed and randomises only hidden marker colour among three discrete values. A single seed with 150 episodes already samples each colour densely ($\sim$50 episodes per colour), therefore this task is evaluated with only a single seed (0). For this task we report success rate with a 95% Wilson confidence interval over episodes, with the per-stage breakdown in Table 10.

Baselines for comparison. All baselines are trained on the same demonstrations and episode splits as CoLA and, unless stated otherwise, on the same Octo-Base backbone. Within each task, every method uses the same number of optimiser steps at the same effective batch size of 128 (12,600, 20,250 and 26,850 steps for the two-arm handover, tray and three-arm handover tasks, respectively). This matches the optimisation budget, not compute or trainable capacity. Learning rates and optimisers follow each method's standard recipe (Adam without weight decay for CoLA; AdamW with weight decay 10^{-4} for the Octo baselines), and full architectural and optimisation settings are listed in Table 3 (Section B). Since CoLA trains on cached backbone features, matching optimiser steps gives the fine-tuned baselines substantially more compute than CoLA.

- **Independent decentralised policies:** one policy per arm, conditioned on its wrist camera on the wrist-only tasks, or on both its wrist camera and the shared overhead camera on the overhead tasks, trained either (a) *head-only*, with the backbone frozen and a freshly initialised action head, or (b) with *full fine-tuning* of all backbone weights on the multi-robot demonstrations. All Octo baselines receive proprioception through Octo's proprio tokenizer: each decentralised policy sees its own arm's 7-D joint state, and the centralised policy both arms' (14-D). Because Octo-Base was pretrained without proprioception, this projection is newly initialised; head-only baselines keep it frozen at that initialisation, while full fine-tuning trains it.
- **Centralised policy:** a single model that outputs the joint 14-D action, in the same two regimes, from both arms' wrist cameras on the wrist-only tasks, or from both arms' wrist cameras plus the shared overhead camera on the overhead tasks. It violates decentralised execution and serves as a privileged reference.
- **CoLA with zeroed messages:** the full CoLA architecture, trained and evaluated with all messages set to zero. It has the same architecture, parameter count, and training recipe as CoLA; the only difference is that the channel carries no information, which isolates the contribution of communication.
- **Cross-backbone transfer (occluded tray-handover task):** independent per-arm $\pi_{0.5}$ policies fine-tuned with openpi's LoRA recipe (LoRA on both Gemma experts plus the full vision encoder; 467.0M trainable parameters per arm), compared against CoLA on the frozen $\pi_{0.5}$ backbone (8.5M per arm), testing whether CoLA depends on a particular backbone.

4.3 Main Empirical Results

Table 1: **Main results.** Success rate (%), ordered from least to most shared observability. HO = handover. Handover columns: mean ± std over 200-episode evaluation blocks (Section 4.2); every row is a single training run. Tray: single 150-episode block. –: not evaluated.

Method	Occluded Tray-HO (wrist only)	2-Arm HO (wrist only)	2-Arm HO (overhead)	3-Arm HO (overhead)
CoLA (with message)	**76.0**	74.8 ± 2.6	90.6 ± 1.7	**63.5**[†]
CoLA (message zeroed)	1.3	64.3 ± 6.7	88.3 ± 2.9	60.0[†]
Decentralised, head-only	22.7	43.8 ± 2.0	68.3 ± 2.6	20.5 ± 2.7
Centralised, head-only	16.0	21.0 ± 2.2	65.9 ± 4.7	—
Centralised, full fine-tune	10.7	$\mathbf{80.0 \pm 2.6}$	$\mathbf{92.1 \pm 2.0}$	—
Decentralised, full fine-tune	12.0	69.3 ± 1.8	82.4 ± 3.1	31.6 ± 4.5

[†] single 200-episode block. Tray chance is 33.3% conditional on reaching a tray. Per-seed and per-stage results: Section D.

Q1: Coordination Efficacy. Table 1 shows that CoLA outperforms every decentralised baseline on all four tasks, while modifying only lightweight adapters and an action head atop a frozen backbone. On the occluded tray-handover task, independent decentralised policies perform poorly, while CoLA reaches 76.0%. On the handover tasks, CoLA exceeds the decentralised head-only baseline by 31.0 percentage points on wrist-only handover (74.8% vs. 43.8%), 22.3 percentage points with a shared overhead view (90.6% vs. 68.3%), and 43.0 points at three arms (63.5% vs. 20.5%). CoLA also approaches the privileged centralised full fine-tune on handover (90.6% vs. 92.1% with a shared view; 74.8% vs. 80.0% wrist-only) while retaining fully decentralised execution. Fig. 3 contrasts a CoLA rollout with an uncoordinated baseline, and Section F categorises the failures.

