ROBOFL: FEDERATED EXPERT ASSEMBLY FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

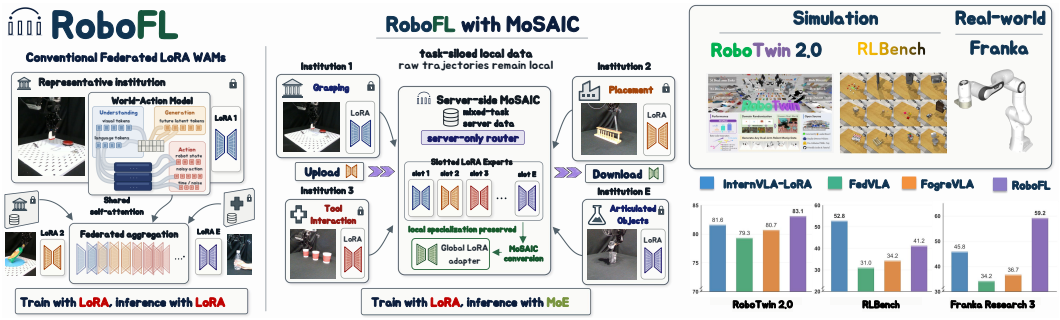

Figure 1: **From federated parameter averaging to prior-informed expert assembly. Left:** Aggregating independently trained LoRA-MoE models can dilute local specialization and introduce inconsistent routing across heterogeneous clients. **Middle:** ROBOFL organizes institutions into task silos and directly installs their task-trained LoRA into the expert branches of a server MoE. **Right:** ROBOFL is evaluated in the RoboTwin 2.0 and RLBench simulator, and on a real-world Franka robot.

ABSTRACT

Vision-language-action and world-action models are increasingly popular, yet remain bottlenecked by physical interaction data that is scarce, institutionally siloed, and task-heterogeneous. A natural federated solution is to let each client adapt a shared foundation model through parameter-efficient fine-tuning, avoiding the exchange of full-model updates. However, federating these adapters is nontrivial, as naive aggregation can entangle incompatible updates, while incorporating MoE-style routing into federated aggregation may dilute specialization and destabilize expert selection. We present ROBOFL, which instantiates MOSAIC (Mixture of Slotted Adapters) for federated world-action learning. MoSAIC directly installs locally trained LoRA adapters as the expert branches of a server MoE. Server-side routers learn token assignments over these prior-informed branches while jointly refining routing and expert parameters. Foresight-to-Action Routing Distillation (FARD) aligns routing across the model's three paths, while Path-Consensus Expert Aggregation (PCEA) converts complete expert updates into a compact global adapter for personalized redistribution. Experiments on RoboTwin 2.0, RLBench, and a real-world Franka robot arm show the superiority of ROBOFL with structured expert assembly, as it outperforms centralized PEFT InternVLA-A1 by 12.23% on the Franka arm, while reducing per-round client communication by up to 86.81% relative to MoE-based federated VLA baselines.

1 INTRODUCTION

Vision-language-action (VLA) models (Brohan et al., 2023; Driess et al., 2023; Ghosh et al., 2024; Kim et al., 2025; Black et al., 2025) and world-action models (WAMs) (Cai et al., 2026; Bi et al., 2026; Ye et al., 2026b) unify perception, language understanding, future prediction, and control in a single policy. Their progress has been driven by broader multimodal and multi-robot training, flow-based action generation, and future-state modeling. However, general-purpose embodied intelligence remains constrained by real interaction data. Unlike web data, robot demonstrations require hardware,

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.

expert teleoperation, repeated resets, safety monitoring, and synchronized multimodal sensing. They are expensive to collect, especially for contact-rich, long-horizon, and multi-embodiment tasks, making high-quality trajectories both scarce and valuable (Wu et al., 2025; Luo et al., 2025b).

This data bottleneck is compounded by privacy and ownership constraints. Robot trajectories can expose private environments, user routines, and proprietary procedures, and their substantial acquisition cost makes them valuable institutional assets that owners are reluctant to share. Federated learning (FL) (McMahan et al., 2017; Kairouz et al., 2021) therefore provides a natural infrastructure for large-scale embodied training, letting institutions contribute complementary experience without centralizing raw data. This task-silo setting is practical rather than artificial: an institution typically owns the hardware, expertise, and suite of tasks for a focused family of manipulation tasks, so disjoint silos arise directly from existing practice. The resulting federation is nevertheless highly heterogeneous: client task families are largely disjoint and uneven in size, so each client optimizes over a different label distribution, and the aggregate problem is strongly non-IID. Under such task-level skew, local optima drift apart, and naive averaging blends transformations that are valid only within their own families. **The objective is thus to combine complementary task-specific capabilities into a broadly competent policy without exposing or erasing their source.**

Mixture-of-Experts (MoE) (Jacobs et al., 1991; Shazeer et al., 2017; Fedus et al., 2022) provides a natural representation for such task diversity, combining specialized experts with a shared backbone and input-dependent routing. For example, FedVLA (Miao et al., 2025) trains dual-gating MoE models on clients and aggregates their aligned-MoE trunk parameters for performance enhancement. Yet a modular architecture does not, by itself, ensure coherent federated learning. When each client trains its own MoE on a narrow task distribution, the corresponding expert branches can acquire distinct specializations, while the routers learn assignments shaped by local task frequencies. Averaging these expert updates can dilute task-specific knowledge, and aggregating the routers can disrupt the assignments that make that knowledge useful, leading to performance degradation as shown in Section 4. **The key is therefore to separate local specialization from the learning of a shared routing policy.**

To address this gap, we propose RoboFL, a federated framework that combines a task-silo protocol with server-side expert assembly as shown in Figure 1. Designed for communication-constrained FL settings, RoboFL leverages parameter-efficient fine-tuning (PEFT) (Hu et al., 2022; Wu et al., 2024) to enable clients to exchange compact LoRA adaptations rather than full model updates (Miao et al., 2025; Zhou et al., 2026). Specifically, ❶ each institution trains a single LoRA adapter on a focused family of tasks and uploads only its adaptation parameters to the server. ❷ The server assigns the uploaded adapters to fixed expert slots and composes them into a layer-wise MoE, where a router is trained to select among task-specialized LoRA experts. ❸ After server-side routing and expert refinement, the resulting expert updates are converted into a compact global adapter and blended with each client-associated expert before the next communication round.

This procedure yields a **Mixture-of-Slotted-Adapters** with integration and conversion (MoSAIC). By separating local expert formation from global router learning, RoboFL enables the router to build on task-trained capabilities rather than on initially unspecialized branches. The fixed client-to-slot mapping preserves parameter correspondence across rounds, while still allowing flexible token-to-expert assignments. This design supports more stable routing without imposing multi-expert training or communication on clients: local training and exchanged adapters retain the footprint of single-adapter fine-tuning, while routing and multi-expert optimization remain on the server.

In addition, contemporary WAMs (Cai et al., 2026; Bi et al., 2026; Ye et al., 2026b) use aligned mixture-of-transformers (MoT) architectures (Liang et al., 2024) whose understanding, visual foresight, and action paths provide complementary evidence about the same interaction. Such a three-path structure of WAMs offers an additional source of guidance for this assembly process. MoSAIC exploits this structure to inform both expert selection and adapter conversion. Therefore, we propose **Foresight-to-Action Routing Distillation** (FARD), which distills reliable routing consensus between the understanding and generation paths into the action router, using a detached teacher to guide expert selection for action generation. In addition, **Path-Consensus Expert Aggregation** (PCEA) uses detached consensus across all three paths to weight complete server-refined LoRA updates and reconstruct a rank-constrained global adapter. Together, these mechanisms use agreement across what the model understands, predicts, and executes to guide which experts it activates during server training and how it consolidates their knowledge for subsequent client training.

Finally, we evaluate ROBOFL on RoboTwin 2.0 (Mu et al., 2025), RLBench (James et al., 2020), and a real-world Franka robot arm against centralized PEFT WAMs, as well as FL-based VLA methods, as shown on the right side of Figure 1. Experiments across simulation and the real world show that ROBOFL outperforms existing federated VLA methods and, in certain scenarios, even centralized PEFT WAMs, as it attains 83.12% overall on RoboTwin 2.0, 4.2% above federated averaging and 1.52% above centralized InternVLA-A1, reaches 59.17% against 46.94% for centralized InternVLA on six real-world Franka tasks, and cuts client communication by up to 86.81% against MoE-based federated VLA baselines.

Our contributions are summarized as follows.

1. **Task-siloed federated expert assembly.** We propose ROBOFL with MOSAIC to assemble client-trained LoRA adapters into a server-routed MoE, keeping raw trajectories local and retaining single-adapter client training and communication.
2. **Foresight-guided action routing.** We introduce FARD to distill reliable understanding-generation routing consensus into the action router for MoT-based world action models.
3. **Path-consensus adapter conversion.** We introduce PCEA to aggregate complete server-refined expert updates using three-path consensus and reconstruct a rank-constrained global adapter for personalized redistribution.

2 RELATED WORKS

Vision-language-action and world-action models. VLA models connect multimodal knowledge to robot control, from RT-2 (Brohan et al., 2023) and PaLM-E (Driess et al., 2023) to multi-robot policies such as Octo (Ghosh et al., 2024) and OpenVLA (Kim et al., 2025). Subsequent work advances flow-based control (Black et al., 2025), dual-system architectures (Bjorck et al., 2025), action tokenization (Pertsch et al., 2025), spatial representations (Qu et al., 2025), and efficient adaptation (Zhang et al., 2026). Cosmos (Agarwal et al., 2025) and recent WAMs (Ye et al., 2026a; Li et al., 2026) further exploit future-state prediction. These systems generally assume centralized training rather than assembling future-aware policies from institutionally distributed task data.

VLA and WAMs with MoE and FL. MoE models (Fedus et al., 2022; Zhang et al., 2025) support specialized computation, including expert selection and weighting decoupled in AdaMoE (Shen et al., 2025), force-aware routing in ForceVLA (Yu et al., 2025), and layer-wise activation in MoLe-VLA (Zhang et al., 2026). FLAME (Betran et al., 2025) benchmarks robotic FL, while FedVLA (Miao et al., 2025) combines federated training with dual-gating MoE and ForgeVLA (Zhou et al., 2026) addresses unannotated vision-action logs through instruction recovery, contrastive planning, and adaptive aggregation. Training MoE routers on narrow client distributions can nevertheless produce inconsistent expert assignments. ROBOFL separates local adapter specialization from server-side routing and jointly refines uploaded experts and module-specific routers on server data.

