FEDWM: PERSONALIZED FEDERATED LEARNING FOR HETEROGENEOUS VISUAL WORLD MODELS

ABSTRACT

Federated learning can combine experience across clients to train visual world models without pooling their trajectories. When clients differ in appearance and dynamics, the central design question is which parts of the predictor to share and where to personalize. We introduce FedWM, a federated training framework that places private residual modules at both the interfaces and internal layers of action-conditioned world models. Clients first learn a shared core through parameter averaging, then freeze it and train private modules locally with the backbone's native prediction objective. We instantiate FedWM separately with predictive DINO-WM and diffusion-based NanoWM across PushT, PointMaze, Wall, and Rope. With matched training steps, FedWM lowers mean held-out prediction error relative to interface-only personalization in all eight backbone–environment settings, each evaluated with three training seeds. A near-capacity-matched DINO-WM PushT ablation also favors internal residuals over wider interfaces. In DINO-WM studies, FedWM retains its advantage over interface-only adaptation with eight clients at fixed total data and in most low-data new-client settings. Closed-loop PushT comparisons favor FedWM over FedAvg on validation but not on the frozen test bank, highlighting the need to evaluate prediction and control separately.

1 INTRODUCTION

Action-conditioned visual world models predict how a scene changes under candidate actions, allowing a planner to evaluate possible behaviors before executing them. Models based on pretrained visual representations support goal-conditioned planning without learning a separate policy for each goal (Zhou et al., 2024); diffusion-based models instead generate future video conditioned on action sequences (Chen et al., 2024; Huang et al., 2026). In both cases, useful experience may be distributed across robots or data holders whose trajectories cannot be pooled. Federated training offers a way to learn from that experience while keeping trajectories local.

Collaboration is complicated by differences in client appearance, geometry and physical behavior. Clients may share transition structure, yet the same action can lead to different next states or observations under local conditions. Personalizing a predictor's interfaces can adjust how observations enter and leave the model. Residual corrections inside the predictor can also adapt the intermediate computations that determine future states. The design question is how to allocate shared and private parameters across these computations.

We propose FedWM, a federated training framework for action-conditioned world models with a shared prediction core and private residual modules. Training proceeds in two stages: clients collaboratively learn the core through parameter averaging, then freeze it and optimize residuals at the model interfaces and selected internal sites. Each backbone retains its architecture, conditioning and prediction objective. This separates the federation protocol from the choice of world model, as illustrated in Figure 1.

FedWM builds on shared/private parameter decompositions, staged personalization and internal adapters (Arivazhagan et al., 2019; Collins et al., 2021; Oh et al., 2021; Sun et al., 2023; Houlsby et al., 2019). Its focus is their use in recursive action-conditioned prediction: which components can be shared, where client-specific corrections are applied, and how the resulting models perform in prediction and planning. We implement the framework with DINO-WM and NanoWM in separate predictive and diffusion federations. The main implementation places internal residuals in feed-forward networks (FFNs) or multilayer perceptrons (MLPs); controlled studies compare alternative placements.

Our evaluation covers four environments and three training seeds for each backbone–environment pair. FedWM lowers mean prediction error relative to interface-only personalization in all eight

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.