Q2: Is Communication Causally Necessary? The head-only baseline underperforms no-message CoLA on every handover task (e.g., 68.3% vs. 88.3% with a shared overhead view). Additionally, CoLA replaces Octo's action head with a diffusion U-Net with FiLM conditioning, predicts the gripper with a separate classification head, adds a learned proprioceptive embedding, and uses a different chunk length and training schedule. Comparisons against the Octo baselines therefore measure CoLA's whole pipeline, not communication. The no-message ablation isolates the channel: it keeps CoLA's architecture and training recipe unchanged and removes only the message content. Removing messages collapses the occluded tray handover (76.0% to 1.3%), although most of this drop occurs before tray selection: the message-zeroed model completes the handover itself in only 8.0% of episodes, against 88.7% for the head-only baseline, which also has no channel (Table 10). Training with a constant zero message therefore impairs the whole policy on this task, so 1.3% is not just the cost of withholding the colour alone. The evidence that the channel carries the colour is tray choice given a placement (85.7% for CoLA against 26.4% for the head-only baseline, chance 33.3%) and the probes of Section 4.4. Removing messages also degrades the wrist-only handover (74.8% to 64.3%), but has no measurable effect when a shared overhead view shows each agent its partner (90.6% vs. 88.3% at two arms; 63.5% vs. 60.0% at three, within noise). The channel therefore matters to the extent that task-relevant information is private to one agent. When agents share a view, CoLA's advantage over the independent baselines comes from the pipeline differences above, which we do not isolate individually. Table 11 shows that performance is largely insensitive to message width $d_m \in \{16, 64, 128\}$.

Q3: Does CoLA's Advantage Persist with Team Size and Backbone? At three arms, CoLA outperforms decentralised full fine-tuning by 31.9 percentage points (63.5% vs. 31.6%), up from 8.2 percentage points at two arms (90.6% vs. 82.4%), and the head-only baseline by 43.0 points (up from 22.3). Because zeroed-message CoLA reaches 60%, this widening margin reflects CoLA's architecture rather than communication when an overhead view is shared; whether the channel's own contribution grows with team size requires a three-arm task with private information. Per-stage transfer rates for the chain are reported in Table 9.

Table 2: **Cross-backbone transfer on the occluded tray-handover task.** CoLA on a frozen $\pi_{0.5}$ backbone versus independently LoRA-fine-tuned per-arm $\pi_{0.5}$ policies. *Placed*: the box reaches any tray. *Correct | placed*: the box reaches the correct tray, given that it reaches one (chance 33.3%). *Overall*: the box reaches the correct tray, over all episodes. Brackets give 95% Wilson intervals; **bold** marks a value whose interval does not overlap the other method's.

Method	Params (per arm)	Train (GPU-h)	Placed (%)	Correct | placed (%)	Overall (%)
CoLA	8.5M	0.2	52.0 [44.1, 59.8]	**71.8** [61.0, 80.6]	37.3 [30.0, 45.3]
Decentralised, LoRA fine-tune	467.0M	24.4	**100.0** [97.5, 100]	34.7 [27.5, 42.6]	34.7 [27.5, 42.6]

To test whether the channel's benefit is specific to Octo, we repeat the occluded tray-handover task with a frozen $\pi_{0.5}$ backbone (Table 2). The two methods fail in different places. The LoRA-fine-tuned per-arm policies place the box on a tray in every episode but choose the correct tray at chance level (34.7% against a chance rate of 33.3%), consistent with the placing arm having no access to the hidden colour. CoLA places the box in only 52.0% of episodes, but when it does, it selects the correct tray 71.8% of the time (95% CI $[61.0, 80.6]$), well above both chance and the baseline. The message channel therefore still carries the hidden task information on $\pi_{0.5}$, what does not transfer is manipulation reliability, and overall success is statistically indistinguishable (37.3% vs. 34.7%).

4.4 Probing What the Channel Encodes

We analyse the information transmitted in the message by fitting linear probes (multinomial logistic regression for colour, ridge regression for pose) from three representations of the sending agent: its own features $\boldsymbol{h}_i^t$ (*self*), its message $\boldsymbol{m}_i^t$ from the trained checkpoint (*message*), and the encoder output from the message-zeroed checkpoint (*control*), which receives no gradient when messages are zeroed (Section 3.5) and so stays at random initialisation. On the occluded tray-handover task, the message decodes occluded box colour at 73.3% versus chance-level 33.3% for the control (episode-level majority vote; per frame, 44.2% vs. 39.7%), a 40-point gap attributable to training through the channel that corroborates the necessity results (Fig. 4 and Table 12, Section D).

5 Conclusion and Limitations

Conclusion. We presented CoLA, a framework that trains a message-passing channel and a new action-generation pipeline on top of frozen, independently deployed VLA backbones, and used it to measure when that channel is actually necessary for coordination. Isolating the channel's effect from the rest of the architecture, we found communication is not uniformly beneficial. It is necessary when task-relevant state is private to one agent, and has no measurable effect once agents share a view of the workspace. On the occluded tray-handover task, the channel also carried the hidden colour with a frozen $\pi_{0.5}$ backbone, although overall success there did not exceed that of the LoRA-fine-tuned baseline. Our results suggest that explicit communication and implicit, shared-context coordination are not competing solutions but complementary ones, each suited to a different information regime, and that determining which regime a task falls into is itself a necessary step in designing multi-agent VLA systems.

Limitations. All experiments run in simulation on scripted expert demonstrations, and we have not tested transfer to physical robots. CoLA also assumes zero-latency synchronous communication. Although realistic for local high-speed bus architectures, performance under asynchronous network dropouts and variable latency remains to be benchmarked. Additionally, CoLA connects agents with a fully connected topology and a separate decoder for every ordered pair (Section 3.4), so both the number of channels and the number of decoders grow as $O(N^2)$, with six of each at $N = 3$. This is practical at the team sizes we study, but scaling to dozens of robots will require sharing decoder weights across senders, or aggregating messages through graph neural network or cross-attention routers (Das et al., 2019).

