WHOSE TURN TO ADAPT? DYNAMICALLY RANKED SPARSE LOW-RANK ADAPTATION FOR ONBOARD SATELLITE TASKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Earth-observation satellite clusters increasingly rely on heterogeneous multi-agent reinforcement learning policies to schedule tasks and allocate resources, yet these policies are trained offline and flown frozen, leaving them pursuing outdated goals whenever mid-mission priority shifts outpace ground-station uplink windows. Adapting avoid this delays,low rank adapter (LoRA) finetune small portion of parameter locally reducing computational burden. However, exisitng rank allocators measure importance from parameter on a fixed calibration set, treating as a single network in isolation and enforcing sparsity by masking, still carrying full-sized tensors that are infeasible for resource-constrained satellites. We propose DRS-LoRA, an onboard *D*ynamic *R*anking based *S*parse LoRA method that prunes rank directions by the singular value decomposition of each adapter's output drift to train and fientune extremely few paramter when performance degrades when new tasks arrives. Across satellite agents, the budget is split by each agent's contribution to the HAPPO advantage decomposition, a signal the policy update already computes. Within each agent, rank directions are ordered by the singular value decomposition of the output drift the adapter induces, so truncation gives the minimum output error for that rank. On a simulated four-satellite heterogeneous cluster with adaptation onboard through Jetson Orin Nano, DRS-LoRA takesd only 13.7% of the joint policy trainable parameter during transfer learning on new tasks, retaining 92.1% of the full fine-tuning target completion rate.

1 INTRODUCTION

Earth-observation (EO) satellite tasking has traditionally relied on ground-based planners that compute schedules offline and uplink them during contact windows, leaving spacecraft unable to respond between contacts. Recent work instead learns tasking policies on the ground and executes them onboard (Herrmann & Schaub, 2022; 2023), and heterogeneous multi-agent reinforcement learning (MARL) extends this to clusters of satellites with different payloads acting toward a shared objective (Kuba et al., 2022; Zhong et al., 2024). These policies stay fixed after deployment. When the mission objective changes in orbit, a change we call *tasking drift*, the constellation keeps pursuing its original objective until a retrained policy is uplinked, which reintroduces the contact dependence that onboard execution was meant to remove.

Fine-tuning the policy onboard, from its own rollouts, avoids this delay, but full fine-tuning keeps gradients and optimizer state for every parameter of every agent on a processor with limited memory, compute and power (Lin et al., 2024; Zhu et al., 2023). Parameter-efficient fine-tuning (PEFT) bounds both the cost of an update and its deviation from the validated policy. PEFT has, however, been developed for large models adapted offline on fixed data, and to our knowledge it has not been studied for onboard fine-tuning of deployed multi-agent policies.

Low-rank adaptation (LoRA) (Hu et al., 2022) trains a low-rank update over frozen weights, and later methods allocate rank across modules using sensitivity scores, sparse gates or random rank dropout (Zhang et al., 2023a; Ding et al., 2023). These schemes infer importance from parameters, estimate it once, and adapt a single network, and none of these conditions holds onboard. Parameter statistics do not measure how a rank component changes actions in visited states, on-policy data

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.

shift as training proceeds, and in a constellation one agent's update changes what the others need, a coupling that fixed per-agent ranks cannot capture (Zhang et al., 2025). Masked or gated sparsity also keeps full-sized tensors in memory until training ends.

We propose DRS-LoRA, which uses the output drift of each adapter as its importance measure (Li et al., 2026). Sparsity is defined over rank directions. Each agent's directions are ranked by the drift energy they induce on current rollouts and re-ranked at fixed intervals, and a single rank budget is shared across agents according to their contribution to the joint advantage. Pruned directions are deleted with their gradients and optimizer state, and the adapter is merged back after fine-tuning, so onboard inference cost is unchanged. We evaluate on a four-satellite heterogeneous constellation with inference and fine-tuning executed on a Jetson Orin Nano. At a matched budget, drift-based ranking outperforms Fisher-based ranking on every transfer, whereas weight-norm ranking falls below the unpruned adapter. Our main contributions are as follows.

- To our knowledge, the first framework that both executes and fine-tunes a heterogeneous MARL tasking policy onboard, adapting to tasking drift without ground retraining.
- DRS-LoRA, a sparse low-rank fine-tuning method that ranks, re-ranks and removes rank directions by their output drift under a single rank budget shared across agents.
- With 13.7% of the joint policy trainable, DRS-LoRA attains 92.1% of the completion rate of full fine-tuning and outperforms every onboard-feasible baseline on all ten transfers, including baselines that train up to 2.2× as many parameters.

2 Related Work

LoRA freezes the pretrained weights and trains a low-rank update that is merged back after fine-tuning, so inference cost is unchanged (Hu et al., 2022). Later work varies how much capacity each module receives. AdaLoRA allocates rank by parameter sensitivity (Zhang et al., 2023b), and E^2-LoRA uses output feature drift to identify which directions to retain (Li et al., 2026). Among methods that combine rank adaptation with sparsity, SoRA learns gates over LoRA directions (Ding et al., 2023), RoSA trains low-rank and sparse weight updates jointly (Hameed et al., 2024), and LoSA adjusts adapter rank while adapting a sparse base model (Zhang et al., 2025). These methods were developed for a single large model fine-tuned offline on fixed data. Their importance estimates are computed from parameters or a fixed dataset, and gated or masked variants keep full-sized tensors and optimizer state until pruning is applied. DRS-LoRA instead ranks directions by the output drift they produce on current rollouts, recomputes this ranking as the policy changes, deletes pruned directions during training, and divides a single rank budget across several separately parameterized policies.

3 Method

We model the constellation as a Dec-POMDP with N heterogeneous agents, where agent i acts through its own policy $\pi^i_{\theta^i}(a^i_t \mid o^i_t)$ and the joint policy $\boldsymbol{\pi}_\theta = (\pi^1_{\theta^1}, \ldots, \pi^N_{\theta^N})$ is trained cooperatively with HAPPO (Kuba et al., 2022). Its sequential update yields an agent-specific advantage $\hat{A}^i$, which we reuse for rank allocation in Sec. 3.1 without an extra network or rollout. A task is a reward weighting $\omega = (\omega_1, \omega_2)$ that trades the priority $\rho_{j,t}$ of each captured target $j \in \mathcal{C}_t$ against its cloud-related utility $\varsigma_{j,t}$. The reward averages this trade-off over the $\tau_t = |\mathcal{C}_t|$ captures at time t, and the return accumulates it over an episode:

$$R_\omega(s_t, \mathbf{a}_t) = \frac{1}{\max(\tau_t, 1)} \sum_{j \in \mathcal{C}_t} \big(\omega_1 \rho_{j,t} + \omega_2 \varsigma_{j,t}\big), \qquad J_\omega(\theta) = \mathbb{E}_{\boldsymbol{\pi}_\theta}\Big[\textstyle\sum_{t=0}^{T-1} \gamma^t R_\omega(s_t, \mathbf{a}_t)\Big]. \quad (1)$$