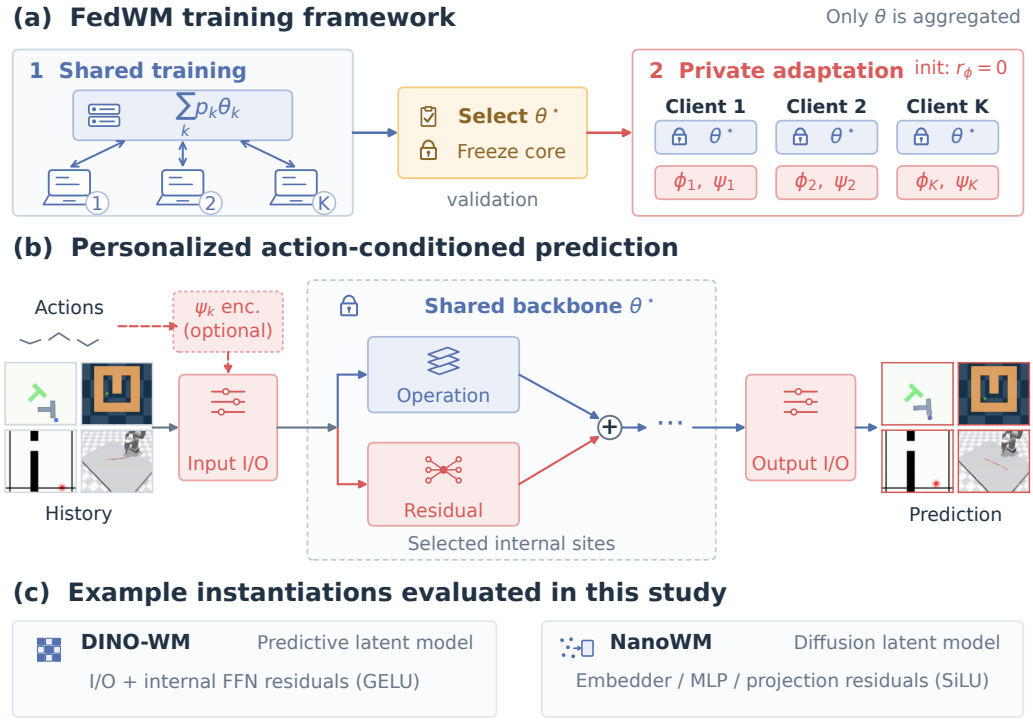


Figure 1: FedWM training framework. (a) Clients aggregate θ, freeze the selected core and train private residuals ϕ_k initialized to output zero; optional encoders ψ_k remain local. (b) Private boundary and internal modules adapt action-conditioned prediction. Thumbnails pair recorded NanoWM inputs and FedWM outputs for PushT, PointMaze, Wall and Rope in row-wise order. (c) DINO-WM and NanoWM instantiate the framework in separate federations.

settings. A near-capacity-matched PushT study tests whether wider interfaces account for this gain, followed by placement, client-count and new-client studies. Closed-loop evaluation then examines whether prediction improvements translate into better action selection. We report training and communication costs by stage so that these comparisons can be interpreted alongside their resource budgets.

2 Related Work

Personalized federated learning. FedAvg aggregates locally trained parameters into a shared model (McMahan et al., 2017). Personalized methods partition this learning problem in different ways: FedPer keeps personalization layers local (Arivazhagan et al., 2019), FedRep learns a shared representation with local heads (Collins et al., 2021), and FedBABU separates body training from adaptation (Oh et al., 2021). FedWM uses a shared/private decomposition and staged training for action-conditioned visual prediction. Its interface-only comparator follows the same frozen-core adaptation schedule, providing a structural comparison within this protocol rather than a reproduction of the original FedPer schedule.

Internal and parameter-efficient adaptation. Bottleneck adapters and low-rank updates adapt pretrained models through a small set of trainable parameters (Houlsby et al., 2019; Hu et al., 2021). In federated learning, FedPerfix personalizes transformer components and evaluates private MLP adapters (Sun et al., 2023), while FedTP personalizes attention through a hypernetwork (Li et al., 2022). FDLoRA combines shared and private low-rank adaptation for language models (Lu et al., 2024), and pFedMMA addresses vision-language adaptation (Ghiasvand et al., 2025). Our study examines internal residual adaptation in world models, where client-specific changes affect recursive state prediction and the action costs used by a planner. The interface, capacity and placement controls in Section 4 assess these choices under a common training protocol.