Reproducibility Statement

Environments use MuJoCo 3.4.0 with the Trossen ViperX 300s models from MuJoCo Menagerie (physics 500 Hz, control 50 Hz). All demonstrations are successful expert episodes, split 80/10/10 by episode. Frozen Octo-Base features (and Octo-Small, used only in Table 13) are pre-extracted in JAX and cached. Hyperparameters are listed in Table 3 and the full procedure in Algorithm 1. Experiments ran on one node with 8 NVIDIA A100 (40 GB) GPUs. Code: `https://anonymous.4open.science/r/CoLA-F372`.

AI Use Statement

Generative AI tools were used exclusively for rephrasing and condensing author written text, formatting assistance, spelling and grammar refinement, and LaTeX template conversion according to ICLR guidelines. All conceptual formulations, experimental analyses, and technical claims were designed and verified by the authors.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024.

Dylan M. Asmar and Mykel J. Kochenderfer. Efficient multiagent planning via shared action suggestions. *ArXiv*, 2024. URL `https://api.semanticscholar.org/CorpusID:274776469`.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

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

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2025.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Sharmita Dey, Robert Riener, Strahinja Dosen, and Stefano V Albrecht Albrecht. From autonomy to alliance: Robotic foundation models must learn with us, not just for us. *Science robotics*, 11 (113):aea1822, 2026.

Dayi Dong, Maulik Bhatt, Seoyeon Choi, and Negar Mehr. Mimic-d: Multi-modal imitation for multi-agent coordination with decentralized diffusion policies. *arXiv preprint arXiv:2509.14159*, 2025.

Ria Doshi, Tian Gao, Annie Chen, Chelsea Finn, and Jeannette Bohg. Chorus: Decentralized multi-embodiment collaboration with one vla policy, 2026. URL `https://arxiv.org/abs/2606.12352`.

Jakob Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

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

Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2003.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 2020.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Hokyun Im, Euijin Jeong, Andrey Kolobov, Jianlong Fu, and Youngwoon Lee. TwinVLA: Data-efficient bimanual manipulation with twin single-arm vision-language-action models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Aleksandar Krnjaic, Raul D. Steleac, Jonathan D. Thomas, Georgios Papoudakis, Lukas Schäfer, Andrew Wing Keung To, Kuan-Ho Lao, Murat Cubuktepe, Matthew Haley, Peter Börsting, and Stefano V. Albrecht. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Andrew Lee, Ian Chuang, Ling-Yuan Chen, and Iman Soltani. InterACT: Inter-dependency aware action chunking with hierarchical attention transformers for bimanual manipulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Kehui Liu, Zixin Tang, Dong Wang, Zhigang Wang, Xuelong Li, and Bin Zhao. COHERENT: Collaboration of heterogeneous multi-robot system with large language models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2025a.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: a diffusion foundation model for bimanual manipulation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025b.

Guanxing Lu, Tengbo Yu, Haoyuan Deng, Season Si Chen, Yansong Tang, and Ziwei Wang. Anybimanual: Transferring unimanual policy for general bimanual manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Yiyang Lu, Yufeng Tian, Zhecheng Yuan, Xianbang Wang, Pu Hua, Zhengrong Xue, and Huazhe Xu. H^3dp: Triply-hierarchical diffusion policy for visuomotor learning. In *International Conference on Learning Representations*, 2026.

Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, 2021.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Chen Yifan, Mingjie Wei, Xuesong Wang, Yuanxing Liu, Jizhe Wang, Hao Song, Longxuan Ma, Donglin Di, Churui Sun, Kaifeng Liu, Le Qi, Jianing Yu, Xiang Tian, Shiqi Liang, Chuanxi Duan, Zijian Hong, Weinan Zhang, and Ting Liu. Embodied ai: A survey on the evolution from perceptive to behavioral intelligence. *SmartBot*, 1(3):e70003, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

APPENDIX

A Training Algorithm

B Detailed Architectural Parameters & Training Budget

- B.1 CoLA Specifications and Hyperparameters
- B.2 Language Instructions
- B.3 Trainable Capacity and Training Cost

C Task Suite

- C.1 Qualitative Trajectory Comparison

D Full Experiment Results

- D.1 Two-Arm Handover Results
- D.2 Three-Arm Handover Chain Results
- D.3 Occluded Tray-Handover Results
- D.4 Linear Probes of Channel Content
- D.5 Zero-Shot Pretrained Octo
- D.6 Validation Gripper Accuracy

E Additional Training Dynamics

F Failure Mode Taxonomy

- F.1 Upstream Grasp Failures
- F.2 Temporal Desynchronisation Failures
- F.3 Post-Transfer Release Failures

A TRAINING ALGORITHM

Algorithm 1 CoLA Optimisation with Offline Feature Caching (diffusion U-Net head)

Require: Multi-agent demonstration dataset $\mathcal{D} = \{(\boldsymbol{o}^t_{1:N}, \boldsymbol{q}^t_{1:N}, \boldsymbol{a}^t_{1:N})\}_{t=1}^{T}$ (images, proprioception, actions); frozen VLA backbone f_θ with instruction $\boldsymbol{l}$; chunk horizon $K = 10$; message dimension $d_m = 64$; diffusion steps $S = 100$ with cosine schedule $\{\bar{\alpha}_\tau\}_{\tau=0}^{S-1}$; gripper weight $\lambda = 0.25$; proprio dropout $p = 0.1$.