Federated PEFT and MoE. LoRA (Hu et al., 2022) provides compact low-rank updates for federated adaptation. FLoRA (Wang et al., 2024), FRLoRA (Yan et al., 2025), FedSA-LoRA (Guo et al., 2025), and LoRA-A2 (Koo et al., 2025) study low-rank aggregation and sharing, while FedFisher (Jhunjhunwala et al., 2024) addresses one-shot aggregation. Mixture of LoRA Experts routes among adapters (Wu et al., 2024), FedMoE (Mei et al., 2024) integrates client-specific sub-MoEs, and pFedMoAP (Luo et al., 2025a) exchanges prompt experts for personalization. In contrast, ROBOFL installs single-adapter client updates as server MoE experts without federating selection routers, then uses cross-path consensus to convert complete expert updates into compact adapters for personalized redistribution.

3 METHOD

3.1 PRELIMINARY

Three-Path World-Action Architecture. ROBOFL is built on a three-path world-action architecture, in which the understanding (U), generation (G), and action (A) paths are coupled through unified

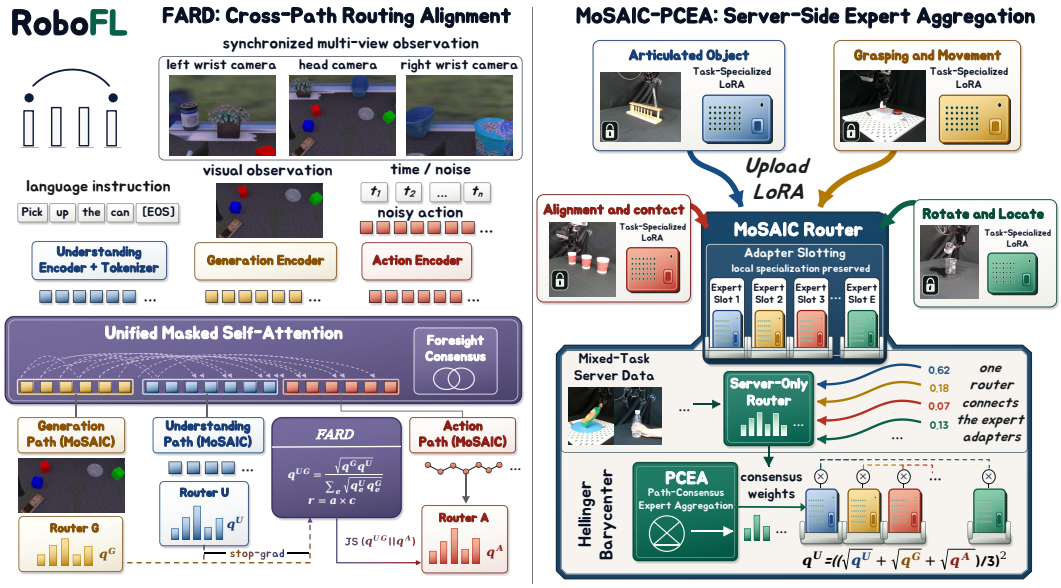

Figure 2: **Overview of RoboFL. Left:** Modality-specific tokens interact through unified masked self-attention, while FARD distills detached understanding-generation consensus into action routing. **Right:** MoSAIC installs task-trained LoRA adapters as server MoE experts and jointly refines routers and experts. PCEA aggregates complete expert updates using three-path consensus to form a rank-constrained global adapter, which is blended with each server-refined client expert.

masked self-attention. U encodes current images and instructions with Qwen3-VL. G processes Cosmos-tokenized past and current frames to predict future-frame latents. A receives a projected state token, followed by noisy action tokens fused with a time embedding. At each aligned layer, path-specific queries, keys, and values are concatenated along the token dimension for joint attention, then split for path-specific output projections and MLPs. Thus, the paths share the attention computation, not projection or MLP weights. The block-causal mask orders tokens as $U \mid G \mid$ state $\mid$ actions: each block attends to itself and preceding blocks, subject to padding masks. Action tokens attend bidirectionally within a chunk and use U/G hidden context, not decoded future images.

At matching layer/projection m, let $q^P_{m,b} \in \mathbb{R}^E$ be the sample-wise mean of the full *pre-top-k* router probabilities for path $P \in \{U, G, A\}$. Specifically, pooling masks out invalid observations and language positions in the understanding path U and invalid visual positions in the generation path G. In A, it includes all action positions, excluding the state token but not action padding. Cross-path comparisons use samples with valid pooled distributions in all three paths.

Flow-Matching Action Policy. For a demonstrated action chunk $a \in \mathbb{R}^{H \times d_a}$, noise $\epsilon \sim \mathcal{N}(0, I)$, and time $t \sim p_{\text{time}}$, the action path predicts the conditional velocity

$$\begin{aligned} x_t &= (1-t)a + t\epsilon, \qquad u_t = \epsilon - a, \\ \mathcal{L}_{\text{action}} &= \mathbb{E}_{a,\epsilon,t}\left[\text{MSE}_{\text{valid}}(v_\theta(x_t, t, c), u_t)\right], \end{aligned} \tag{1}$$

where c denotes U/G context and the robot state, and MSE averages over actual action dimensions and non-padded steps when padding labels are available. At inference, explicit Euler integration proceeds from $x_1 \sim \mathcal{N}(0, I)$ at $t = 1$ to $t = 0$. The generation loss $\mathcal{L}_{\text{gen}}$ is the MSE between predicted and target future Cosmos latents over valid views.

3.2 MoSAIC: Mixture of Slotted Adapters

As shown in Figure 2, each of the K clients trains a standard LoRA adapter for each layer on its private task-family dataset D_k. The server maintains $E = K > 1$ complete expert branches and one selection router per adapted projection. A *slot* identifies a client-associated LoRA adapter within a module, not a separate model.

Adapter Slotting. In round τ, client k uploads its adapter factors, which directly overwrite slot k at each adapted module m:

$$(A^{\tau,0}_{m,k}, B^{\tau,0}_{m,k}) \leftarrow (A^{\mathrm{loc},\tau}_{m,k}, B^{\mathrm{loc},\tau}_{m,k}). \tag{2}$$

No averaging occurs during installation. Client-to-slot identities remain fixed without restricting token assignments, and expert factors are refreshed each round, while selection routers persist only on the server. The protocol exchanges parameters, not raw trajectories.

Routed Expert Integration. For token representation z, the adapted projection is, omitting bias,

$$h_m(z) = W^0_m z + \gamma_m \sum_{e=1}^{E} \pi_{m,e}(z) B_{m,e} A_{m,e} z, \qquad \gamma_m = \alpha_m / r, \tag{3}$$

where W^0_m is frozen and π_m contains renormalized top-k_{route} weights, distinct from the dense probabilities pooled into q^P_m. The server jointly refines selection routers and both expert factors on mixed-task server data, starting from task-trained adapters. Path indices on weights are suppressed while each path retains its own parameters. Uploaded shared action-head weights are uniformly averaged across clients, refined at the server, and copied back to all clients.

Global Adapter Conversion. Let $\Delta W^{\mathrm{srv},\tau}_{m,e} = \gamma_m B^{\mathrm{srv},\tau}_{m,e} A^{\mathrm{srv},\tau}_{m,e}$ denote the complete server-refined adapter residual relative to the frozen backbone. PCEA converts these updates into a rank-constrained global adapter using detached routing evidence, as detailed below. This static conversion prepares client parameters rather than guaranteeing equivalence to the input-dependent MoE.

Personalized Adapter Redistribution. Client k receives standard LoRA factors representing

$$\Delta W^{\mathrm{init},\tau+1}_{m,k} = \Pi_r\left(\tfrac{1}{2}\Delta W^{\mathrm{srv},\tau}_{m,k} + \tfrac{1}{2}\Delta W^{\mathrm{glob},\tau}_{m}\right), \tag{4}$$

where Π_r is a truncated SVD approximation of rank at most r. The client-specific component is the *post-server expert*, not the original upload, which enables personalized parameter aggregation.

3.3 FARD: Foresight-to-Action Routing Distillation

FARD uses understanding-generation agreement to guide action routing. For each valid module-sample pair (m, b), define the geometric-mean teacher and its reliability:

$$\begin{aligned} a_{m,b} &= \sum_e \sqrt{q^U_{m,b,e} q^G_{m,b,e}}, \qquad q^{UG}_{m,b} = \frac{\sqrt{q^U_{m,b} q^G_{m,b}}}{a_{m,b}}, \\ r_{m,b} &= a_{m,b}\left[1 - \frac{H(q^{UG}_{m,b})}{\log E}\right]_+ . \end{aligned} \tag{5}$$

Thus, reliability combines path agreement with teacher concentration, so that unreliable routing signals are down-weighted during training. For valid samples $\mathcal{V}_m$, the loss is

$$\mathcal{L}_{\mathrm{FARD}} = \mathrm{mean}_m \left[\frac{1}{|\mathcal{V}_m|} \sum_{b \in \mathcal{V}_m} \mathrm{sg}(r_{m,b}) D_{\mathrm{JS}}\big(\mathrm{sg}(q^{UG}_{m,b}), q^A_{m,b}\big)\right], \tag{6}$$

averaged over modules with nonempty $\mathcal{V}_m$ (zero if none). Here sg stops gradients while entropy and JS use natural logarithms, with numerical probability floors omitted for clarity. Gradients pass through the action-distribution argument, not the teacher or reliability, and upstream U/G parameters can still receive indirect gradients through joint attention. The overall server objective therefore minimizes

$$\mathcal{L}_{\mathrm{MoSAIC}} = \mathcal{L}_{\mathrm{action}} + \lambda_{\mathrm{gen}}\mathcal{L}_{\mathrm{gen}} + \lambda_{\mathrm{aux}}\mathcal{L}_{\mathrm{aux}} + \lambda^{(\tau)}_{\mathrm{FARD}}\mathcal{L}_{\mathrm{FARD}}, \tag{7}$$

where $\mathcal{L}_{\mathrm{aux}}$ denotes the module-level auxiliary load-balancing loss averaged over the E experts of each module, and $\lambda^{(\tau)}_{\mathrm{FARD}}$ follows the round-wise warmup schedule.