The ground-trained parameters satisfy $\theta_0 \in \arg\max_\theta J_{\omega_{\mathrm{src}}}(\theta)$. *Tasking drift* is a post-deployment change $\omega_{\mathrm{src}} \to \omega_{\mathrm{tgt}}$ under fixed states, dynamics, observations, actions and discount, so the pretrained policy remains feasible but optimizes the wrong objective. We correct it by keeping each layer's pretrained weight $W_0^{i,l} \in \mathbb{R}^{d^l_{\mathrm{out}} \times d^l_{\mathrm{in}}}$ frozen and adding a LoRA update (Hu et al., 2022) whose rank r_i is shared by the adapted layers of agent i. Adaptation then seeks the best target return

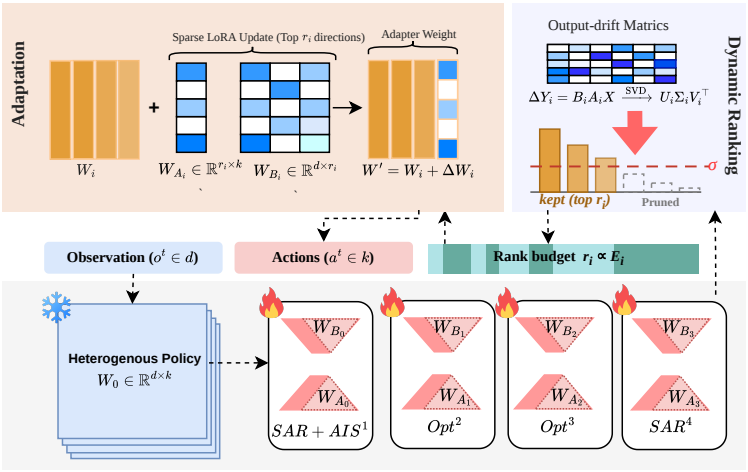

Figure 1: **DRS-LoRA overview.** shared rank budget is split across agents by $r_i \propto \mathcal{E}_i$, and each adapter's directions are dyanmicly ranked by output-drift energy $\sigma^2_{i,l,k}$ from the SVD of $\Delta Y_i = B_i A_i X$. Only the top-r_i directions are kept and low-energy ones are deleted from B_i, A_i and their optimizer state, re-ranked every τ episodes.

that a shared rank budget R allows:

$$W^{i,l} = W_0^{i,l} + \tfrac{\alpha_i}{r_i} B^{i,l} A^{i,l}, \qquad \max_{\Delta\theta} J_{\omega_{\text{tgt}}}(\theta_0 + \Delta\theta) \ \text{ s.t. } \ \textstyle\sum_{i=1}^{N} r_i = R, \ \ 0 \le r_i \le r_{\max}, \tag{2}$$

where $B^{i,l} \in \mathbb{R}^{d^l_{\text{out}} \times r_i}$ and $A^{i,l} \in \mathbb{R}^{r_i \times d^l_{\text{in}}}$. The inputs reaching layer l over a rollout batch, $X^{i,l} = [x_1^{i,l}, \ldots, x_n^{i,l}] \in \mathbb{R}^{d^l_{\text{in}} \times n}$, are the internal representations on which drift is later measured, not raw observations. The following subsections decide how R is split across agents (§3.1), which directions each agent keeps and when they are re-ranked (§3.2), which parameters are trainable at a given moment (§3.3), and how fast they are updated (§3.4), as summarized in Fig. 1.

3.1 Selection: Cross-Agent Rank Allocation

The shared budget determines how much capacity each agent receives, but not which directions fill that capacity. We drive the split with the agent-specific advantage which HAPPO already computes. The allocation signal $\mathcal{E}_i$ is the square of this advantage averaged over the rollout, as defined in Eq. (3).

$$\mathcal{E}_i = \frac{1}{n} \sum_{t=1}^{n} \left(\hat{A}_t^i\right)^2 \tag{3}$$

The budget is then divided in proportion to this signal and rounded down to integer ranks, as shown in Eq. (4).

$$r_i = \left\lfloor R \, \frac{\mathcal{E}_i}{\sum_{j=1}^{N} \mathcal{E}_j} \right\rfloor, \qquad \textstyle\sum_{i=1}^{N} r_i \le R \tag{4}$$

The signal is available as soon as HAPPO has formed the reweighted advantage of agent i, so Eq. (4) needs no separate forward pass, auxiliary critic or calibration rollout, unlike importance-scoring schemes that score parameters on a dedicated pass before allocating (Molchanov et al., 2019; Zhu et al., 2023). The split is recomputed every τ episodes from the value of $\mathcal{E}_i$ at that episode, and the resulting r_i is the cut applied to the ranking of §3.2.

3.2 Dynamic Ranking: Ordering Directions by Output-Drift Energy

The useful directions of an adapter can change as the policy collects new rollouts. Every τ episodes, we therefore measure the change in each adapted layer's output on the current rollout. For agent i

and layer l, let r_i^- and r_i^+ be the ranks before and after allocation. The output drift via its singular value decomposition (shown in 5 measure how much of the observed output changes in each ordered direction. Following the output-drift construction of Li et al. (2026), we retain the smallest prefix that captures a fraction ρ of its energy based the allocated rank as Eq. (6).

$$\Delta Y^{i,l} = \Delta W^{i,l} X^{i,l} = \frac{\alpha}{r_i} B^{i,l} A^{i,l} X^{i,l} \in \mathbb{R}^{d_{\text{out}}^l \times n} \tag{5}$$

$$k_{i,l}^{\rho} = \min \left\{ k : \sum_{j=1}^{k} \sigma_{i,l,j}^2 \geq \rho \sum_{j} \sigma_{i,l,j}^2 \right\}, \qquad k_{i,l} = \min\{r_i^+, k_{i,l}^{\rho}\}. \tag{6}$$

This spectrum is then cut to the budget of §3.1. Let r_i^- be the rank used since the previous event and r_i^+ the rank now assigned by Eq. (4). Layer l keeps the $k_{i,l}$ leading directions given by Eq. (7), so a direction survives only if it lies within both the energy threshold and the agent's budget.The adapter is re-expressed in the kept directions and extended with fresh ones up to rank r_i^+, as given in Eq. (8).

$$k_{i,l} = \min\left\{r_i^+,\, r_{i,l}^{\rho}\right\} \tag{7}$$

$$B^{i,l} \leftarrow \left[U^{i,l}_{:,1:k_{i,l}} \quad \mathbf{0} \right], \qquad A^{i,l} \leftarrow \begin{bmatrix} \frac{r_i^+}{r_i^-} \left(U^{i,l}_{:,1:k_{i,l}}\right)^\top B^{i,l} A^{i,l} \\ A^{i,l}_{\text{new}} \end{bmatrix} \tag{8}$$

In Eq. (8), the zero block has $r_i^+ - k_{i,l}$ columns, $A^{i,l}_{\text{new}}$ is initialized as in LoRA, the first blocks are empty when $k_{i,l} = 0$, and the factor r_i^+/r_i^- compensates for the change of the scale α/r_i in Eq. (8). On the recorded batch, the rebuilt adapter produces $U^{i,l}_{:,1:k_{i,l}}(U^{i,l}_{:,1:k_{i,l}})^\top \Delta Y^{i,l}$, the truncated-SVD reconstruction of the previous drift.As a result, reallocation therefore changes the layer output there by exactly $\sum_{j>k_{i,l}} \sigma_{i,l,j}^2$ in squared Frobenius norm. Further, the ranking is dynamic in that the spectrum is recomputed and Eq. (6) & (8) are re-applied every τ episodes, so the kept directions follow the drift under the current on-policy distribution rather than the distribution at initialization (Shenfeld et al., 2026; Wołczyk et al., 2024). Moreover, if the agent later receives more rank, the added capacity is a fresh direction rather than a restoration of the dropped one.