// Phase 1: One-Time Offline Feature Extraction

Split $\mathcal{D}$ by episode into $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}$ (80/10/10).

for each split $\mathcal{D}_s \in \{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}\}$ **do**

  for each batch of multi-agent observations $\boldsymbol{o}_{1:N}$ **do**

    $\boldsymbol{f}_i \leftarrow \text{sg}\big[f_\theta(\boldsymbol{o}_i, \boldsymbol{l}).\texttt{readout_action}[:, -1, 0, :]\big], \quad \forall i \in \{1, \ldots, N\}$

    Cache $(\boldsymbol{f}_{1:N}, \boldsymbol{q}_{1:N}, \boldsymbol{a}_{1:N})$ to disk.

  end for

end for

// Phase 2: Action Chunk Indexing

Index valid starting timesteps $\mathcal{V} = \{t \mid [t, t+K-1]$ belongs to the same episode$\}$.

// Phase 3: Adapter Parameter Optimisation

Initialise $\boldsymbol{\phi} = \{\phi^i_{\text{prop}}, \phi^i_{\text{enc}}, \{\phi^{j\to i}_{\text{dec}}\}_{j\in\mathcal{N}(i)}, \phi^i_{\text{unet}}, \phi^i_{\text{grip}}\}_{i=1}^{N}$; best_val_loss $\leftarrow \infty$.

for epoch $= 1$ to 150 **do**

  for minibatch $\mathcal{B} \subset \mathcal{V}_{\text{train}}$ of size 128 **do**

    Load cached $\{\boldsymbol{f}^t_i, \boldsymbol{q}^t_i\}_{i=1}^{N}$ and target chunks $\{\boldsymbol{A}^t_i = \boldsymbol{a}^{t:t+K-1}_i\}_{i=1}^{N}$.

    for agent $i = 1$ to N **do**

      Drop proprioception: $\tilde{\boldsymbol{q}}^t_i \leftarrow b_i\, \boldsymbol{q}^t_i,\ b_i \sim \text{Bernoulli}(1-p)$.

      Self representation: $\boldsymbol{h}^t_i \leftarrow [\text{sg}[\boldsymbol{f}^t_i] \,\|\, \phi^i_{\text{prop}}(\tilde{\boldsymbol{q}}^t_i)]$.

      Encode message: $\boldsymbol{m}^t_i \leftarrow \phi^i_{\text{enc}}(\boldsymbol{h}^t_i)$.

    end for

    for agent $i = 1$ to N **do**

      Decode partner messages: $\boldsymbol{c}^t_{j\to i} \leftarrow \phi^{j\to i}_{\text{dec}}(\boldsymbol{m}^t_j),\ \forall j \in \mathcal{N}(i)$.

      Conditioning vector: $\boldsymbol{z}^t_i \leftarrow [\boldsymbol{h}^t_i \,\|\, \{\boldsymbol{c}^t_{j\to i}\}_{j\in\mathcal{N}(i)}]$.

      Split target: joint chunk $\boldsymbol{A}^{\text{jnt}}_i \in \mathbb{R}^{K\times 6}$, gripper labels $\boldsymbol{y}_i \leftarrow \mathbb{1}[\tilde{\boldsymbol{g}}^t_i > 0] \in \{0,1\}^K$.

      Sample $\tau_i \sim \mathcal{U}\{1, \ldots, S-1\}$ and $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, independently per agent.

      Noise the joints: $\tilde{\boldsymbol{A}}_i \leftarrow \sqrt{\bar{\alpha}_{\tau_i}}\, \boldsymbol{A}^{\text{jnt}}_i + \sqrt{1-\bar{\alpha}_{\tau_i}}\, \boldsymbol{\epsilon}_i$.

      Predict noise: $\hat{\boldsymbol{\epsilon}}_i \leftarrow \phi^i_{\text{unet}}(\tilde{\boldsymbol{A}}_i, \tau_i, \boldsymbol{z}^t_i)$ ▷ 1D temporal U-Net, FiLM on $\text{MLP}([\text{emb}(\tau_i) \| \boldsymbol{z}^t_i])$

      Predict gripper logits: $\boldsymbol{g}_i \leftarrow \phi^i_{\text{grip}}(\boldsymbol{z}^t_i) \in \mathbb{R}^K$.

    end for

    Compute joint loss: $\mathcal{L}(\boldsymbol{\phi}) \leftarrow \sum_{i=1}^{N} \Big(\frac{1}{6K} \|\hat{\boldsymbol{\epsilon}}_i - \boldsymbol{\epsilon}_i\|_2^2 + \lambda\, \frac{1}{K} \sum_{k=1}^{K} \text{BCE}(g_{i,k}, y_{i,k}) \Big)$, averaged over $\mathcal{B}$.