3.4 PCEA: PATH-CONSENSUS EXPERT AGGREGATION

PCEA weights complete expert updates rather than averaging their factors, since, in general,

$$\sum_e \omega_e B_e A_e \neq \left(\sum_e \omega_e B_e\right)\left(\sum_e \omega_e A_e\right). \tag{8}$$

During server training, it collects detached, unnormalized Hellinger-barycenter evidence:

$$g_{m,b,e} = \left(\frac{\sqrt{q^U_{m,b,e}} + \sqrt{q^G_{m,b,e}} + \sqrt{q^A_{m,b,e}}}{3}\right)^2. \tag{9}$$

Summing the valid sample occurrences across server steps and ranks within a round gives:

$$s_{m,e} = \sum_b g_{m,b,e}, \qquad w_{m,e} = \frac{s_{m,e}}{\sum_j s_{m,j}}, \qquad \bar{a}_m = \frac{\sum_e s_{m,e}}{N_m}, \tag{10}$$

where N_m counts sample occurrences. The evidence retains its agreement mass rather than being normalized per sample. For observed module triplets $\mathcal{M}$, PCEA computes agreement-weighted global weights and a Hellinger midpoint:

$$\bar{w}_e = \frac{\sum_{m\in\mathcal{M}} \bar{a}_m w_{m,e}}{\sum_{m\in\mathcal{M}} \bar{a}_m}, \qquad \widetilde{w}_{m,e} = \frac{(\sqrt{w_{m,e}} + \sqrt{\bar{w}_e})^2}{\sum_j (\sqrt{w_{m,j}} + \sqrt{\bar{w}_j})^2}. \tag{11}$$

Matched $U/G/A$ modules share these weights. Eligible unobserved modules use $\bar{w}$, and PCEA requires nonempty evidence. The unmatched state-projection adapter is aggregated uniformly.

For each physical module's own post-server factors, conversion gives:

$$\Delta W^{\mathrm{glob},\tau}_m = \Pi_r\left(\sum_e \widetilde{w}_{m,e} \Delta W^{\mathrm{srv},\tau}_{m,e}\right). \tag{12}$$

Implementation uses QR reduction on the stacked weighted factors, followed by a small SVD. FARD guides routing during server training, while PCEA consolidates the updates for client training.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Evaluation Protocol. We evaluate ROBOFL under the task-silo protocol on RoboTwin 2.0 (Mu et al., 2025), RLBench (James et al., 2020) and Franka robots[1]. We partition the 50 tasks into 8 client task-family groups and 1 server residual group for RoboTwin 2.0 `random`, and split the 8 RLBench tasks into 4 client partitions and 1 server partition, with 6 tasks distributed across the four clients and 2 tasks retained on the server. As for real-world applications, we partition 6 manipulation tasks into 4 clients and 1 server for Franka, creating an extreme `non-IID data scenario`.

Federated Optimization. All experiments initialize from the pretrained *InternVLA-A1-3B* checkpoint, use full client participation, 100 communication rounds, and one local epoch. Client and server steps are $500/200/300$ (RoboTwin / RLBench / Franka); client batch size is 32, and server batch size is $8/16/16$. Both optimizers are AdamW with learning rate 1×10^{-4}, betas $(0.9, 0.95)$, epsilon 10^{-8}, weight decay 0.01, and gradient-norm clipping at 1.0, using cosine decay from 1×10^{-4} to 1×10^{-5} over $50{,}000/20{,}000/30{,}000$ decay steps and $2{,}500/400/600$ warmup steps.

MoSAIC Configuration. MoSAIC uses eight expert branches per adapted module with top-4 server-side routing for RoboTwin 2.0, and four expert branches per adapted module with top-2 server-side routing for RLBench and the real-world applications. Each expert uses rank $r = 16$, LoRA parameter $\alpha = 32$ (effective scaling $\alpha/r = 2$), and dropout 0.05. The generation and load-balancing coefficients are $\lambda_{\mathrm{gen}} = 0.01$ and $\lambda_{\mathrm{aux}} = 0.001$. FARD uses $\lambda_{\mathrm{FARD}} = 0.01$ with a 10-round warmup.

[1]Full task settings and evaluation results are shown in the Appendix.

Table 1: RoboTwin 2.0 success rates on a 25-task challenging subset under random condition (selected as tasks where FedAvg $< 80\%$). Overall reports performance on all 50 tasks.

Method	beat ham.	rank size	hand blo.	hang mug	lift pot	move can	move sta.	pick div.	pick dual	place L	place R	bread ski.	car basket
CENTRALIZED TRAINING													
InternVLA	73	78	*62*	27	32	76	**53**	68	74	*85*	**84**	*79*	60
Motus	**82**	*86*	60	20	**40**	*86*	48	*70*	80	*85*	80	**80**	69
FEDERATED LEARNING(LoRA/MoE)													
FedAvg	75	74	52	24	33	67	39	69	77	79	79	74	72
FedMoE	69	71	42	21	*34*	57	34	61	70	84	80	66	64
ForgeVLA	75	83	57	**35**	**40**	83	44	67	*82*	**87**	82	78	68
FedVLA	73	85	61	26	33	85	45	62	**84**	80	82	77	*74*
RoboFL	*78*	**92**	**73**	*30*	24	**87**	*50*	**72**	**84**	**87**	*83*	**80**	**82**

Method	dual shoe	mouse pad	place bas.	place sca.	press sta.	bottles dus.	put cab.	rotate QR	scan obj	stack bow.	stamp seal	turn swi.	**Overall**
CENTRALIZED TRAINING													
InternVLA	64	**70**	**82**	**81**	74	**90**	**68**	77	64	**86**	*68*	43	*81.60*
Motus	*73*	*64*	79	73	72	*89*	*67*	**87**	65	77	66	45	80.96
FEDERATED LEARNING(LoRA/MoE)													
FedAvg	67	56	79	75	*75*	77	**68**	79	58	79	57	48	78.92
FedMoE	60	54	75	69	74	79	62	73	31	78	60	38	75.12
ForgeVLA	70	59	*80*	*77*	**78**	83	61	77	69	75	59	*52*	80.70
FedVLA	71	53	70	75	*75*	84	60	75	*70*	*80*	65	51	79.32
RoboFL	**74**	58	72	74	74	**90**	**68**	*81*	**74**	**86**	**70**	**53**	**83.12**

Baselines. We compare against **InternVLA-A1** (Cai et al., 2026) and **Motus** (Bi et al., 2026), which are centralized PEFT references; **FedAvg** (McMahan et al., 2017) performs standard weight averaging; **FedMoE** (Mei et al., 2024) uses client-specific expert sub-MoEs; **FedVLA** (Miao et al., 2025) combines instruction-oriented scene parsing, dual-gating MoE, and expert-driven aggregation; and **ForgeVLA** (Zhou et al., 2026) targets heterogeneous federated robot data with weak or missing language supervision. We adapt them to the same FL-PEFT with Kaiming-random-initialized adapter.

4.2 Quantitative Results for Simulation

Model Performance on RoboTwin 2.0. Table 1 reports the 25 RoboTwin 2.0 tasks on which FedAvg stays below 80%, since the remaining tasks are near-saturated and thus uninformative. RoboFL attains the highest overall success rate of 83.12%, exceeding FedAvg by 4.20% and also surpassing centralized PEFT with InternVLA-A1 and Motus. This pattern matches MoSAIC's design: retaining locally specialized adapters as expert branches and learning their input-dependent composition rather than immediately averaging them. Best or joint-best results on 15 displayed tasks show that the gains extend across tasks. Overall, RoboFL delivers the strongest performance in terms of single-adapter client cost, demonstrating that server-side expert assembly can outperform both federated averaging and centralized PEFT across heterogeneous task silos.

Model Performance on RLBench. We further examine how RoboFLperforms on RLBench with only eight tasks under a 4-client, 4-expert MoSAIC in Table 2. RoboFL remains the best FL method at 41.25%, yet the margin over centralized PEFT reverses relative to RoboTwin, where RoboFL also beats InternVLA-A1 and Motus. With fewer tasks and clients, each expert sees less complementary evidence for routed composition, so the advantage of structured expert assembly is smaller. These results suggest that RoboFL is most effective in large-scale, task-heterogeneous regimes rather than in small, low-data suites.

Table 2: Average success rates (%) for eight RLBench tasks with 4-expert MoSAIC for RoboFL.

Method	InternVLA	Motus	ForgeVLA	FedVLA	RoboFL
Type	Central-LoRA		FL-LoRA	FL-MoE	LoRA→MoE
Success	52.75	52.50	34.25	31.00	41.25

Client Resources and Communication. Table 3 compares adaptation resources at the client boundary, combining upload and download into a per-round communication total. With one rank-16 adapter per client, RoboFL matches FedAvg's 40.85M trainable parameters and 311.63 MiB payload, while retaining eight server-side expert branches. Relative to MoE-based FL methods FedMoE and FedVLA, client parameters decrease by up to 86.95% and per-round communication by up to 86.81%, while RoboTwin 2.0's success rates increase by 8 and 3.8 percentage points, respectively. Peak allocated GPU memory falls from 15.59 to 8.33 GiB in the standardized client-architecture microbenchmark. These results demonstrate that RoboFL achieves the best accuracy-efficiency trade-off: it attains the highest success rate with the same computational and communication overhead as single-adapter baselines, while remaining far below client-side MoE.

Table 3: Client resources and logical communication on RoboTwin 2.0. Communication combines upload and download per participating client per round.

Method	Adapters c / s	Params (M)	Comm. MiB	Mem. GiB	Success (%)
FedAvg	1 / 1	40.85	311.63	8.33	78.92
FedMoE	4 / 8	158.11	1,206.28	11.44	75.12
ForgeVLA	1 / 1	40.85	312.06	8.33	80.70
FedVLA	8 / 8	312.93	2,362.85	15.59	79.32
RoboFL	**1 / 8**	**40.85**	**311.63**	**8.33**	**83.12**

4.3 Ablation Studies

Influence of FARD and PCEA. We present a sequential ablation of FARD and PCEA, using vanilla MoSAIC as the baseline and centralized LoRA as an external reference under its own training schedule with RoboTwin 2.0, as shown in Figure 3. We evaluate PCEA on top of FARD because its consensus-weighted conversion is designed to exploit the routing agreement. FARD alone increases the number of tasks exceeding 80% success from 31 to 33, but reduces coverage at $\geq 90\%$ from 19 to 17. Adding PCEA raises these counts to 34 and 24, surpassing both vanilla MoSAIC and centralized LoRA at the latter threshold. Tasks with 100/100 successes likewise increase from 4 for both references to 6 with FARD and 9 with FARD+PCEA. At $\geq 95\%$, however, both variants cover 14 tasks versus 15 for vanilla MoSAIC. These results support combining routing distillation with expert conversion, potentially preserving coordinated task knowledge across federated rounds and supporting more consistent execution and broader coverage of high-success tasks.