World models and their adaptation. DINO-WM predicts pretrained spatial visual features for goal-conditioned planning (Zhou et al., 2024; Oquab et al., 2023). NanoWM uses a compact

diffusion-forcing model to predict future video (Huang et al., 2026; Chen et al., 2024). We use both as examples of FedWM, preserving their native prediction objectives and architecture-specific computations. Related adaptation settings include ReDRAW's residual corrections to latent dynamics (Lanier et al., 2026) and FedWorld's federation of structured transition rules for language-model agents (Hou, 2026). FedWM instead trains continuous neural visual dynamics across clients and evaluates the resulting personalized predictors in both rollout prediction and closed-loop control.

3 FEDWM

3.1 PROBLEM AND PARAMETER OWNERSHIP

Client k holds a dataset $\mathcal{D}_k$ of visual observations, actions and any required proprioception. A backbone defines the representation, action conditioning and prediction loss $\mathcal{L}_k$. FedWM specifies the ownership of its trainable parameters and the shared and private training stages.

We partition the trainable parameters into a shared core θ, private residuals ϕ_k, and optional private encoders ψ_k. Only θ is aggregated. The residuals $\phi_k = (\phi_k^{\text{boundary}}, \phi_k^{\text{internal}})$ are added after shared training. When the backbone uses client-specific action or proprioception encoders, their parameters ψ_k remain local and are updated in both stages. Raw trajectories stay with their clients.

Boundary residuals adjust the representations entering and leaving the predictor; internal residuals modify intermediate prediction computations. An instantiation defines the tensors eligible for aggregation, any local encoders, and the insertion sites and shapes of the residual modules. Each residual matches the shape of the operation it corrects, allowing the client to adapt that operation while preserving the backbone's representation and conditioning mechanisms (Figure 1).

3.2 COLLABORATIVE SHARED TRAINING

At round r, client k loads $\theta^{(r)}$ and its current local encoders, when present. It minimizes $\mathcal{L}_k$ for the prescribed number of local updates and returns the updated shared parameters $\theta_k^{(r+1)}$. The server aggregates them with nonnegative weights p_k:

$$\theta^{(r+1)} = \sum_{k=1}^{K} p_k \theta_k^{(r+1)}, \qquad \sum_k p_k = 1. \tag{1}$$

All clients participate in every reported round. DINO-WM weights clients by their number of training windows; NanoWM uses $p_k = 1/K$. Local encoder states persist across rounds. Validation selects the shared checkpoint $\theta^\star$ and, where applicable, its corresponding client-local encoder states to initialize private adaptation.

3.3 PRIVATE RESIDUAL ADAPTATION

Each client freezes $\theta^\star$ and adds bottleneck residuals at the boundaries and selected internal sites:

$$r_{k,\ell}(v) = U_{k,\ell}\, \sigma_{\mathcal{B}}\big(D_{k,\ell} v + b_{k,\ell}^D\big) + b_{k,\ell}^U, \tag{2}$$

where $\sigma_{\mathcal{B}}$ is the backbone-specific activation. The insertion site determines the input v, including any shared normalization applied before the residual. We initialize $U_{k,\ell} = 0$ and $b_{k,\ell}^U = 0$, so every added branch initially outputs zero and preserves the selected core's computation.

The client then applies its fixed private training schedule:

$$(\phi_k^\star, \psi_k^\star) \;\leftarrow\; \text{LocalTrain}[\mathcal{L}_k(\theta^\star, \phi_k, \psi_k; \mathcal{D}_k)]\,, \tag{3}$$

Only ϕ_k and any trainable ψ_k are updated, using the backbone's prediction objective. Private parameters remain local throughout training. Validation selects the private checkpoint, except in the new-client study, which uses a fixed final epoch. Omitting the internal residuals yields the interface-only structure used by the FedPer-style comparator.

3.4 BACKBONE INSTANTIATIONS AND DEPLOYMENT

Predictive example: DINO-WM. The dynamics predictor forms the shared core. Private modules comprise input/output (I/O) adapters and a width-32 residual with a Gaussian error linear unit (GELU) activation parallel to the FFN in each of six Transformer blocks. The shared FFN and private branch receive the same normalized input; the original identity path and final normalization are preserved. The I/O adapters leave action slots unchanged. PointMaze, Wall and Rope also use local action/proprioception encoders ψ_k; PushT uses frozen representations.