3.3 What to Adapt: Physical Truncation and Optimizer State

Dynamic Ranking fixes which directions an adapter keeps, and this section fixes what is physically present on the flight processor between events. After reallocation, each adapted layer of agent i holds exactly r_i^+ directions, the $k_{i,l}$ reprojected ones and the fresh ones, and its update takes the form in Eq. (9).

$$\Delta W^{i,l} = \frac{\alpha}{r_i^+} B^{i,l} A^{i,l}, \qquad B^{i,l} \in \mathbb{R}^{d_{\text{out}}^l \times r_i^+}, \quad A^{i,l} \in \mathbb{R}^{r_i^+ \times d_{\text{in}}^l} \tag{9}$$

Directions outside the kept set are removed from $B^{i,l}$ and $A^{i,l}$ rather than zeroed, so their parameters, gradients and optimizer moments cease to exist. Weight-level masking instead keeps full tensors in the optimizer, since the dense gradient is computed before the mask is applied (Zhang et al., 2024; Chen et al., 2025). Comparoing with gate driven toward zero by proximal descent, as in SoRA (Ding et al., 2023), reaches sparsity only asymptotically, so a nominally pruned direction can retain residual weight at any finite step. Deleting rank directions ties the trainable and optimizer footprint of agent i directly to its rank, as counted in Eq. (10) first and second moments of Adam optimizer.

$$|\theta_i^{\text{adapter}}| = r_i \sum_{l \in \mathcal{L}_i} \left(d_{\text{in}}^l + d_{\text{out}}^l\right), \qquad |\theta_i^{\text{optimizer}}| = 2\,|\theta_i^{\text{adapter}}| \tag{10}$$

All satellite actors in our constellation share layer shapes, as a result, the joint count is proportional to $\sum_i r_i \leq R$. Each adapter is initialized with $r_0 = 16$ directions, and a sparsity level s corresponds to $R = N\lceil(1-s)r_0\rceil$.We initialize each adapter at $r_0 = 16$; a requested rank sparsity s sets $R = N\lceil(1-s)r_0\rceil$. The retained factors form the update in $W^{i,l}_{\text{adapted}} = W_0^{i,l} + (\alpha_i/r_i)B^{i,l}A^{i,l}$. Comparing with layer or channel level sparsity (Zhu et al., 2023; Li et al., 2024; Yin et al., 2023) also avoids dense optimizer state, but within a rank-r_i adapter, removing a layer removes an agent's

only capacity there, whereas removing rank directions shrinks every adapter and leaves each layer some capacity (Lin et al., 2024).

3.4 Update Mechanism: Trust-Adaptive Gradient Pacing

Selection and Dynamic Ranking fix what is trainable, and pacing sets how far it moves through $\mathcal{E}_i$ which is estimated from a single finite rollout and shifts whenever any agent's policy changes, reflecting the variance of on-policy advantage estimates. We track the reliability of $\mathcal{E}_i$ with the exponential moving average in Eq. (11) and measure its current volatility as the relative deviation from that average in Eq. (12) and Eq. (13) converts this volatility into a bounded trust coefficient c_i.

$$\hat{\mathcal{E}}_i \leftarrow \beta\,\hat{\mathcal{E}}_i + (1-\beta)\,\mathcal{E}_i \tag{11}$$

$$v_i = \frac{\left|\mathcal{E}_i - \hat{\mathcal{E}}_i\right|}{\hat{\mathcal{E}}_i + \epsilon} \tag{12}$$

$$c_i = \mathrm{clip}\left(\frac{1}{1+v_i},\ \phi,\ 1\right), \qquad \phi \in (0,1) \tag{13}$$

The instantiated factors $\theta^i_{\mathrm{LoRA}} = \{B^{i,l}, A^{i,l}\}_{l\in\mathcal{L}_i}$ are updated with a step scaled by c_i, as given in Eq. (14) where u^i is the Adam direction computed from $\nabla_{\theta^i_{\mathrm{LoRA}}}\mathcal{L}^i_{\mathrm{HAPPO}}$.

$$\theta^i_{\mathrm{LoRA}} \leftarrow \theta^i_{\mathrm{LoRA}} - \eta\, c_i\, u^i \tag{14}$$

The ranking decide which directions exist, according to how strongly each changes the output, whereas c_i reflects how far the allocation signal behind that decision can currently be trusted and scales all directions of agent i equally.

3.5 Whole Process

Selection, Dynamic Ranking, What to Adapt, and the update mechanism compose into one onboard procedure operating at two nested timescales: a fast per-episode loop that trains the currently retained subspace, and a slower per-τ-episode loop that re-ranks and reallocates it. Within an episode, the constellation collects a rollout under the joint policy; the per-agent signal $\mathcal{E}_i$ is read directly off HAPPO's advantage decomposition (Eq. 3); the trust coefficient c_i is updated (Eqs. 11–13); and the parameters inside the current retained set are updated (Eq. 14). Every τ episodes, the drift energy $\sigma^2_{i,l,k}$ is recomputed for every layer of every agent (Eq. 5); Selection re-splits the shared budget R across agents (Eq. 4); Dynamic Ranking crops each agent's rank and reprojects the adapter onto it (Eqs. 6–8); and What to Adapt recomputes the retained set $\mathcal{S}^l_i$ (Eq. 7). No component in either loop requires an auxiliary network, a stored replay buffer, or a rollout beyond the one already collected for the policy update; the only added cost of reallocation is the drift-energy computation, incurred once per τ episodes rather than once per step. At the end of an adaptation window the adapter is merged into the base weight exactly as in Eq. 9, so between windows the constellation runs $W^{i,l}_0$ alone, with no adapter, no optimizer state, and no rank structure to track: the deployed forward pass is identical in cost to the unadapted policy.

3.6 Experimental Setup

3.6.1 Environmental Setup

We simulate a four-satellite heterogeneous cluster with one SAR-AIS trailing satellite, two optical satellite and one SAR platform, designed in Basilisk (Carneiro & Schaub, 2024) via BSK-RL (Stephenson & Schaub, 2024) under randomized hardware resource constraints. We trained HAPPO (Kuba et al., 2022) based joined policy based on these four satellite agent based on ground under a fixed source weighting ω_{src} and flown frozen; onboard adaptation is then evaluated across ten $\omega_{\mathrm{src}} \to \omega_{\mathrm{tgt}}$ tasking-drift transfers spanning three source classes including, Cloud, Target, and Balanced Priority. All inference and onboard adaptation run on a Jetson Orin Nano; Basilisk supplies only the orbital environment the cluster experiences, not the computation the method performs. Orbital configuration, per-satellite physical parameters, observation/action heterogeneity, resource-scenario design, and the onboard HAPPO architecture are given in Appendix 7.1.