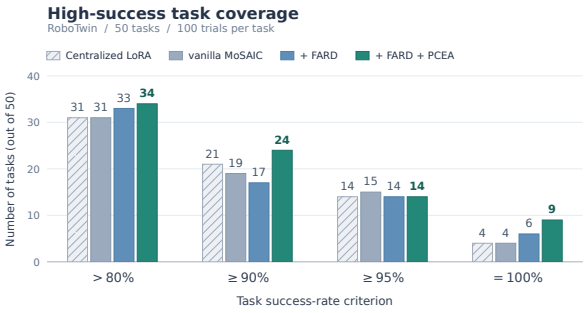

Figure 3: Success-threshold coverage on RoboTwin. Bars count tasks satisfying each success-rate criterion across 50 tasks with 100 trials per task.

Routing Coupling and Intervention. We probe route-to-action coupling across 50 RoboTwin 2.0 tasks while holding the initial observation and robot state fixed. Instruction substitution changes only the language condition, whereas uniformized and scrambled controls intervene only on the action-router distribution. As shown in Figure 4, instruction swaps produce strong pooled route-to-action coupling, establishing that task semantics reach action generation through routing. The direct intervention isolates the action router more strictly. Scrambling produces a larger routing displacement for FARD, yet a smaller action displacement than vanilla RoboFL. Thus, FARD makes routing more selective without making the resulting policy brittle. Together, the probes support FARD as a cross-path routing-alignment mechanism, while PCEA performs the corresponding path-consistent expert conversion.

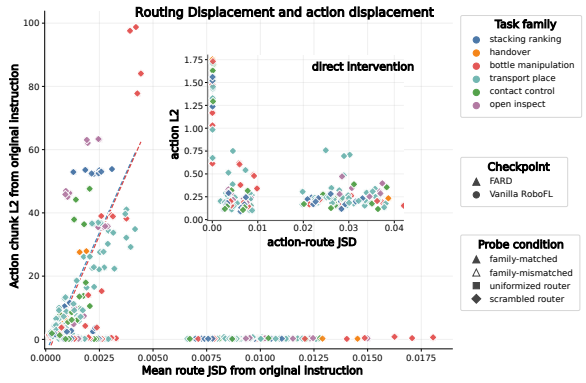

Figure 4: Route-to-action coupling under instruction swaps and direct action-router interventions. The inset magnifies the direct intervention regime.

Table 4: Real-world applications with six single-arm Franka tasks SR (%) with three replicates.

Method	Type	adjust bottle	stamp seal	stack cups	screw bottle	pour water	test tube	Mean ± Avg. SD
InternVLA	Centralized-LoRA	63.33	53.33	41.67	31.67	48.33	43.33	46.94 ± 7.80
ForgeVLA	FL-LoRA	41.67	31.67	46.67	28.33	31.67	33.33	35.56 ± 7.03
FedVLA	FL-MoE	43.33	41.67	33.33	13.33	40.00	30.00	33.61 ± 5.60
RoboFL	FL-LoRA→MoE	**68.33**	**66.67**	**68.33**	**36.67**	**56.67**	**58.33**	**59.17 ± 7.61**

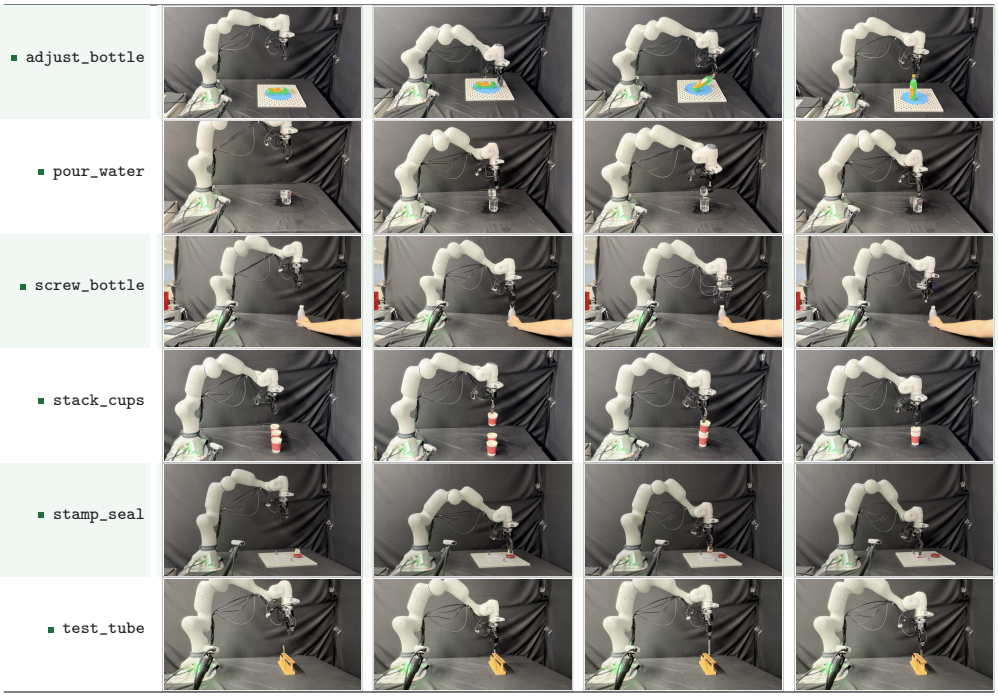

Figure 5: Qualitative successful real-world Franka rollouts for each of six single-arm Franka tasks.

4.4 Real-world Applications with Franka

Quantitative Analysis. Table 4 reports success rates on six single-arm Franka tasks. We report mean success rates over three evaluation rounds of 20 trials per task. RoboFL attains the highest Overall success rate of 59.17%, exceeding centralized InternVLA by 12.23% and federated ForgeVLA and FedVLA by 23.61% and 25.56%. MoSAIC therefore extends beyond simulation: installing each client's single LoRA as a routed server expert, rather than averaging updates, preserves input-dependent specialization under real perception, contact, and hardware variability, even surpassing pooled centralized PEFT at the cost of a single adapter per client.

Qualitative Analysis. Figure 5 shows successful rollouts on six Franka tasks. Execution follows the simulation's three-phase structure: approach, sustained contact, and sequenced state change. This closed-loop coherence reflects the method's design: task-trained experts remain distinct under server routing, while understanding-generation consensus guides action routing, so contact-rich behaviors remain input-dependent rather than averaging into generic motions. For example, *adjust bottle* task seats rather than strikes, *screw bottle* task sustains rotation through the cap.

5 Conclusion

We presented RoboFL, a federated framework for learning world-action models from task-siloed robot data without centralizing raw trajectories. MoSAIC installs locally trained LoRA adapters as server MoE experts and jointly refines routing and expert parameters. Specifically, FARD aligns action routing with understanding-generation consensus and PCEA consolidates expert updates into a rank-constrained global adapter for personalized redistribution. Across simulation and real-robot experiments, their combination delivers the highest overall success rates among the evaluated federated methods while keeping client training at single-adapter cost.

AI USE STATEMENT

The authors used AI-based writing and coding assistants to help draft and polish portions of the text and to support the implementation and analysis scripts. All problem formulation, method design, experimental execution, data collection, and interpretation of results were carried out and verified by the authors, who take full responsibility for the content of this paper.

ETHICS STATEMENT

This work studies federated learning of world-action models for robotic manipulation. Experiments in simulation use the publicly released RoboTwin 2.0 and RLBench benchmarks, and the real-robot evaluations collect no personally identifiable information. The motivating concern of the paper is that robot trajectories can reveal private environments, user routines, or proprietary procedures, and that centralized pooling of such data may be undesirable. The proposed protocol keeps raw trajectories at each institution and exchanges only adaptation parameters, thereby reducing this exposure; however, we do not claim formal privacy guarantees, such as differential privacy or secure aggregation, and parameter updates may still leak information in adversarial settings. We therefore position the method as an architectural alternative to raw-data centralization rather than as a complete privacy solution. We also note that broad manipulation policies can be misused, and we follow the ICLR Code of Ethics in conducting and reporting this work.

REPRODUCIBILITY STATEMENT

We summarize the resources provided to support reproducibility. The full set of training and evaluation settings appears in Table 5 and Appendix A; the task-silo assignments for the eight RoboTwin 2.0 clients and the four RLBench clients are listed in Tables 6 and 8; and the observation, action, and optimization interfaces are specified in the same appendix. The client resources, logical communication accounting, and the architecture profiles behind the memory measurements are documented in Appendix A.1. The MoSAIC, FARD, and PCEA formulations, including the adapter slotting, router integration, personalized redistribution, and consensus-weighting rules, are given in the Method section and implemented as described in Appendix A.1. We plan to release the source code, launch configurations, and evaluation scripts as supplementary materials; these will include data preprocessing and normalization steps, evaluation packages for both benchmarks, and scripts used to generate the resource and ablation figures. Simulation results use a single fixed evaluation seed; real-robot results use a fixed scripted protocol with 20 trials per task. Differences below a few points, and per-task swings on the 8-task RLBench suite, should be read as descriptive rather than statistically established.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Santiago Bou Betran, Alberta Longhini, Miguel Vasco, Yuchong Zhang, and Danica Kragic. Flame: A federated learning benchmark for robotic manipulation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2494–2500. IEEE, 2025.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi0: A vision-language-action flow

model for general robot control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, California, USA, 2025. doi: 10.15607/RSS.2025.XXI.010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026.

Hong-You Chen and Wei-Lun Chao. FedBE: Making bayesian model ensemble applicable to federated learning. In *International Conference on Learning Representations*, 2021.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 8469–8488. PMLR, 2023.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, 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*, Delft, Netherlands, 2024.

Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. Selective aggregation for low-rank adaptation in federated learning. In *International Conference on Learning Representations*, 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedfisher: Leveraging fisher information for one-shot federated learning. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238, pp. 1612–1620. PMLR, 2024.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2): 1–210, 2021.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Grace Lam, Pannag R. Sanketi, et al. Openvla: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, pp. 2679–2713. PMLR, 2025.

Jabin Koo, Minwoo Jang, and Jungseul Ok. Towards robust and efficient federated low-rank adaptation with heterogeneous clients. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 416–429, 2025.

Zhe Li, Zhenzhe Zhang, Yangyang Wei, Wenjie Zhang, Xichen Yuan, Peiyuan Zhi, Gen Li, Xinying Guo, Fengjie Gao, Jianfei Yang, et al. omega-0: A latent predictive world action model for concurrent humanoid loco-manipulation. *arXiv preprint arXiv:2608.06375*, 2026.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2351–2363, 2020.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

Jun Luo, Chen Chen, and Shandong Wu. Mixture of experts made personalized: Federated prompt learning for vision-language models. In *International Conference on Learning Representations*, 2025a.

Yulin Luo, Chun-Kai Fan, Menghang Dong, Jiayu Shi, Xiangju Mi, Mengdi Zhao, Bo-Wen Zhang, Cheng Chi, Jiaming Liu, Gaole Dai, et al. Robobench: A comprehensive evaluation benchmark for multimodal large language models as embodied brain. *arXiv preprint arXiv:2510.17801*, 2025b.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Hanzi Mei, Dongqi Cai, Ao Zhou, Shangguang Wang, and Mengwei Xu. Fedmoe: Personalized federated learning via heterogeneous mixture of experts. *arXiv preprint arXiv:2408.11304*, 2024.

Cui Miao, Tao Chang, Meihan Wu, Hongbin Xu, Chun Li, Ming Li, and Xiaodong Wang. Fedvla: Federated vision-language-action learning with dual gating mixture-of-experts for robotic manipulation. *arXiv preprint arXiv:2508.02190*, 2025.

Yao Mu, Tianxing Chen, Shijia Peng, Zanxin Chen, Zeyu Gao, Yude Zou, Lunkai Lin, Zhiqiang Xie, and Ping Luo. Robotwin: Dual-arm robot benchmark with generative digital twins. *arXiv preprint arXiv:2409.02920*, 2025.

Karl Pertsch, Konrad Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025.

D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, J. Gu, Z. Wang, Y. Ding, B. Zhao, D. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025.

Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Sparsely-gated mixture-of-experts. In *International Conference on Learning Representations*, 2017.

Weijie Shen, Yitian Liu, Yuhao Wu, Zhixuan Liang, Sijia Gu, Dehui Wang, Tian Nian, Lei Xu, Yusen Qin, Jiangmiao Pang, Xinping Guan, Xiaokang Yang, and Yao Mu. Expertise need not monopolize: Action-specialized mixture of experts for vision-language-action learning. *arXiv preprint arXiv:2510.14300*, 2025.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *Advances in Neural Information Processing Systems*, 2024.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.152.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. In *International Conference on Learning Representations*, 2024.

Yunlu Yan, Chun-Mei Feng, Wangmeng Zuo, Rick Siow Mong Goh, Yong Liu, and Lei Zhu. Federated residual low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2025.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. Gigaworld-policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026a.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026b.

Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, Wenqiang Zhang, and Cewu Lu. Forcevla: Enhancing vla models with a force-aware moe for contact-rich manipulation. In *Advances in Neural Information Processing Systems*, 2025.

Rongyu Zhang, Yijiang Liu, Huanrui Yang, Shenli Zheng, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Moant: Mixture-of-rank-one-experts with semantic-aware intuition for multi-task large language model finetuning. In *Third Conference on Language Modeling*, 2025.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18764–18772, 2026.

Yuhao Zhou, Yunpeng Zhu, Yang Zhou, Jindi Lyu, Jian Lan, Zhangyuan Wang, Dan Si, Thomas Seidl, Qing Ye, and Jiancheng Lyu. Forgevla: Federated vision-language-action learning without language annotations. *arXiv preprint arXiv:2605.07474*, 2026.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 12871–12882. PMLR, 2021.

APPENDIX

A ADDITIONAL EXPERIMENTAL DETAILS FOR SIMULATIONS

A.1 SHARED IMPLEMENTATION DETAILS

Parameter exchange. Both benchmarks use the same adapter-based implementation. LoRA adapts the attention, MLP, and state projections. Clients upload complete LoRA factors together with the trainable action input/output projections and action-time MLP weights, while frozen backbone weights are excluded. Shared action-head weights are averaged uniformly and synchronized separately from expert conversion. Server MoE checkpoints are saved before global adapter conversion and personalized redistribution.

Server participation and shared-compute assumption. ROBOFL lets the central server take part in training rather than act only as an aggregator. We regard this as a reasonable and common use of the decentralized setting: the server is typically the best-provisioned node, and federated methods routinely exploit server-side computation, for example through server-level adaptive optimization (Reddi et al., 2021), distillation of the client ensemble on server or surrogate data (Lin et al., 2020; Chen & Chao, 2021), or data-free server-side generators (Zhu et al., 2021). Under this assumption, the server holds one residual task group and performs the routed expert-refinement stage on it, which enables a single shared router over task-trained experts; the alternative reading, in which the server has no data of its own, would leave the router without a common distribution to learn from. Crucially, this does not amount to centralized training: the server group covers only a small fraction of each benchmark ($7/50$, $2/8$, and $2/6$ tasks on RoboTwin 2.0, RLBench, and Franka), so the bulk of the task distribution still resides with the clients. The server is also not given a disproportionate optimization budget: its per-round steps match the per-client local steps ($500/200/300$ for the three scenarios), and its task group is comparable in size to the client partitions (3-8, 1-2, and 1 task per client). Clients therefore continue to train a single adapter, and the additional routing and multi-expert computation remains on the server.

Comparison Fairness with Centralized Training Baselines We treat centralized PEFT as a deliberately *stronger* reference than any federated method, not as an equal-budget competitor. A centralized run observes the union of all task families in a single optimizer, with no task silos, no non-IID client drift, and no communication rounds. That ROBOFL matches or exceeds this reference on RoboTwin 2.0 and Franka while operating under task-silo federated learning, and, consistent with its mechanism, trails it only on the low-data RLBench suite, where less complementary evidence exists for expert composition, is the intended empirical claim of Section 4.

The comparison is also capacity-matched at the point of deployment. The entity that is trained, redistributed, and evaluated is a *single* rank-16 LoRA adapter per client, identical in family, rank, and scaling to the adapters used by federated averaging and by the centralized PEFT references. The eight-expert server MoE is a training-time construct that never leaves the server; clients never hold routers or multiple experts, and the deployed policy is the redistributed single adapter, not the server MoE. At the client boundary, ROBOFL is therefore indistinguishable from single-adapter FL in trainable parameters, communication, and memory (Table 3), and is far below client-side MoE methods. The reported gains consequently cannot be attributed to a larger deployed model or to additional client-side capacity: they follow from *how* task-specialized adapters are composed on the server under heterogeneity, not from giving the learned policy more parameters than its baselines. Notably, this advantage is obtained under an extreme non-IID, task-disjoint regime in which every client still trains and communicates a single adapter.

Configuration. Table 5 summarizes the numerical settings. RLBench training entries marked † are launcher defaults, not reconstructed settings of the reported checkpoints. Task assignments and evaluation semantics are specified separately below. Last but not least, all the FL-based baselines follow the same training procedure as ROBOFL.

Table 5: Training and evaluation settings for the simulation and real-world scenarios. Execution horizon is specified independently of prediction length.

Setting	RoboTwin 2.0	RLBench	Franka
`Optimization and adapters`			
Pretrained checkpoint		InternVLA-A1-3B	
Clients / experts	8 / 8	4 / 4	4 / 4
Routing top-k	4	2	2
Communi. rounds	100	100	100
Local steps / round	500	200	300
Local epochs	1	1	1
Server steps / round	500	200	300
Peak / final LR	10^{-4} / 10^{-5}	10^{-4} / 10^{-5}	10^{-4} / 10^{-5}
Warmup / decay steps	2,500 / 50,000	400 / 20,000	600 / 30,000
LoRA rank / α	16 / 32	16 / 32	16 / 32
LoRA dropout	0.05	0.05	0.05
λ_{gen} / λ_{aux}	0.01 / 0.001	0.01 / 0.001	0.01 / 0.001
`Observation and action interface`			
Views	Head + two wrists	Front + two masked slots	External stand + wrist + masked slot
Resolution	224×224	224×224	224×224
State	Joint configuration	XYZ + Euler + gripper	Joint + gripper
Action	Joint-space offsets	ΔXYZ + absolute Euler + gripper	Absolute joint targets
History stride	15	1	15
Predicted chunk	50	8	50
Inference steps	10	10	10
`Evaluation protocols`			
Evaluated tasks	50	8	6
Trials / task	100	50	20
Executed actions	30	8	50
Initialization	Randomized	seed 42	Operator reset

A.2 Hyperparameter and Design Choices

The non-obvious hyperparameters follow standard practice. The personalization coefficient $\frac{1}{2}$ in Eq. 4 represents an equal global-local model interpolation, a standard personalization strategy with generalization guarantees (Mansour et al., 2020). The local term preserves client specialization, while the global term shares server-acquired knowledge, and fixing it avoids introducing a per-client hyperparameter. The auxiliary weights are kept small because $\mathcal{L}_{\text{action}}$ remains the primary objective and only the magnitude of such weights matters: we use $\lambda_{\text{aux}} = 0.001$ (Kendall et al., 2018), with the load-balancing scale following the sparse-MoE convention (Shazeer et al., 2017; Fedus et al., 2022). FARD is likewise a routing regularizer based on detached Jensen-Shannon distillation (Hinton et al., 2015), so $\lambda_{\text{FARD}} = 0.01$, and it is warmed up over 10 rounds because the reliability weight is uninformative before the action router stabilizes. Warmup is a standard variance-reduction device (Liu et al., 2020).

A.3 RoboTwin 2.0

Task-silo manifest. Table 6 lists the task-disjoint client and server assignments implemented by `fedforesight_category`. Episodes remain within their assigned task partition, while the server residual group is used for routed optimization.

Table 6: Task assignments in the explicit eight-client task-silo implementation. Counts refer to tasks, not episodes or training frames.