    Update $\boldsymbol{\phi} \leftarrow \text{Adam}(\boldsymbol{\phi}, \nabla_{\boldsymbol{\phi}}\mathcal{L})$ with gradient clipping $\|g\|_2 \le 1.0$ and cosine learning-rate decay $10^{-4} \to 10^{-5}$.

  end for

  Evaluate $\mathcal{L}_{\text{val}}$ on $\mathcal{V}_{\text{val}}$ (no proprio dropout; fixed $\tau, \boldsymbol{\epsilon}$ draws).

  if $\mathcal{L}_{\text{val}} <$ best_val_loss **then**

    Save optimal parameters $\boldsymbol{\phi}^* \leftarrow \boldsymbol{\phi}$; best_val_loss $\leftarrow \mathcal{L}_{\text{val}}$.

  else if epochs without validation improvement > 25 **then**

    break (Early stopping triggered).

  end if

end for

return $\boldsymbol{\phi}^*$

B Detailed Architectural Parameters & Training Budget

B.1 CoLA Specifications and Hyperparameters

Table 3: **CoLA specifications and hyperparameters.** Parameter counts are exact, read from the trained checkpoints; times are on one NVIDIA A100-SXM4-40GB. Channel $= \phi_{\text{enc}} + \phi_{\text{dec}}$; the proprioceptive encoder ($4{,}672$ parameters per arm) is counted with the action head. Percentages are relative to the frozen backbone (93M Octo-Base, $\sim$3.3B $\pi_{0.5}$).

Component	Value
Backbone feature ($\boldsymbol{f}_i$)	Octo-Base `readout_action` token, $d = 768$, frozen
Proprio encoder (ϕ_{prop})	Dense$(7 \to 64)$–ReLU–Dense$(64 \to 64)$; dropout $p = 0.1$ in training
Self representation ($\boldsymbol{h}_i$)	$\mathbb{R}^{832}$ (wrist-only); $\mathbb{R}^{1600}$ with the overhead feature
$\pi_{0.5}$ state input	Own 7-d state as prompt tokens (CoLA and LoRA alike); CoLA also via ϕ_{prop}
Message encoder (ϕ_{enc})	Dense$(832 \to 96)$–ReLU–Dense$(96 \to 64)$, $d_m = 64$ (input 1600 with overhead)
Message decoder (ϕ_{dec})	Dense$(64 \to 96)$–ReLU–Dense$(96 \to 768)$, one per ordered pair
Joint denoiser (ϕ_{unet})	1D U-Net, channels $(128, 256)$, kernel 5, GroupNorm(8), Mish; FiLM from Dense(256)–Mish–Dense(256) on $[\text{emb}_{64}(\tau) \,\|\, \boldsymbol{z}_i]$
Gripper head (ϕ_{grip})	Dense$(\boldsymbol{z}_i \to 128)$–Mish–Dense$(128 \to K)$, binary logits
Diffusion and loss	$S = 100$, cosine schedule, ϵ-prediction, 10-step DDIM; ϵ-MSE $+ 0.25$ BCE
Trainable: 2-arm and tray, wrist	14.47M (15.6%), channel 0.33M (0.36%)
Trainable: 2-arm, overhead	15.20M (16.3%), channel 0.48M (0.52%)
Trainable: 3-arm, overhead	23.93M (25.7%), channel 0.96M (1.04%)
Trainable: tray, $\pi_{0.5}$	16.93M ($\sim$0.5%), channel 0.83M ($\sim$0.03%)
Optimisation	Adam ($\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, no weight decay), clip 1.0; cosine LR $10^{-4} \to 10^{-5}$ over 150 epochs; batch 128
Model selection	Best validation loss; early stopping patience 25
Episodes (train/val/test)	2-arm $120/15/15$; tray $120/15/15$; 3-arm $104/13/13$
Compute	Feature caching $\sim$5–12 min per dataset; training $\sim$35 min (2-arm), $\sim$45 min (3-arm), $\sim$11 min ($\pi_{0.5}$)

B.2 Language Instructions

Table 4: **Language instructions.** The box is red in every task, and in the tray task only its marker changes colour, so no instruction reveals the hidden information. The role-specific instructions give the baselines more task information than CoLA; on the tray task, the $\pi_{0.5}$ baseline's Arm B is also told that a tray exists.

Model	Arm	Instruction
CoLA (Octo-Base and $\pi_{0.5}$), all tasks	all	`coordinate with partner`
Octo baselines, decentralised	A	`pick up the red box and hand it to the other arm`
	B, C	`take the red box from the other arm`
Octo baselines, centralised	–	`pick up the red box with one arm and hand it to the other`
$\pi_{0.5}$ LoRA baseline (tray)	A	`hand the box to the other arm`
	B	`take the box from the other arm and put it in a tray`

B.3 Trainable Capacity and Training Cost

Table 5: **Trainable capacity and training cost by task and method.** Every method within a task uses the same optimiser steps at batch size 128, which matches the *optimisation* budget but not compute or capacity: CoLA trains on cached backbone features, so it does a fraction of the work per step. Trainable parameters are the modules updated by gradient descent, summed over the policies a method deploys (two for the 2-arm tasks, three for the 3-arm task); the Octo baselines' action head is U-Net at horizon 12 (68.5M), and the centralised variants differ only in output width ($7 \rightarrow 14$), adding <0.1M. GPU-hours are job wall-clock on one NVIDIA A100-SXM4-40GB, summed over the policies a method deploys; CoLA trains all arms in a single model. They exclude CoLA's one-time feature extraction (5–12 min per Octo-Base dataset, amortised over every CoLA run and ablation on that dataset). The centralised policy encodes both arms' cameras in every forward pass, which is why on the tray task it costs more than the two decentralised policies combined.