Algorithm 1 Onboard adaptation: dynamic ranking with trust-adaptive pacing

Require: pretrained policy π_θ, agents $i = 1, \dots, N$, shared rank budget R, retention threshold ρ, reallocation interval τ, EMA rate β, trust floor ϕ

initialize adapters $\{B^{i,l}, A^{i,l}\}$ at a uniform rank, $\hat{\mathcal{E}}_i \leftarrow 0$
for episode $e = 1, 2, \dots$ **do**
  collect a rollout under π_θ; compute $\mathcal{E}_i$ for all i ▷ Eq. 3
  update $\hat{\mathcal{E}}_i$; compute v_i and c_i ▷ Eqs. 11–13
  update $\{[B^{i,l}, A^{i,l}]_k : k \in \mathcal{S}_i^l\}$ ▷ Eq. 14
  if $e \bmod \tau = 0$ **then**
    recompute $\sigma^2_{i,l,k}$ for all i, l ▷ Eq. 5
    re-split R across agents ▷ Eq. 4
    crop and reproject each agent's rank ▷ Eqs. 6–8
    recompute the retained sets $\mathcal{S}_i^l$ ▷ Eq. 7
  end if
end for
merge: $W_0^{i,l} \leftarrow W_0^{i,l} + \frac{\alpha_i}{|\mathcal{S}_i^l|} B^{i,l}_{:,\mathcal{S}_i^l} A^{i,l}_{\mathcal{S}_i^l,:}$ for all i, l; discard $B^{i,l}, A^{i,l}$
return adapted policy π_θ (Eq. 9 merged; no adapter remains at inference)

Table 1: **Completion rate (CR, %) across source to target reward-weight transfers** $(\mathbf{w}^s \to \mathbf{w}^t)$

Method	Param. (%)	Cloud Priority $\mathbf{w}^s$=(0.2, 0.8)			Avg	Target Priority $\mathbf{w}^s$=(0.8, 0.2)			Avg	Balanced Priority $\mathbf{w}^s$=(1.0, 1.0)				Avg	Overall Avg
		(0.8,0.2)	(1.0,0.0)	(1.0,1.0)		(0.2,0.8)	(0.0,1.0)	(1.0,1.0)		(0.2,0.8)	(0.8,0.2)	(0.0,1.0)	(1.0,0.0)		
Full FT	*100*	$41.1_{\pm 5.7}$	$40.2_{\pm 4.4}$	$41.4_{\pm 5.5}$	40.9	$40.2_{\pm 4.7}$	$42.2_{\pm 4.5}$	$41.4_{\pm 4.5}$	41.3	$38.7_{\pm 4.7}$	$40.7_{\pm 5.4}$	$39.0_{\pm 4.6}$	$41.1_{\pm 5.1}$	39.9	40.6
Sparse FT	29.9	$35.1_{\pm 5.4}$	$34.0_{\pm 4.6}$	$35.5_{\pm 5.2}$	34.9	$34.2_{\pm 5.0}$	$36.2_{\pm 5.0}$	$35.6_{\pm 4.8}$	35.3	$34.5_{\pm 5.4}$	$35.9_{\pm 5.5}$	$33.6_{\pm 4.9}$	$34.6_{\pm 5.1}$	34.6	34.9
LoRA (non-unif.)	27.4	$35.6_{\pm 4.9}$	$34.5_{\pm 4.5}$	$36.2_{\pm 4.9}$	35.4	$34.8_{\pm 5.0}$	$36.6_{\pm 4.6}$	$35.4_{\pm 4.4}$	35.6	$34.2_{\pm 4.0}$	$36.4_{\pm 4.3}$	$34.5_{\pm 4.4}$	$35.3_{\pm 4.4}$	35.1	35.4
SoRA (gated)	27.4	$33.2_{\pm 4.7}$	$32.4_{\pm 4.4}$	$34.8_{\pm 4.9}$	33.5	$32.5_{\pm 5.2}$	$34.9_{\pm 4.9}$	$33.6_{\pm 4.6}$	33.7	$32.1_{\pm 4.6}$	$34.3_{\pm 4.9}$	$31.9_{\pm 4.7}$	$33.4_{\pm 4.8}$	32.9	33.3
Sparse LoRA (Fisher)	13.7	$35.8_{\pm 4.0}$	$34.7_{\pm 3.7}$	$37.4_{\pm 4.7}$	36.0	$34.7_{\pm 4.9}$	$37.7_{\pm 4.2}$	$35.9_{\pm 4.2}$	36.1	$34.0_{\pm 3.6}$	$36.1_{\pm 3.6}$	$34.2_{\pm 4.0}$	$36.5_{\pm 3.9}$	35.2	35.7
Sparse LoRA (Mag.)	13.7	$34.9_{\pm 3.8}$	$31.8_{\pm 4.2}$	$36.4_{\pm 4.2}$	34.4	$32.2_{\pm 4.8}$	$35.6_{\pm 5.1}$	$33.5_{\pm 4.4}$	33.8	$33.1_{\pm 5.1}$	$35.8_{\pm 5.2}$	$32.8_{\pm 5.1}$	$34.8_{\pm 4.3}$	34.1	34.1
DRS-LoRA (s=0.7)	8.6	$34.6_{\pm 4.1}$	$33.0_{\pm 4.8}$	$34.1_{\pm 5.0}$	33.9	$33.5_{\pm 4.5}$	$34.7_{\pm 4.1}$	$33.7_{\pm 4.4}$	34.0	$32.9_{\pm 3.8}$	$35.0_{\pm 3.6}$	$33.4_{\pm 5.0}$	$33.7_{\pm 4.5}$	33.8	33.9
DRS-LoRA (s=0.5)	13.7	$\underline{38.0}_{\pm 3.3}$	$\underline{36.5}_{\pm 4.3}$	$\underline{37.7}_{\pm 4.5}$	<u>37.4</u>	$\underline{37.1}_{\pm 4.0}$	$\underline{38.3}_{\pm 3.6}$	$\underline{37.2}_{\pm 3.9}$	<u>37.5</u>	$\underline{36.4}_{\pm 3.3}$	$\underline{38.5}_{\pm 3.1}$	$\underline{37.0}_{\pm 4.5}$	$\underline{37.3}_{\pm 4.0}$	<u>37.3</u>	<u>37.4</u>
DRS-LoRA (s=0.3)	20.5	$\mathbf{39.4}_{\pm 3.6}$	$\mathbf{37.9}_{\pm 4.1}$	$\mathbf{39.0}_{\pm 4.3}$	**38.8**	$\mathbf{38.6}_{\pm 3.8}$	$\mathbf{39.6}_{\pm 3.4}$	$\mathbf{38.5}_{\pm 3.7}$	**38.9**	$\mathbf{37.9}_{\pm 3.2}$	$\mathbf{39.9}_{\pm 3.0}$	$\mathbf{38.4}_{\pm 4.3}$	$\mathbf{38.6}_{\pm 3.8}$	**38.7**	**38.8**

3.6.2 BASELINES

We compare DRS-LoRA against a ground-side upper bound (Full FT) and five onboard-feasible parameter-efficient baselines, chosen to isolate two design axes separately: *how much* of the policy is trainable, and *which* directions within that budget are selected. *Sparse FT* and *LoRA (non-unif.)* fix the trainable set without any drift-aware ranking; *SoRA* learns its sparsity rather than setting it; *Sparse LoRA (Fisher)* and *Sparse LoRA (Mag.)* match DRS-LoRA's exact rank granularity and sparsity levels $s \in \{0.3, 0.5, 0.7\}$, differing only in the ranking criterion. All PEFT baselines and DRS-LoRA are tuned and trained under matched optimizer, learning-rate, and rollout budgets; full hyperparameters, architectural placement of adapters, and the ranking-criterion implementations are given in Appendix 7.3.