Partition	Count	Tasks
Client 0	5	grab_roller, pick_diverse_bottles, pick_dual_bottles, put_bottles_dustbin, put_object_cabinet
Client 1	7	place_object_basket, place_bread_basket, place_bread_skillet, place_can_basket, place_cans_plasticbox, place_container_plate, place_empty_cup
Client 2	8	place_a2b_left, place_a2b_right, place_fan, place_mouse_pad, place_object_scale, place_object_stand, place_phone_stand, place_shoe
Client 3	6	stack_blocks_three, stack_blocks_two, stack_bowls_three, stack_bowls_two, blocks_ranking_rgb, blocks_ranking_size
Client 4	5	adjust_bottle, lift_pot, move_pillbottle_pad, move_playingcard_away, move_stapler_pad
Client 5	3	beat_block_hammer, press_stapler, stamp_seal
Client 6	6	click_alarmclock, click_bell, turn_switch, open_laptop, open_microwave, rotate_qrcode
Client 7	3	scan_object, shake_bottle, shake_bottle_horizontally
Server	7	place_burger_fries, place_dual_shoes, dump_bin_bigbin, handover_block, handover_mic, hanging_mug, move_can_pot

Observation and action interface. As shown in the Table 5, predicted joint-space offsets are converted to joint targets using the configuration at the start of each action queue. The evaluator truncates predictions using its execution-horizon argument, while prediction length does not determine the number of executed actions.

Sampling and update accounting. Local training cycles through a shuffled frame-level loader, without task-balanced sampling. The epoch setting repeats the local-step loop rather than guaranteeing a full dataset traversal of the dataset. For C clients, K rounds, L local steps, E repetitions, and M server steps, the update counts are KLE per client, $CKLE$ across clients, and KM at the server. The schedule is given in Table 5.

Batch-size conventions. Client batch size is per local update, while server batch size is per distributed rank. With R server ranks and per-rank batch size B_{srv}, the effective server batch is RB_{srv}. Client optimizer states are retained separately across rounds, while server optimizer state persists across expert replacement.

Randomized initialization. As shown in Table 5, unstable or expert-unsuccessful candidate scenes are excluded before policy evaluation. Accepted trials therefore need not correspond to consecutive candidate seeds. The evaluator defaults to the unseen instruction split.

A.4 RLBench

Task-silo manifest. Table 8 identifies the reported evaluation tasks, not client IDs. The training implementation provides task-disjoint by_task partitions with four clients and a separate server.

Observation and action interface. See Table 5. Unlike RoboTwin, each queued translation increment is added to the current end-effector position, while the predicted Euler angles specify an absolute world-frame orientation. The resulting pose is executed by a motion planner. Stored translation labels are already relative while preprocessing must not subtract the state again.

Table 7: RoboTwin 2.0 task success rates (%) on the complementary 25 tasks not shown in Table 1. Best and second-best reported results are boldfaced and italicized.

Method	adjust bot.	rank RGB	click alarm	click bell	dump bin	grab roller	hand mic	move pill	move card	open laptop	open micro.	bread bas.	burger fries
CENTRALIZED TRAINING													
INTERNVLA	*99*	91	**85**	91	**98**	**100**	**95**	87	**100**	**98**	**91**	*87*	**99**
MOTUS	**100**	*94*	83	90	91	**100**	81	*91*	97	92	80	81	93
FEDERATED LEARNING(LoRA/MoE)													
FEDAVG	*99*	91	*84*	*93*	95	**100**	89	83	98	90	84	82	*98*
FEDMOE	**100**	91	76	91	94	**100**	82	80	95	88	65	80	97
FORGEVLA	**100**	90	**85**	92	94	**100**	82	*91*	*98*	*93*	82	82	93
FEDVLA	*99*	88	82	89	92	*99*	83	83	96	90	78	82	96
ROBOFL	**100**	**95**	83	**96**	*97*	**100**	*92*	**92**	**100**	90	*85*	**90**	93

Method	care box	cont. plate	empty cup	place fan	obj. stand	phone stand	place shoe	shake bot.	shake horiz.	stack 3 blo.	stack 2 blo.	stack 2 bow.	**Overall**
CENTRALIZED TRAINING													
INTERNVLA	**97**	97	**100**	**91**	88	*94*	*93*	*99*	97	**87**	**100**	98	*81.60*
MOTUS	**97**	*99*	**100**	85	87	93	91	**100**	**100**	83	*99*	98	80.96
FEDERATED LEARNING(LoRA/MoE)													
FEDAVG	*94*	**100**	**100**	86	84	91	*93*	**100**	97	*84*	*99*	**100**	78.92
FEDMOE	85	*99*	**100**	84	*89*	91	86	*99*	*99*	80	*99*	**100**	75.12
FORGEVLA	93	**100**	**100**	88	85	*94*	92	**100**	**100**	81	**100**	*99*	80.70
FEDVLA	89	**100**	**100**	83	84	90	90	**100**	*98*	84	95	**100**	79.32
ROBOFL	*94*	**100**	**100**	*90*	**91**	**95**	**95**	**100**	**100**	82	**100**	**100**	**83.12**

Table 8: RLBench evaluation manifest. Rows identify evaluation tasks, not federated clients.

Partition	Count	Tasks
Client 0	1	`take_umbrella_out_of_umbrella_stand`
Client 1	2	`close_fridge`, `toilet_seat_down`
Client 2	2	`close_laptop_lid`, `close_box`
Client 3	1	`sweep_to_dustpan`
Server	2	`put_rubbish_in_bin`, `phone_on_base`

Sampling and update accounting. The implementation uses the same shuffled-loader and update-counting conventions as RoboTwin. Table 5 records launcher defaults.

Batch-size conventions. The same per-client-update and per-server-rank conventions apply as in RoboTwin for the independent-client configuration. A grouped-client distributed run also requires the number of ranks per client to determine the effective local batch size.

Randomized initialization. Simulator resets use the variation and seed in Table 5, without RoboTwin's expert-based candidate screening. Invalid actions and failures terminate the episode and count as failures. Expert actions and future observations are not supplied to the policy.

B ADDITIONAL EXPERIMENTAL DETAILS FOR REAL-WORLD FRANKA TASKS

B.1 IMPLEMENTATION DETAILS

Task-silo manifest. Table 9 identifies the six reported single-arm Franka tasks and their client and server assignments. The implementation provides task-disjoint partitions among four clients and a separate server, with each client owning one task family and the server retaining the remaining two. Evaluation is performed via real-robot inference on the physical platform described in Appendix B.2.

Table 9: Franka real-robot manifest. Rows identify evaluation tasks, not federated clients.

Partition	Count	Tasks
Client 0	1	`stack_cups`
Client 1	1	`screw_bottle`
Client 2	1	`stamp_seal`
Client 3	1	`test_tube`
Server	2	`adjust_bottle`, `pour_water`

Observation and action interface. See Table 5. Unlike RoboTwin, each action is an absolute joint target executed directly by the robot, while queued targets are not added to the state at the start of the action queue. The policy consumes the external stand and wrist views, with the third view slot masked. State and action are both eight-dimensional, and auxiliary tactile and wrench channels are not supplied to the policy.

Sampling and update accounting. The implementation uses the same shuffled-loader and update-counting conventions as RoboTwin and RLBench. Table 5 records the schedule.

Batch-size conventions. The same per-client-update and per-server-rank conventions apply as in RoboTwin and RLBench for the independent-client configuration. The client batch size is 32 per local update, while the server batch size is 16 per distributed rank.

Evaluation protocol. Each method is evaluated in three rounds of 20 trials per task. Each real-robot trial is started and reset by the operator, which places the objects and fixtures in their nominal task configuration. The policy then executes in a closed loop on the physical robot, and a trial is scored as successful only if the task goal is reached; otherwise, the episode terminates and counts as a failure.

B.2 Franka Single-Arm Platform

Robot and gripper. Figure 6 shows the hardware configuration for our Franka single-arm experiments. The platform consists of a Franka Research 3 manipulator equipped with a Robotiq 2F-85 gripper. The robot is mounted beside a tabletop workspace containing the objects and fixtures used in the manipulation tasks.

Camera arrangement. Two RealSense D435i cameras provide complementary views of the workspace. One camera is mounted externally on a stand and faces the tabletop, providing an overview of the scene and object arrangement. The second camera is mounted on the robot wrist, providing a local view of the manipulation area as the arm moves. This configuration combines an external scene view with a close-range view of gripper-object interactions.

Workspace and task objects. The workspace contains a perforated board and task-specific objects, including bottles, cups, a stamp and ink pad, glassware, and a test-tube rack. These objects support the six Franka tasks reported in the main text: *adjust bottle*, *stamp seal*, *stack cups*, *screw bottle*, *pour water*, and *test tube*.

C Additional Experimental Results

C.1 Complementary Quantitative Analysis for RoboTwin 2.0

The complementary results explain the emphasis on challenging tasks in Table 1. Most reported scores on this subset exceed 90%, with many at or near 100%, leaving limited headroom to distinguish methods. The main table therefore focuses on a lower-success subset where FedAvg scores below 80%. This presentation makes RoboFL's improvements on difficult manipulation tasks more visible, rather than relying on comparisons among near-saturated results, while the appendix preserves complete task coverage.

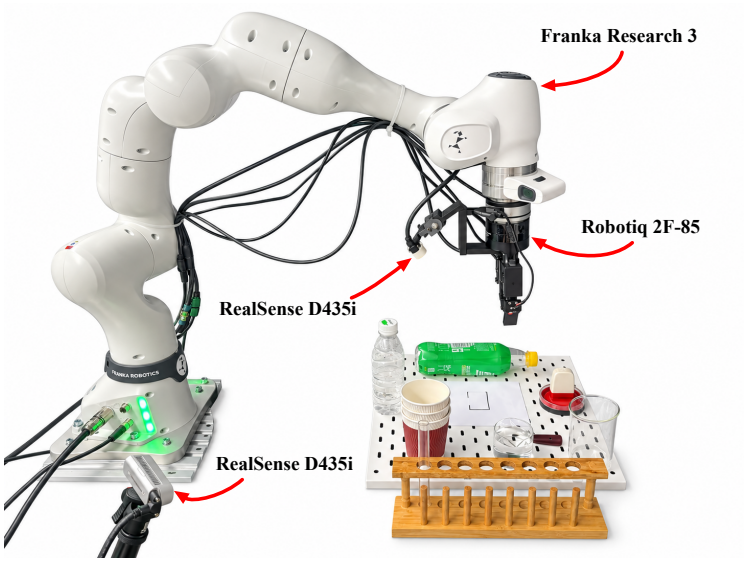

Figure 6: **Franka single-arm experimental setup.** The platform comprises a Franka Research 3, a Robotiq 2F-85 gripper, and two RealSense D435i cameras mounted on the wrist and an external stand, respectively. The tabletop contains the objects and fixtures used in the manipulation tasks.