Task	Method	Steps	Trainable	GPU-h
Occluded tray handover (wrist only), 2 arms				
	CoLA (with message)	20,250	14.5M	0.6
	CoLA (message zeroed)	20,250	14.5M	0.6
	Decentralised, head-only	20,250	137.0M	6.0
	Centralised, head-only	20,250	68.5M	6.4
	Decentralised, full fine-tune	20,250	323.0M	6.1
	Centralised, full fine-tune	20,250	161.5M	6.7
2-arm handover (overhead)				
	CoLA (with message)	12,600	15.2M	0.6
	CoLA (message zeroed)	12,600	15.2M	0.6
	Decentralised, head-only	12,600	137.0M	3.7
	Centralised, head-only	12,600	68.5M	2.1
	Decentralised, full fine-tune	12,600	323.0M	3.8
	Centralised, full fine-tune	12,600	161.5M	2.1
3-arm handover chain (overhead)				
	CoLA (with message)	26,850	23.9M	0.8
	CoLA (message zeroed)	26,850	23.9M	0.8
	Decentralised, head-only	26,850	205.5M	11.8[†]
	Decentralised, full fine-tune	26,850	484.5M	12.0[†]
Occluded tray handover, $\pi_{0.5}$ backbone				
	CoLA (with message)	20,250	16.9M	0.2
	Decentralised, LoRA fine-tune	20,250	934.0M	48.8

[†] Per-arm job wall-clock: head-only 3.95, 3.94 and 3.94 h; full fine-tune 4.02, 4.01 and 4.01 h.

C Task Suite

C.1 Qualitative Trajectory Comparison

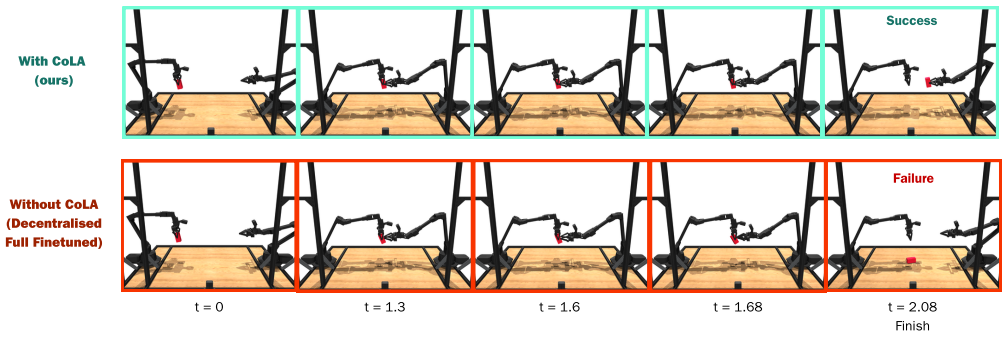

Figure 3: **Qualitative Trajectory Comparison on Handover.** Top: CoLA enables Arm A to maintain grasp stability until Arm B firmly closes its gripper. Bottom: Uncoordinated baseline desynchronises, causing premature release and dropped objects.

D Full Experiment Results

D.1 Two-Arm Handover Results

Table 6: **2-Arm handover (overhead camera), per evaluation seed.** Success rate (%) per 200-episode block; the seed sets the randomised start state. Octo-Base baselines use their wrist camera and the overhead camera.

Method	0	1000	2000	3000	4000	Mean ± std
CoLA (with message)	88.0	91.0	91.5	92.5	90.0	90.6 ± 1.7
CoLA (message zeroed)	90.5	85.0	89.5	–	–	88.3 ± 2.9
Decentralised, head-only	66.5	66.0	69.0	67.5	72.5	68.3 ± 2.6
Centralised, head-only	62.5	62.0	68.5	63.5	73.0	65.9 ± 4.7
Centralised, full fine-tune	91.0	91.0	94.5	94.0	90.0	92.1 ± 2.0
Decentralised, full fine-tune	78.0	85.5	81.0	82.5	85.0	82.4 ± 3.1

Table 7: **2-Arm handover (wrist cameras only), per evaluation seed.** Success rate (%) per 200-episode block at evaluation-seed offsets 0, 1000 and 2000.

Method	0	1000	2000	Mean ± std
CoLA (with message)	75.5	77.0	72.0	74.8 ± 2.6
CoLA (message zeroed)	61.5	59.5	72.0	64.3 ± 6.7
Decentralised, head-only	42.0	43.5	46.0	43.8 ± 2.0
Centralised, head-only	19.5	23.5	20.0	21.0 ± 2.2
Centralised, full fine-tune	81.5	81.5	77.0	80.0 ± 2.6
Decentralised, full fine-tune	71.0	69.5	67.5	69.3 ± 1.8

D.2 THREE-ARM HANDOVER CHAIN RESULTS

Table 8: **3-Arm handover chain (overhead camera), per evaluation seed.** Success = both transfers completed and arm C holds the box for 20 steps; 200-episode blocks. CoLA rows have one block each; paired on the same 200 start states, message on vs. zeroed is 127 vs. 120 successes (exact McNemar $p = 0.50$).