4 MAIN RESULTS

The experiments examine whether a HAPPO policy pretrained on a source reward weighting can be fine-tuned onboard, on the Jetson Orin Nano, to recover the performance it loses under tasking drift, and what governs the quality of that recovery when the trainable budget is fixed by the hardware. Performance is measured by the completion rate (CR), the fraction of tasked targets captured after adaptation, and budgets are expressed as the percentage of the joint four-satellite policy that is trainable. Full fine-tuning (Full FT) serves as the ground-side reference, while every other method restricts the update to a sparse or low-rank subset of parameters, as onboard fine-tuning requires.

4.1 PERFORMANCE UNDER AN ONBOARD PARAMETER BUDGET

Fewer, better-chosen directions outperform more parameters. At $s \leq 0.5$, DRS-LoRA outperforms every onboard-feasible baseline on all 10 transfers (Table 1). With 13.7% of the joint policy

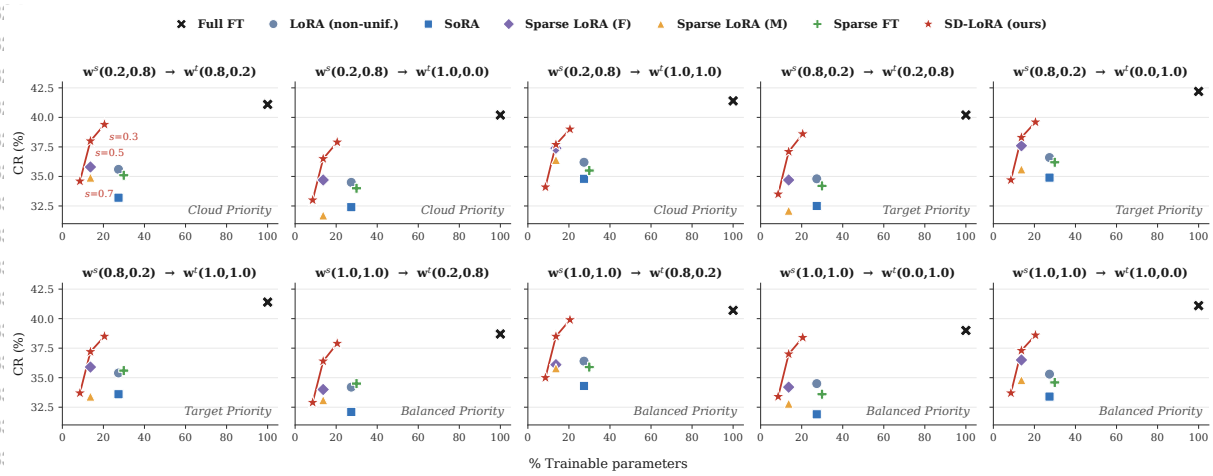

Figure 2: **Completion rate versus trainable parameters.** Each panel is one source→target transfer $\mathbf{w}^s \to \mathbf{w}^t$, with its source class labelled at the lower right. The horizontal axis gives the percentage of the joint four-satellite policy that is trainable, and the vertical axis gives CR (%) on a common scale. Full FT ($\times$) is the ground-side reference at 100%. The onboard-feasible baselines lie between 13.7% and 29.9%: LoRA (non-unif.) $\bullet$, SoRA $\blacksquare$, Sparse LoRA (Fisher) $\blacklozenge$, Sparse LoRA (Mag.) $\blacktriangle$ and Sparse FT $+$. Red stars joined by a line trace DRS-LoRA at s=0.7, 0.5, 0.3 (8.6%, 13.7% and 20.5% of parameters, left to right).

trainable, it retains 92.1% of the Full FT completion rate and improves on LoRA, Sparse FT and SoRA by 5.6%, 7.2% and 12.3% while training half as many parameters as each. These dense adapters allocate capacity by a fixed rank, a fixed mask or a learned gate, none of which reflects how the update changes the policy's behaviour. DRS-LoRA instead spends its budget only on directions that measurably change each satellite's output under the current policy. The same criterion explains a dependence on the source policy that Fisher-ranked Sparse LoRA does not show. From the Balanced source, whose policy already encodes both objectives, the remaining deficit to Full FT is about half that from the skewed sources, because re-weighting can be carried by the few output directions that drift energy ranks first. On the flight processor, accuracy is thus gained by selecting capacity rather than adding it, and the method that selects best also holds the fewest gradients and optimizer states.

The required adaptation is low-dimensional. The trade-off curves in Figure 2 show that performance saturates early. On every transfer, moving from s=0.7 to s=0.5 yields more than twice the gain of moving to s=0.3, although the latter adds 50% more parameters. The update induced by tasking drift therefore lies in a small subspace of each layer's output, the premise on which DRS-LoRA ranks and truncates directions, and this places the onboard operating point at s=0.5. At s=0.7, the shared budget no longer covers the satellite whose drift is least concentrated (§4.3), and methods with 1.6 - 3.5$\times$ the budget recover the lead, although DRS-LoRA still exceeds SoRA with less than a third of its parameters. Within its competitive range, DRS-LoRA is also the least variable method across seeds, Full FT included. This bounds how far a single adaptation run can land from its expected outcome, a requirement onboard, where a degraded policy cannot be rolled back before the next ground contact.

4.2 Recovery from Tasking Drift

Final completion rates conflate how well an adapted policy performs with how much of its drift-induced loss has been repaired. To separate the two, we evaluate recovery on the Balanced source, whose four targets span every form of tasking drift in our setting, namely a shift of emphasis toward either objective and the removal of either objective. Recovery is defined as the fraction of the gap between the zero-shot policy and Full FT that a method closes (Table 2). The zero-shot policy attains 26.0% CR against 39.9% for Full FT, so tasking drift alone costs about a third of the achievable performance while the environment itself is unchanged. DRS-LoRA closes 82.7% of this gap, against

Table 2: Transfer-learning gap recovery for the Balanced Priority ($\mathbf{w}^s = (1.0, 1.0)$) source scenario.

Method	$\mathbf{w}^t = (0.2, 0.8)$		$\mathbf{w}^t = (0.8, 0.2)$		$\mathbf{w}^t = (0.0, 1.0)$		$\mathbf{w}^t = (1.0, 0.0)$		Avg	
	CR	Gap	CR	Gap	CR	Gap	CR	Gap	CR	Gap
Zero-shot	24.2±3.8	0.0%	27.8±4.1	0.0%	25.1±3.6	0.0%	26.9±4.0	0.0%	26.0	0.0%
SoRA (gated)	32.1±4.6	54.5%	34.3±4.9	50.4%	31.9±4.7	48.9%	33.4±4.8	45.8%	32.9	49.9%
Sparse FT (50%)	34.5±5.4	71.0%	35.9±5.5	62.8%	33.6±4.9	61.2%	34.6±5.1	54.2%	34.6	62.3%
Vanilla LoRA	34.9±4.0	74.0%	37.3±4.3	74.0%	35.4±4.4	74.0%	37.4±4.4	74.0%	36.2	73.9%
DRS-LoRA (ours)	**36.4**±3.3	**84.1%**	**38.5**±3.1	**82.9%**	**37.0**±4.5	**85.6%**	**38.0**±4.0	**78.0%**	**37.5**	**82.7%**
Full FT	38.7±4.7	100.0%	40.7±5.4	100.0%	39.0±4.6	100.0%	41.1±5.1	100.0%	39.9	100.0%