Table 10: Ablation of MoSAIC components on RoboTwin 2.0. All rows report overall success rates (%) on all 50 tasks with 100 trials per task. The three MoSAIC rows use the same federated pipeline and differ only in the server mechanism; FedAvg and FedMoE are external references.

Configuration	Overall
FedAvg (averaged single adapter)	78.92
FedMoE (client-side routed MoE)	75.12
Vanilla MoSAIC (task-trained experts + routing)	81.94
+ FARD	82.42
+ FARD + PCEA (RoboFL)	**83.12**

Despite the higher success rates and reduced room for improvement, RoboFL remains highly competitive, achieving best or joint-best results on 16 of the 25 tasks and 100% observed success on nine. Strong results across ranking, object placement, and bottle shaking show that its performance extends beyond the difficult cases highlighted in the main table. Together, the two subsets demonstrate that RoboFL combines improvements on challenging tasks with strong performance on the more saturated portion of the benchmark.

In addition, Table 10 reports the overall success rate of MoSAIC components on the full 50-task RoboTwin 2.0 benchmark. Installing task-trained adapters as expert branches with learned routing (vanilla MoSAIC) reaches 81.94%; adding routing distillation (FARD) raises this to 82.42%; and adding consensus-weighted conversion (PCEA) reaches 83.12%. The two external references put these numbers in context: federated averaging reaches 78.92%, while a client-side-routed LoRA-MoE (FedMoE) reaches 75.12%. Most of the gain, therefore, comes from retaining task-trained experts and learning their composition on the server (3.02% over FedAvg), with FARD and PCEA contributing smaller but consistent additional improvements (0.48% and 0.70%, respectively). FARD and PCEA mainly affect the distribution of high-success tasks as shown in Section 4.3.

C.2 Quantitative Analysis for RLBench

We further evaluate the methods in a lower-data, single-view RLBench setting with eight tasks and 50 trials per task. RoboFL reaches 41.25%, the best among federated methods, improving over FL baselines. Unlike RoboTwin, where RoboFL also surpasses centralized PEFT, it trails centralized

Table 11: RLBench task success rates (%) on the 8-task single-view suite. Overall is the equally weighted mean across the eight tasks. Best and second-best results are boldfaced and italicized.

Method	put rubbish	seat down	draw umbrella	close laptop	sweep dustpan	close fridge	close box	phone on base	Overall
CENTRALIZED TRAINING									
INTERNVLA	**60**	*18*	*34*	*62*	*66*	46	*86*	**50**	**52.75**
MOTUS	*44*	*22*	**36**	58	**72**	58	**92**	*38*	*52.50*
FEDERATED LEARNING(LoRA/MoE)									
FEDAVG	0	18	22	4	14	**72**	**92**	0	27.75
FEDMOE	0	12	16	14	22	64	70	8	25.75
FORGEVLA	12	14	32	26	20	*70*	84	16	34.25
FEDVLA	6	16	24	12	30	56	80	24	31.00
ROBOFL	26	**24**	32	**64**	24	50	60	**50**	41.25

InternVLA (52.75%) and Motus (52.50%) on RLBench. We hypothesize that the smaller single-view suite provides less task diversity and complementary evidence for expert composition, whereas centralized PEFT can directly pool all available examples. RoboTwin better represents the broad, task-heterogeneous distributed regime in which preserving specialized adapters is most valuable. These results therefore suggest that ROBOFL is particularly suited to large-scale distributed learning with complementary non-IID data.

C.3 QUALITATIVE ANALYSIS

We visualize successful rollouts to examine whether the reported success rates correspond to sensible task behavior: approach and alignment, contact or grasp, and execution of the intended state change. Each task contributes four chronologically ordered keyframes from a single successful episode, and the frames are exported at their native resolution without cropping or enhancement. Because the evaluators record an observation before each executed action, the final frame may precede the release or contact that triggered the success flag, so visual inspection captures the executed behavior rather than confirming the terminal success event.

RoboTwin 2.0. Figure 7 and 8 show one successful rollout for each of the 50 RoboTwin 2.0 tasks. Across these rollouts, the policy exhibits a consistent three-phase structure: an alignment phase in which the end effector approaches the target, a contact or grasp phase, and a transport or placement phase. Articulated and contact-rich tasks such as *open microwave*, *close laptop lid*, and *rotate qrcode* show the gripper establishing contact and then applying a sustained motion rather than a ballistic strike. Multi-object tasks such as the stacking and ranking families, *place bread basket*, *place cans plasticbox*, and *place dual shoes* involve sequential subgoals, in which the second object is manipulated only after the first has been positioned. The tasks that remain difficult in the quantitative results, notably *hanging mug* and *lift pot*, are also visually harder: they require precise two-sided or handle-based contact under substantial occlusion, and their successful rollouts show a slower, more tentative approach. The remaining failures in the report are therefore concentrated in fine-grained contact and long-horizon placement rather than in gross reaching or scene understanding.

RLBench. Figure 9 shows successful rollouts for the eight-task single-view suite under the evaluation protocol of Table 5: front camera, variation 0, and an eight-action chunk queue. The rollouts reveal several recurring behaviors. Closing tasks, including *close Box*, *close fridge*, and *close laptop lid*, are executed as a short contact-and-push sequence: the gripper reaches the lid or door, establishes contact, and then drives it through its range in a single continuous motion. Grasping-and-relocation tasks such as *put rubbish in bin* and *take umbrella out of umbrella stand* show a descending approach and a closed grasp before the object is lifted clear. Tool-use and placement tasks, *sweep to dustpan* and *phone on base*, require the gripper to align with a narrow target region before the state change.

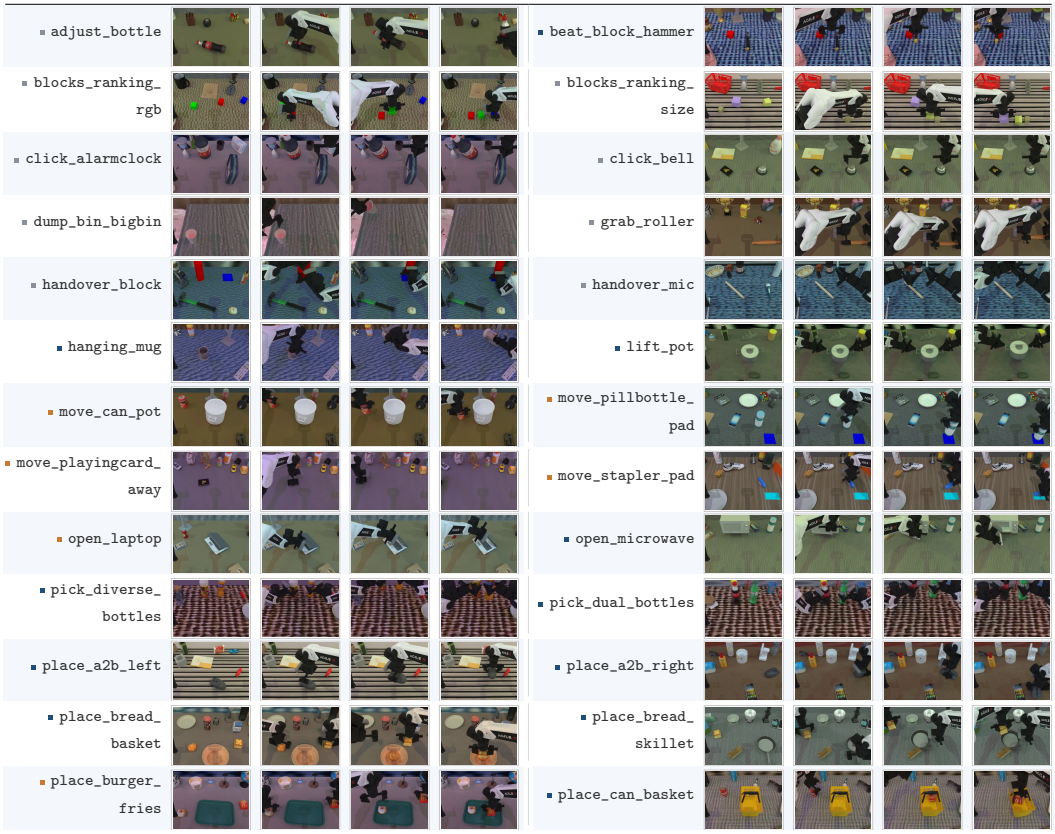

Figure 7: **Qualitative RoboTwin 2.0 rollouts (part 1 of 2).** Each row shows two tasks. Every task is labeled with its name and contributes four chronologically ordered keyframes from one successful rollout, illustrating approach, contact or grasp, and execution of the intended state change. Frames are exported at their native resolution without cropping or enhancement.

C.4 Failure Analysis

Despite the overall gains, we observe recurring failures in both simulation and hardware, as illustrated in Figures 10 and 11. **1) Incomplete Object Engagement:** In *lift pot* and *hanging mug* the gripper topples or tilts the object instead of lifting or hanging it, and on Franka *screw bottle* never achieves threaded engagement. This suggests that a skill absent from every client family must be synthesized by expert composition, which can fall back to a blend of adjacent experts. **2) Stalled or Abandoned Motion:** *stamp seal* hovers without pressing, *scan object* drifts away after approaching, and on Franka *pour water* and *test tube* stop short of the manipulation, consistent with weak understanding-generation consensus under self-occlusion leaving the router under-constrained. **3) Misplaced End State:** *put object cabinet* drags the object instead of stowing it, and *stack cups* lands beside the base, indicating that rank-limited conversion attenuates the expert-specific component of a terminal motion. The RLBench failures in Figure 10 (bottom) differ, all ending in a motion-planning error (*InvalidActionError*) rather than a policy error. Overall, these cases highlight that skills absent from all clients, ambiguous routing, and lossy conversion remain the main bottlenecks, which we plan to instrument in future work.