Method	**0**	**1000**	**2000**	**3000**	**4000**	**Mean ± std**
CoLA (with message)	63.5	–	–	–	–	63.5†
CoLA (message zeroed)	60.0	–	–	–	–	60.0†
Decentralised, head-only	20.0	16.5	21.0	21.0	24.0	20.5 ± 2.7
Decentralised, full fine-tune	36.0	24.0	33.0	32.0	33.0	31.6 ± 4.5

Table 9: **3-Arm chain, per-stage rates (%)**, pooled over all evaluated blocks: each conditional is computed as the total over the total across blocks, not as a mean of per-block rates. A→B: first transfer. B→C|A→B: second transfer among episodes that completed the first. Held|B→C: arm C holds the box for the required 20 steps, among episodes that completed the second transfer. Success is the product of the three stages, e.g. $42.9 \times 51.3 \times 93.2 = 20.5$ for the head-only baseline.

Method	**Blocks**	n	**Success**	**A→B**	**B→C|A→B**	**Held|B→C**
CoLA (with message)	1	200	63.5	79.5	79.9	100.0
CoLA (message zeroed)	1	200	60.0	75.0	80.0	100.0
Decentralised, head-only	5	1000	20.5	42.9	51.3	93.2
Decentralised, full fine-tune	5	1000	31.6	50.1	64.1	98.4

D.3 Occluded Tray-Handover Results

Table 10: **Occluded tray task, seed-0 block.** All methods on the same 150 episodes (seed block 0, 50 per marker colour). Success = correct tray over all episodes, with 95% Wilson interval. Handover and Reached tray are rates over all episodes; Correct|handover and Correct|tray condition on those stages (chance on Correct|tray is 33.3%).

Method	n	Success	Handover	Reached tray	Correct|handover	Correct|tray
CoLA (with message)	150	76.0 [68.6, 82.1]	92.7	88.7	82.0	85.7
CoLA (message zeroed)	150	1.3 [0.4, 4.7]	8.0	6.0	16.7	22.2
Decentralised, head-only	150	22.7 [16.7, 30.0]	88.7	86.0	25.6	26.4
Centralised, head-only	150	16.0 [11.0, 22.7]	68.0	43.3	23.5	36.9
Centralised, full fine-tune	150	10.7 [6.7, 16.6]	79.3	10.7	13.4	100.0
Decentralised, full fine-tune	150	12.0 [7.7, 18.2]	94.7	36.7	12.7	32.7
CoLA on frozen $\pi_{0.5}$	150	37.3 [30.0, 45.3]	53.3	52.0	70.0	71.8
Decentralised, LoRA $\pi_{0.5}$	150	34.7 [27.5, 42.6]	100.0	100.0	34.7	34.7

Table 11: **CoLA ablations on the occluded tray task.** Message width: the same 150 episodes (seeds 0, 50 per marker colour) for every width. Lighting: out-of-distribution scene lighting, 30 episodes each.

Ablation	Setting	n	Correct	Correct|handover
Message width d_m	16	150	73.3	79.7
Message width d_m	32	150	42.0	47.4
Message width d_m	64 (default)	150	76.0	82.0
Message width d_m	128	150	73.3	76.9
Lighting (OOD)	bright	30	3.3	50.0
Lighting (OOD)	dark	30	10.0	15.8
Lighting (OOD)	dim	30	13.3	19.0

D.4 Linear Probes of Channel Content

Table 12: **Linear probes on CoLA messages.** Tray: multinomial logistic regression to marker colour, split by episode (chance 33.3%); frame = per-timestep accuracy, episode = majority vote. Handover: linear regression to box position. "Zeroed" probes the encoder of the model trained with the channel severed.

Task / protocol	Representation	Dim	Frame acc.	Episode acc.
Tray, 5-fold CV (150 ep.)	self_a	832	100.0	100.0
Tray, 5-fold CV (150 ep.)	msg_a	64	44.2	73.3
Tray, 5-fold CV (150 ep.)	msg_a (zeroed model)	64	39.7	33.3
Handover (wrist only)			R^2 (y / z)	**RMSE cm** (y / z)
Box position	self_a	832	1.000 / 1.000	0.04 / 0.08
Box position	msg_a	64	1.000 / 0.985	0.12 / 0.45
Box position	msg_a (zeroed model)	64	0.999 / 0.982	0.13 / 0.50

Pose probe. On the wrist-only 2-arm handover, Arm A's presentation pose (y, z) is decodable from self ($R^2 = 1.000 \,/\, 1.000$), message ($1.000 \,/\, 0.985$), and control ($0.999 \,/\, 0.982$) alike (Fig. 4, right; Table 12). Because Arm A rigidly grasps the object, pose is near-linear in its proprioception and survives even a random projection, so the probe cannot attribute pose to training through the channel. It does, however, show that pose is reliably present in the message, and whether it matters depends on what Arm B can already see. With the overhead view, B observes the handoff directly and zeroing the message leaves success unchanged (90.6% vs. 88.3%). With only a wrist view, the message becomes B's main source of A's pose, and zeroing it drops success from 74.8% to 64.3%.