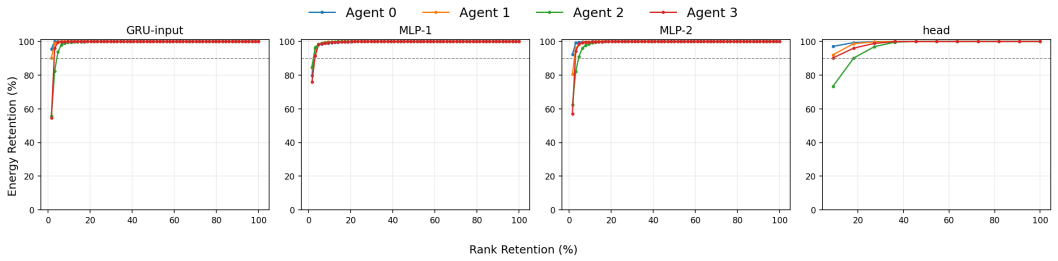

Figure 3: **Output-drift energy retained as rank directions are added**, per satellite and layer. The dashed line marks 90% retention.

73.9% for LoRA, 62.3% for Sparse FT and 49.9% for SoRA. It leads on every target while training half as many parameters as LoRA. However, imaging targets competition rate depends on the type of drift rather than by the magnitude of the initial loss. Although the zero-shot deficit is nearly identical across targets, removing the reward weighting for cloud priority, is the hardest transfer for DRS-LoRA, Sparse FT and SoRA. We attribute this to the structure of the task. Cloud-aware imaging is a coordinated behaviour, in which the lead optical satellite's cloud sensing shapes the payload choices of the others, whereas target priority is resolved largely within each satellite. Dropping the cloud term therefore demands a change in joint behaviour rather than a local re-ranking. LoRA departs from this pattern and recovers almost exactly 74% on every target, consistent with a fixed low-rank subspace that captures the same share of every required update. DRS-LoRA re-selects its retained subspace from the drift observed during adaptation and exceeds that ceiling on all four targets.

4.3 Drift-Energy Conservation Under Dynamic Ranking

The distribution of drift explains why this truncation is inexpensive and how much capacity each satellite requires (Figure 3). In every layer, a few directions carry most of the output drift, so the remaining directions can be removed with little effect on the policy (Lemma 6.1). The degree of concentration, however, differs across satellites. In the recurrent input layer, 1 direction accounts for 90-95% of the drift of 2 agents but only about 55% of that of the other 2. At s=0.5, the shared budget averages 8 of 16 directions per adapter and covers nearly all of the drift of every agent. At s=0.7, the average falls to 5, and the least concentrated agent can no longer receive the rank it needs without taking directions from the others. The performance at this setting therefore reflects the total capacity available rather than the quality of the ranking. It also confirms that satellites differ in how much adaptation capacity they need, which is the premise of allocating a single budget across agents.

4.4 Discussion

Under a fixed onboard budget, the choice of which rank directions to train matters more than their number. At equal budget, ranking by output drift outperforms Fisher and magnitude ranking on every transfer, and pruning half of the LoRA rank by this criterion improves the adapted policy. The recovery experiment explains why. A fixed low-rank subspace recovers the same share of the loss whatever the type of drift, while a subspace re-selected from the observed drift recovers more on

every target. Pruned directions are also deleted rather than masked, so the gradients and optimizer state held by the flight processor shrink with the adapter, a saving that nominal parameter counts do not show. However, this advantage faces limitation when the budget falls below what the least concentrated satellite needs, ranking within each agent cannot compensate, and performance depends on how capacity is divided among agents. For heterogeneous teams, cross-agent allocation is therefore the main open question for onboard adaptation. Our evaluation covers one four-satellite configuration under reward-weight drift, the memory and power savings are inferred from the method's structure rather than measured on the device, and the allocation uses the advantage signal in place of the allocation that minimizes the bound in §6.1.

5 Conclusion

We introduced DRS-LoRA for onboard fine-tuning of heterogeneous multi-agent satellite policies under tasking drift. By ranking rank directions according to the output drift they induce, re-ranking them as the on-policy distribution shifts, and deleting pruned directions together with their optimizer state, it adapts a deployed constellation within a hardware-fixed budget. Across ten transfers, it outperforms every onboard-feasible baseline at $s \leq 0.5$ and recovers most of the loss caused by tasking drift with a small fraction of the policy. Extending cross-agent allocation to high sparsity and evaluating over longer mission horizons are the natural next steps.

References

Joao Vaz Carneiro and Hanspeter Schaub. Scalable architecture for rapid setup and execution of multi-satellite simulations. *Advances in Space Research*, 73(11):5416–5425, 2024.

Minping Chen, Ruijia Yang, and Zeyi Wen. Efficient mask learning for language model fine-tuning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 301–311, 2025.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4133–4145, 2023.

Marawan Gamal Abdel Hameed, Aristides Milios, Siva Reddy, and Guillaume Rabusseau. Rosa: Random subspace adaptation for efficient fine-tuning. *arXiv preprint arXiv:2407.07802*, 2024.

Haoze He, Juncheng Li, Xuan Jiang, and Heather Miller. Smt: Fine-tuning large language models with sparse matrices. In *International Conference on Learning Representations*, volume 2025, pp. 32369–32393, 2025.

Adam Herrmann and Hanspeter Schaub. Reinforcement learning for the agile earth-observing satellite scheduling problem. *IEEE Transactions on Aerospace and Electronic Systems*, 59(5):5235–5247, 2023.

Adam P Herrmann and Hanspeter Schaub. Monte carlo tree search methods for the earth-observing satellite scheduling problem. *Journal of Aerospace Information Systems*, 19(1):70–82, 2022.

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 *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

Longhua Li, Lei Qi, Qi Tian, and Xin Geng. Energy-structured low-rank adaptation for continual learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4OP391X7Qm.

Wei Li, Lujun Li, Mark Lee, and Shengjie Sun. Adaptive layer sparsity for large language models via activation correlation assessment. *Advances in Neural Information Processing Systems*, 37: 109350–109380, 2024.

Yang Lin, Xinyu Ma, Xu Chu, Yujie Jin, Zhibang Yang, Yasha Wang, and Hong Mei. Lora dropout as a sparsity regularizer for overfitting control. *arXiv preprint arXiv:2404.09610*, 2024.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11256–11264. IEEE, 2019.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations*, volume 2026, pp. 59839–59864, 2026.

Mark A Stephenson and Hanspeter Schaub. Bsk-rl: Modular, high-fidelity reinforcement learning environments for spacecraft tasking. In *75th International Astronautical Congress, Milan, Italy, IAF*, pp. 1186–1197, 2024.

Maciej Wołczyk, Bartłomiej Cupiał, Mateusz Ostaszewski, Michał Bortkiewicz, Michał Zajac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem. *arXiv preprint arXiv:2402.02868*, 2024.

Lingling Xu, Haoran Xie, S Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Lu Yin, Gen Li, Meng Fang, Li Shen, Tianjin Huang, Zhangyang Wang, Vlado Menkovski, Xiaolong Ma, Mykola Pechenizkiy, Shiwei Liu, et al. Dynamic sparsity is channel-level sparsity learner. *Advances in Neural Information Processing Systems*, 36:67993–68012, 2023.