D Statistical Reporting Protocol

Recent VLA and WAM work typically reports success rates over a fixed number of execution trials from a single training run rather than averaging over multiple training seeds on the simulation: for example, Motus evaluates each RoboTwin 2.0 task over 100 execution trials from one fine-tuned model (Bi et al., 2026), and comparable single-run protocols are used for InternVLA-A1 (Cai et al.,

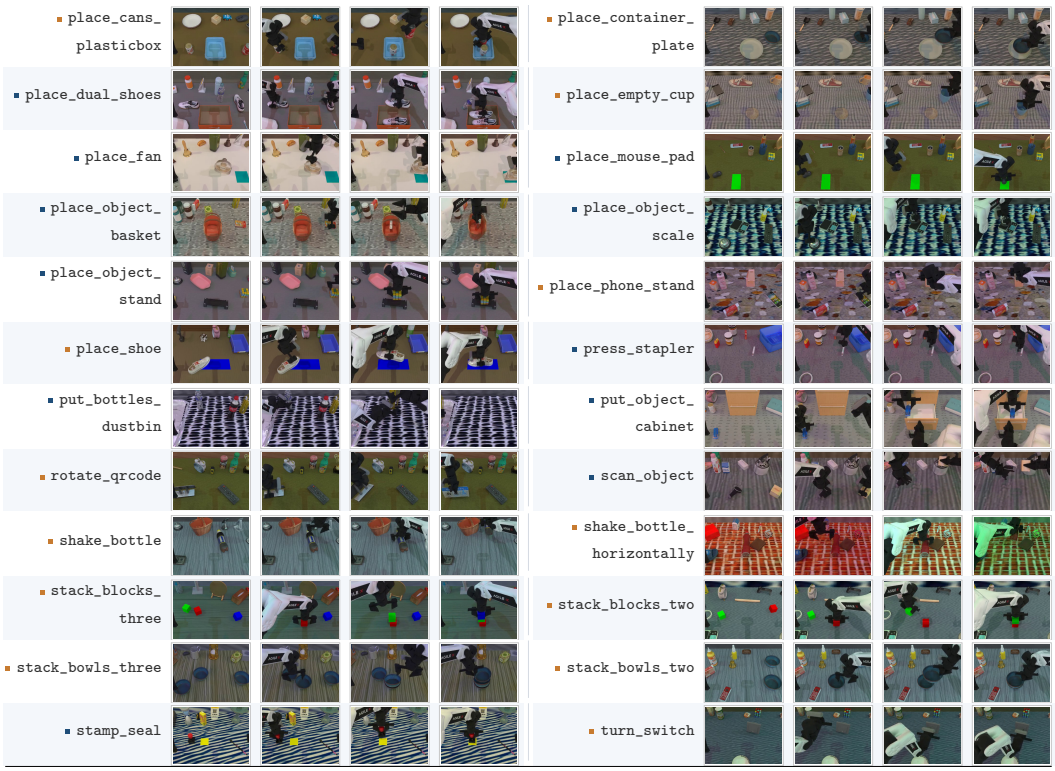

Figure 8: **Qualitative RoboTwin 2.0 rollouts (part 2 of 2).** Each row shows two tasks. Every task is labeled with its name and contributes four chronologically ordered keyframes from one successful rollout, illustrating approach, contact or grasp, and execution of the intended state change. Frames are exported at their native resolution without cropping or enhancement.

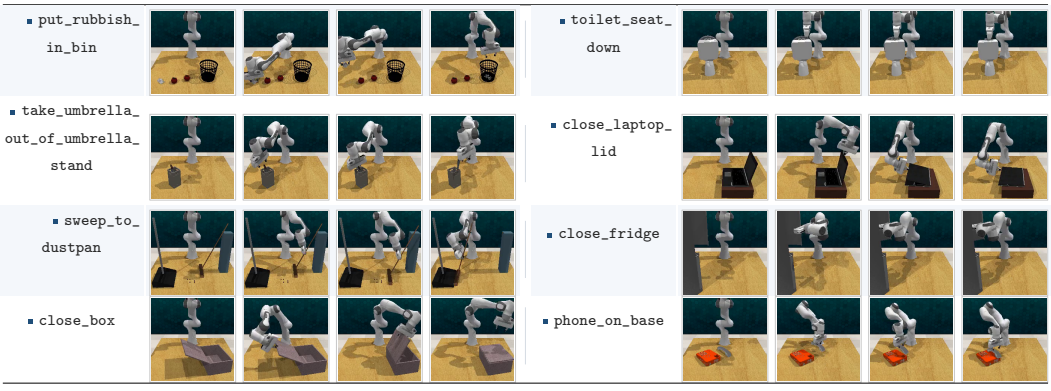

Figure 9: **Qualitative RLBench rollouts.** Each row shows two tasks, each labeled with its name and the number of successful episodes out of 50, and includes up to four chronologically ordered keyframes from one successful rollout. All rollouts use the single-view interface with the front camera at variation 0 and the eight-action chunk-queue protocol. The displayed episode for each task was selected for visual clarity rather than being the longest or most representative rollout.

2026) and ForgeVLA (Zhou et al., 2026). We follow this convention: all simulation results are obtained from a single fixed training seed and a single evaluation seed, with 100 (RoboTwin 2.0) and 50 (RLBench) trials per task, and the real-robot protocol is a fixed, scripted sequence. This keeps our results directly comparable to the benchmarks and baselines that we report. Multi-seed training with confidence intervals would strengthen the evidence for these smaller effects and is a natural direction for future work.

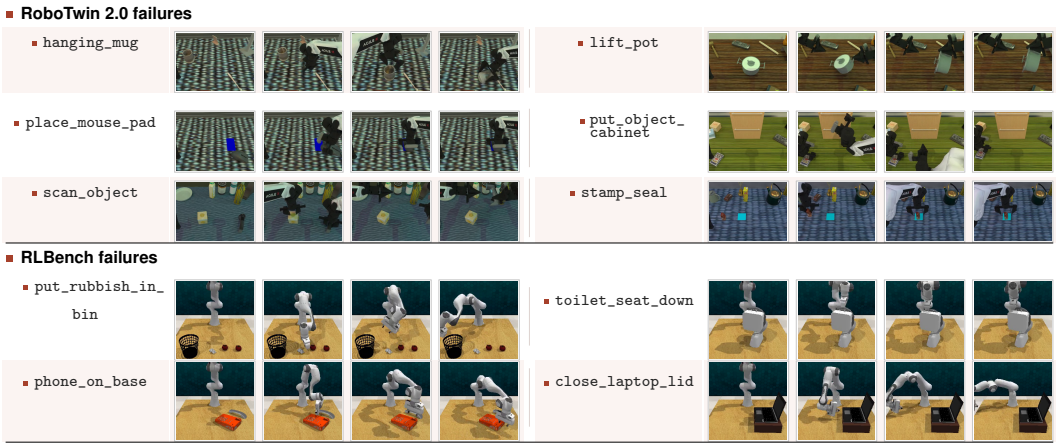

Figure 10: **Failure cases across benchmarks.** RoboTwin 2.0 failures (top) and RLBench failures (bottom) contribute four chronologically ordered keyframes from one failed rollout: initial scene, first meaningful attempt, visible deviation, and final failing state. Because observations are recorded before actions and the video ends at the failure, the last frame shows the episode-terminated state.

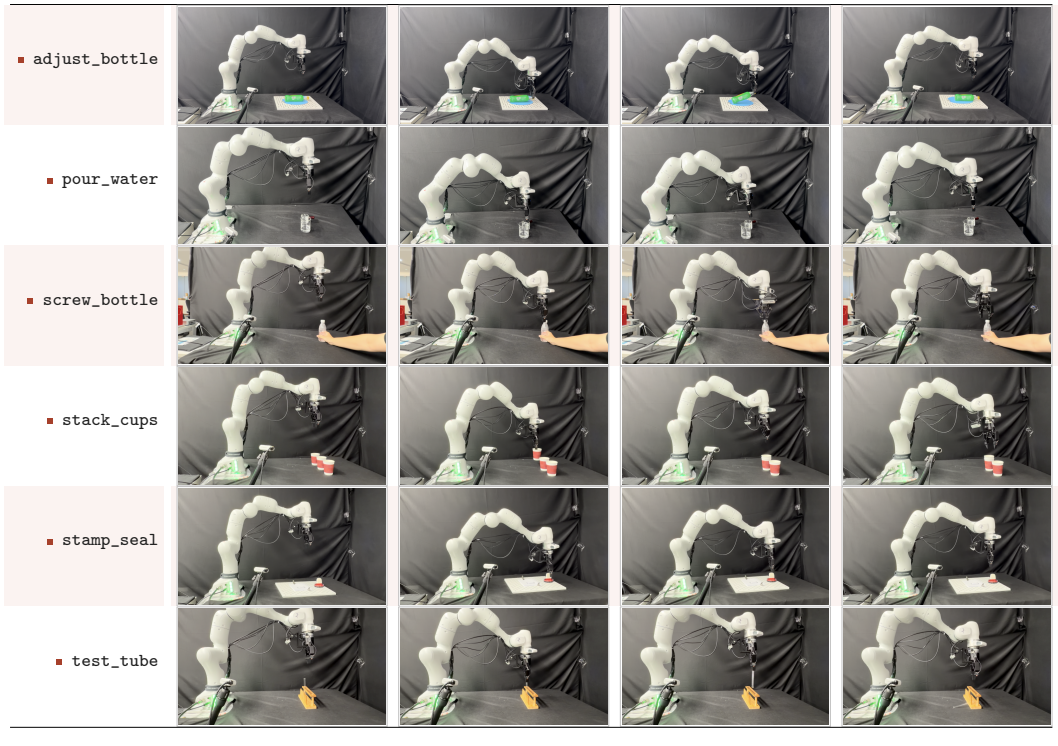

Figure 11: **Franka failure cases.** Failed rollouts for six Franka tasks, with each row contributing four chronologically ordered keyframes (initial scene, first attempt, visible deviation, final state).

E LIMITATIONS

Our study has several limitations. First, all experiments use a single backbone (InternVLA-A1-3B) and one aligned MoT architecture. Second, we target task heterogeneity: clients differ in task distributions but share the embodiment, sensor suite, and action space, so cross-embodiment and cross-modal heterogeneity remain open questions. Finally, simulation results use a single evaluation seed, and the real-robot results use a fixed scripted protocol with 20 trials per task.