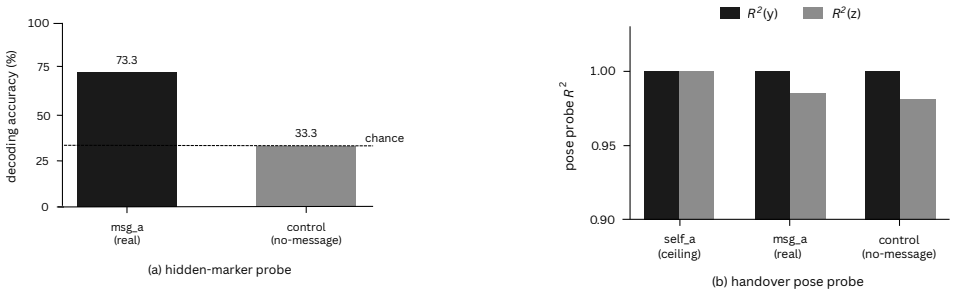

Figure 4: **Linear probes of channel content.** *Left:* on the occluded tray-handover task, the message decodes box colour well above chance while an untrained control does not, indicating channel-specific information transfer. *Right:* on the wrist-only handover, the message and an untrained control both decode Arm A's presentation pose near ceiling, so pose is linearly recoverable from the raw features regardless of training, and this probe cannot isolate the channel's contribution.

D.5 Zero-Shot Pretrained Octo

Table 13: **Zero-shot pretrained Octo** (no fine-tuning; Bridge action statistics; overhead camera; 200 episodes, 350 control steps). Full task: pick up and hand over. HO: episodes start with arm A holding the box.

Backbone	Task	Success	Transfer	A touches box	B touches box
Octo-Small	Full task	0.0	0.0	0.0	0.0
Octo-Small	HO	0.0	0.0	29.0	0.0
Octo-Base	Full task	0.0	0.0	2.5	1.0
Octo-Base	HO	0.0	0.0	11.0	0.0

D.6 Validation Gripper Accuracy

Table 14: **Validation gripper accuracy (%)** per arm, taken at each run's best-validation epoch, for the CoLA runs behind Table 1. —: arm not present in the task.

Task	Arm A	Arm B	Arm C
2-Arm handover (wrist only)	98.8	98.8	—
2-Arm handover (overhead)	98.9	98.7	—
Occluded tray handover	100.0	100.0	—
3-Arm handover chain	99.7	99.5	99.8

E Additional Training Dynamics

Fig. 5 shows validation loss for every training run behind Table 1. All runs use the matched budget of 150 epochs at batch size 128; the Octo baselines' optimiser steps are converted to epochs by dividing by that task's steps per epoch (84, 135 and 179 for the two-arm, tray and three-arm tasks). No CoLA run triggered early stopping, and the best validation epoch ranged from 131 to 149 across all CoLA runs, so every run consumed its 150-epoch budget in full.

Two patterns are worth noting. First, the gap between CoLA with messages and with messages zeroed is present on every task but varies several-fold in relative terms, and its ordering matches the behavioural results: final validation loss is 0.0009 vs. 0.0039 on the occluded tray task (4.3×), against 0.0243 vs. 0.0283 (1.2×) on the two-arm overhead task, where zeroing messages costs nothing behaviourally. Second, loss does not rank methods the way success rate does: the centralised policies have the *highest* validation loss on every task, since fitting a 14-D joint action is harder than fitting a 7-D one, yet centralised full fine-tuning attains the best success rate on the two-arm overhead task. This is why we report success rates rather than losses throughout.

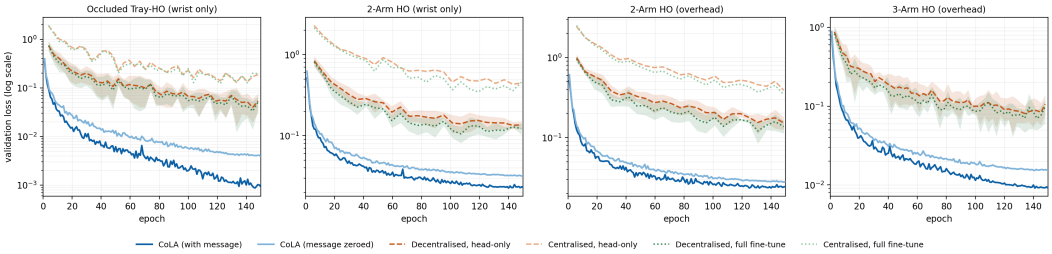

Figure 5: **Validation loss for all training runs in Table 1**, one panel per coordination regime, ordered as in the main table. Octo steps are converted to epochs so both families share the matched 150-epoch budget. Decentralised rows train one policy per arm; the line is their mean and the shading their range.

F Failure Mode Taxonomy

Systematic inspection of failed trials identifies three orthogonal failure modes:

F.1 Upstream Grasp Failures

The initial grasping agent (Arm A) fails to secure the object due to physics edge cases during spawning. This failure is unimanual and independent of communication.

F.2 Temporal Desynchronisation Failures

Observed chiefly in the two-arm handover. Arm A ungrasps before Arm B secures contact, or Arm B closes prematurely.

F.3 Post-Transfer Release Failures

Specific to the three-arm chain, where success additionally requires Arm C to hold the object for 20 steps after the second transfer. Arm C completes that transfer but loses the object during the hold in 15 of the 220 completed second transfers for the decentralised head-only baseline and 5 of 321 for decentralised full fine-tuning, and in none of CoLA's 127. This mode is minor next to the transfers themselves: most failed three-arm episodes end at A→B or B→C rather than during the hold (Table 9).