Beining Zhang, Aditya Kapoor, and Mingfei Sun. Low-rank agent-specific adaptation (lorasa) for multi-agent policy learning. *arXiv preprint arXiv:2502.05573*, 2025.

Chenyang Zhang. Low-rank matrix approximation techniques for efficient large language model compression. In *Proceedings of the 2026 3rd International Conference on Machine Learning and Intelligent Computing*, pp. 1349–1353, 2026.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3013–3026, 2024.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023a. URL `https://openreview.net/forum?id=lq62uWRJjiY`.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023b.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024.

Ligeng Zhu, Lanxiang Hu, Ji Lin, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. Pockengine: Sparse and efficient fine-tuning in a pocket. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 1381–1394, 2023.

6 APPENDIX

6.1 THEORETICAL ANALYSIS

This section formalizes two properties of Dynamic Ranking that the results rely on. The first is what a reallocation event removes from an adapter. The second is why ranking by output drift preserves more of the adapter's effect than ranking by weight magnitude. We use the notation of §3.2. At an event, the drift of agent i at layer l on the current rollout is $\Delta Y^{i,l} = \Delta W^{i,l} X^{i,l}$ (Eq. 5), with singular value decomposition $\Delta Y^{i,l} = \sum_j \sigma_{i,l,j} u_j v_j^\top$ and $\sigma_{i,l,1} \geq \sigma_{i,l,2} \geq \cdots$. The layer keeps $k = k_{i,l}$ directions (Eq. 6), and $\Delta\tilde{W}^{i,l}$ denotes the update rebuilt by Eq. (8). We write $U_k = [u_1, \ldots, u_k]$ and $\hat{\Sigma}^{i,l} = \frac{1}{n} X^{i,l} (X^{i,l})^\top$ for the second moment of the layer inputs.

[What reallocation removes] The rebuilt update satisfies

$$\Delta\tilde{W}^{i,l} = U_k U_k^\top \Delta W^{i,l}, \tag{15}$$

so the fresh directions added up to rank r_i^+ do not change the layer output at the event. On the current rollout, the output removed by reallocation is

$$\left\|\Delta Y^{i,l} - \Delta\tilde{W}^{i,l} X^{i,l}\right\|_F^2 = \sum_{j>k} \sigma_{i,l,j}^2 \;\leq\; (1-\rho) \sum_j \sigma_{i,l,j}^2 \quad \text{whenever } k = k_{i,l}^\rho, \tag{16}$$

and no update of rank at most k removes less.

Lemma 6.1 shows that reallocation discards exactly the tail of the drift spectrum and nothing else. When the budget does not bind, the loss is at most a fraction $1 - \rho$ of the drift energy. When it binds, $k = r_i^+$ and the loss is the tail beyond r_i^+. The loss is therefore small whenever the drift of a layer is concentrated in its leading directions, which is the low-dimensional structure behind the early saturation in §4.1. The lemma also holds at every event, since the spectrum is recomputed on the current rollout each time.

Proposition 1 (Drift ranking versus magnitude ranking)**.** *Let $\Delta W^{i,l} = \sum_j s_j p_j q_j^\top$ be the singular value decomposition of the weight update, and let $\Delta W_k^{\mathrm{mag}} = \sum_{j \leq k} s_j p_j q_j^\top$ keep its k largest directions. Then*

$$\left\|\Delta Y^{i,l} - \Delta W_k^{\mathrm{mag}} X^{i,l}\right\|_F^2 = n \sum_{j>k} s_j^2 \, q_j^\top \hat{\Sigma}^{i,l} q_j \;\geq\; \sum_{j>k} \sigma_{i,l,j}^2, \tag{17}$$

with equality for every k when $\hat{\Sigma}^{i,l} = cI$ for some $c > 0$.

Each weight direction contributes to the output change in proportion to its squared norm s_j^2 multiplied by the input energy $q_j^\top \hat{\Sigma}^{i,l} q_j$ along it. A direction with a large weight norm that the visited states rarely excite therefore changes the output little, and magnitude ranking may keep it at the expense of a smaller direction that the inputs excite strongly. Drift ranking accounts for both factors, and by Lemma 6.1 it removes the least output of any rank-k choice. The 2 criteria agree only when the layer inputs are isotropic. This is the mechanism behind the ordering of ranking criteria in §4.3.

6.2 PROOFS

Proof of Lemma 6.1. Before the event, $\Delta W^{i,l} = \frac{\alpha}{r_i^-} B^{i,l} A^{i,l}$. Substituting the factors of Eq. (8) gives

$$\Delta\tilde{W}^{i,l} = \frac{\alpha}{r_i^+} \begin{bmatrix} U_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \frac{r_i^+}{r_i^-} U_k^\top B^{i,l} A^{i,l} \\ A_{\mathrm{new}}^{i,l} \end{bmatrix} = \frac{\alpha}{r_i^-} U_k U_k^\top B^{i,l} A^{i,l} = U_k U_k^\top \Delta W^{i,l},$$

where the zero columns of $B^{i,l}$ cancel $A_{\mathrm{new}}^{i,l}$. Hence $\Delta\tilde{W}^{i,l} X^{i,l} = U_k U_k^\top \Delta Y^{i,l} = \sum_{j \leq k} \sigma_{i,l,j} u_j v_j^\top$, and the removed part is $\sum_{j>k} \sigma_{i,l,j} u_j v_j^\top$. Its squared Frobenius norm is $\sum_{j>k} \sigma_{i,l,j}^2$ because the u_j and v_j are orthonormal. If $k = k_{i,l}^\rho$, Eq. (6) gives $\sum_{j \leq k} \sigma_{i,l,j}^2 \geq \rho \sum_j \sigma_{i,l,j}^2$, which is the inequality in Eq. (16). Finally, any update $\Delta W'$ of rank at most k gives $\mathrm{rank}(\Delta W' X^{i,l}) \leq k$, so by the Eckart–Young–Mirsky theorem (Zhang, 2026) $\|\Delta Y^{i,l} - \Delta W' X^{i,l}\|_F^2 \geq \sum_{j>k} \sigma_{i,l,j}^2$. □

Proof of Proposition 1. The removed part is $\sum_{j>k} s_j p_j q_j^\top X^{i,l}$. Since $p_j^\top p_{j'} = 0$ for $j \neq j'$, the cross terms vanish, and

$$\Big\| \sum_{j>k} s_j p_j q_j^\top X^{i,l} \Big\|_F^2 = \sum_{j>k} s_j^2 \, q_j^\top X^{i,l} (X^{i,l})^\top q_j = n \sum_{j>k} s_j^2 \, q_j^\top \hat{\Sigma}^{i,l} q_j .$$

$\Delta W_k^{\mathrm{mag}}$ has rank at most k, so the inequality follows from Lemma 6.1. If $\hat{\Sigma}^{i,l} = cI$, then $\Delta Y^{i,l}(\Delta Y^{i,l})^\top = nc\,\Delta W^{i,l}(\Delta W^{i,l})^\top$. The 2 matrices then share left singular vectors with $\sigma_{i,l,j}^2 = nc\, s_j^2$, so both sides of Eq. (17) equal $nc \sum_{j>k} s_j^2$. □

7 Additional Experimental Details

7.1 Environmental Setup

Orbital and formation configuration. All satellites fly a near-circular Sun-synchronous orbit at 500 km, $i \approx 97.4°$, in a phased along-track formation (Table 3) where Synthetic Aperture Radar and Automatic Identification System (SAR-AIS$_1$) leads with being trailing satellite, along with two optical imaging satellite (OPT$_2$, OPT$_3$) and another long range Synthetic Aperture Radar based SAR$_4$ with

Table 3: Satellite orbital and formation configuration.

Sat.	Payload	Alt. (km)	i (deg)	e	Ω (deg)	ω (deg)	f (deg)
#1	SAR+AIS	500	97.4	0.30×10^{-3}	144.94	36.17	102.91
#2	Optical	500	97.4	0.25×10^{-3}	149.07	26.70	98.45
#3	Optical	500	97.4	0.25×10^{-3}	140.76	26.70	94.98
#4	SAR	500	97.4	0.20×10^{-3}	144.88	17.21	99.45

Satellite physical parameters. All four satellites share the parameters in Table 4, drawn from BSK-RL's default EO configuration (Stephenson & Schaub, 2024); only observation content, action legality, and initialization distributions differ per agent.

Table 4: Default satellite physical parameters.

Parameter	Value	Unit
Battery capacity	400	W·h
Solar panel area / eff.	1 / 20	m^2 / %
Base / instrument power	10 / 30	W (draw)
Data storage capacity	500	GB
Instrument baud rate	500	kbps
Transmitter baud rate	100	Mbps
Max. reaction wheel speed	6000	RPM
Image attitude error	0.1	deg

Scenario and Tasks. The mission objective is to maximize data acquisition over spatially distributed Areas of Interest (AoIs) under limited onboard resources. The AoIs are distributed along the satellite ground track and organized into multiple regions, each subdivided into grid-based targets. Each target is characterized by:

- Target Priority $w_p = q_i \in (0, 1)$, representing mission importance
- Cloud coverage $w_c = \sigma \in [0, 1]$, affecting sensing feasibility
- Time-varying visibility windows

We develop task difference to define source and target by weighting (w_p, w_c) which range from priority-only $(1.0, 0.0)$ to cloud-only $(0.0, 0.1)$. With higher and lower and balance priorities, we determined through target reward regret of acting source preference and normalized by the spread of utilities of target with each candidate set. Moreover, we randomized battery, storage capacity and

target distribution to represent task constraints. To our observation, prioritized weighted sets with random battery level across hetergenous satellite agents impact most, hence we built all sources based on this condition, and targets based all resource value randomization; and only considered task different more than 10%. At each decision step, acting satellites select from the discrete action set:

$$A = \{\text{Image}, \text{Downlink}, \text{Charge}, \text{Desaturate}\} \tag{18}$$

The global reward at time step t integrates mission objectives and resource constraints shown in Eq. (19) The payload selection term is defined $C(\sigma, p)$, where $\mathcal{A} = \{\text{SAR+AIS}_1\ \text{OPT}_2, \text{OPT}_3, \text{SAR}_1\}$ denotes the set of acting agents, which enforces sensor selection consistency with environmental conditions. Optical imaging is favored under clear skies, while SAR is preferred in high cloud coverage scenarios. Failure conditions are defined in Eq.(21), ensuring safe operation under energy and attitude constraints.

$$\begin{aligned} R_t = \lambda_1 \sum_{i \in \mathcal{A}} q_i \mathbb{I}_{\text{cap}} + \lambda_2 \Delta D_t - \lambda_3 \Delta E_t \\ + \lambda_4 C(\sigma, p) - \lambda_5 \mathbb{I}_{\text{failure}} \end{aligned} \tag{19}$$

$$C(\sigma, p) = \begin{cases} 1 - \sigma, & p = \text{OPT} \\ \sigma, & p = \text{SAR} \\ -1, & \text{otherwise} \end{cases} \tag{20}$$

$$\mathbb{I}_{\text{failure}} = \mathbb{I}(b_t < b_{\min} \vee \omega_t > \omega_{\max}) \tag{21}$$

Overall, based on the reward function, the heterogeneous policies are pre-trained based RNN & MLP layer based HAPPO agent and used for deploying onboard via split environemnt where simulation runs on host device and inference as well as finetuning training runs on Jetson Orin Nano. This policy based inference is applied for capturing uniformly distrusted targets, and in the certain given time and number gets, the completion rate will depend.

7.2 Onboard Deployment and Reosurce Cost

We considered Jetson Orin Nano similar to EdgeBRAIN by Infinity Avionics for space processor where the pretrained agent policies uns on the Cortex-A78AE CPU alone, selecting actions in 11.2ms which is 10.4% improvment comparing with Full FT with around 538 MB RAM consumption, which is extrememly lower than training onground. During inference and onboard finetuning, the proposed DRS-LoRA adds around 1.52×10^7 reducing around 33% FLOPS per episodes because the frozen base requires no weight gradients.

7.3 Baseline Implementation Details

All methods below adapt the same frozen HAPPO trained ground based policy under the same source to target scenarios with the same per-agent rollout budget per adaptation window, so that differences in Table 1 are related to the trainable set design rather than to tuning budget

Full FT. Every parameter of the joint policy is trainable under the HAPPO objective directly, with no adapter (Xu et al., 2026). This is the ground-side reference at 100% of the parameters.

Sparse FT. We use a sparse subset of the full weight tensors to make trainable directly, with no low-rank factorization (He et al., 2025; Zhu et al., 2023). We fixed the subset by one-shot magnitude pruning of the pretrained weights before adaptation begins, at fixed 50%. Moreover, gradients for the remaining weights are masked at every step and the optimizer nonetheless holds full-size moment buffers.

LoRA (non-uniform). We use rank-16 LoRA adapter that is placed at every linear layer of every satellite agent's actor where rank is redistributed non-uniformly (Hu et al., 2022). As a result, larger layers receive proportionally more of the fixed rank-16 total budget.

SoRA (gated). Following Ding et al. (2023), each of the 16 rank directions of a LoRA adapter is multiplied by a learned scalar gate g_k, trained jointly with B, A with 50% sparsity.

Sparse LoRA (Fisher / Magnitude). Both variants use DRS-LoRA's own rank-16 adapter placement and its selection step for the cross-agent budget split, so that Table 1 differs from DRS-LoRA within agent ranking signal. *Fisher* ranks direction k of layer l by a diagonal empirical Fisher estimate, $\mathbb{E}\big[(\partial \mathcal{L}^i_{\mathrm{HAPPO}}/\partial [B^{i,l}, A^{i,l}]_k)^2\big]$, accumulated over the same rollout window used for the drift-energy computation. Additionally, *Magnitude* keeps the directions whose weights have changed the most, ranking each rank direction by the norm of its contribution to the weight update as in magnitude-based pruning, and so does not account for the inputs the layer receives on current rollouts.

Use of Large Language Models

We used LLMs to polish our own writing and to help find related work; we read and checked every cited paper ourselves. An LLM helped with intermediate algebraic steps in the proofs of Lemma 6.1 and Proposition 1 (Appendix 6.2). The statements, assumptions and proof approach are our own, and we verified every LLM-assisted step by hand.LLMs were not used for research ideas, method or experiment design, or analysis of results, and we take full responsibility for all content of the paper.