Diffusion example: NanoWM. The base diffusion model forms the shared core. Private residuals with sigmoid linear unit (SiLU) activations adapt the outputs of its input/action embedders and final projection. Additional residuals run in parallel with the internal MLPs. Boundary wrappers compute $b(x) + r_k(b(x))$; internal wrappers compute $b(h) + r_k(h)$. The base conditioning, gates and skip paths are preserved. DINO-WM and NanoWM are trained in separate federations.

At deployment, a client combines $\theta^\star$ with its private parameters and uses the backbone's rollout or sampling procedure. The PushT control study uses model predictive control (MPC) with the cross-entropy method (CEM) under the fixed protocol in Section 4.5. Algorithm 1 and Figure 7 in Appendix A detail the training procedure and module paths.

4 EXPERIMENTS

4.1 EVALUATION SETUP

Environments and clients. We evaluate PushT, a tabletop pushing environment (Chi et al., 2023), together with PointMaze, Wall and Rope. Each environment has four client profiles that vary physical, visual or geometric conditions. Wall crosses wall and door locations; the other environments cross dynamics and appearance factors (Figure 8). Each backbone–environment pair is trained in a separate homogeneous federation with three fixed training seeds. Appendix B defines the profiles and dataset sizes, and Appendix C specifies training and checkpoint selection. The client-count and new-client extensions each use one training seed.

Comparators and metrics. We compare FedWM with FedAvg, FedPer-style (private I/O adapters), and Centralized training on pooled data. Existing DINO-WM Local results provide a client-local reference. For NanoWM, FedAvg-S denotes the shared checkpoint that initializes private adaptation. DINO-WM is evaluated by latent composite error on PushT, horizon-averaged observation mean squared error (MSE) on PointMaze, and held-out window prediction loss on Wall and Rope. PushT and PointMaze rollout evaluations use horizons 1, 3 and 5. NanoWM uses horizon-averaged learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018). Lower values indicate better prediction within each environment's metric. We report means and sample standard deviations (SDs) over three training seeds; clients and horizons are aggregated within each run.

Budget controls. FedWM and FedPer-style use matched training steps in the main personalization comparisons. In the older DINO-WM PointMaze, Wall and Rope protocols, both personalized methods use twice as many total optimizer steps as FedAvg. NanoWM matches FedAvg's total steps in 11 of 12 environment–seed combinations; the remaining PushT combination differs by four steps (0.00264%). The separate PushT control study also matches total training steps. Section 4.6 reports these budgets by stage.

4.2 PREDICTION ACROSS ENVIRONMENTS AND BACKBONES

FedWM lowers mean prediction error relative to interface-only personalization in all four environments with both backbones (Tables 1 and 2). The improvement holds for all three training seeds in each of the eight settings, supporting the addition of private internal residuals to the existing interface adapters.

In DINO-WM, paired-seed reductions over FedPer-style range from 1.79% on PushT to 13.38% on Wall. NanoWM shows the same direction under a diffusion objective and also improves over FedAvg with near-matched total training steps. Centralized performance depends on the environment and training schedule: extending PushT training yields 0.080977 ± 0.004401, compared with FedWM's

Table 1: DINO-WM prediction error (mean ± sample SD, three training seeds; lower is better). Each environment uses its native metric. FedWM and FedPer-style have matched steps; their schedules use twice FedAvg's steps in PointMaze, Wall and Rope. PushT includes the extended Centralized reference.

Environment	Centralized Pooled reference	FedAvg Shared model	FedPer-style I/O-only	FedWM I/O + internal
PushT	0.08098 ± 0.00440	0.10495 ± 0.00838	0.08302 ± 0.00653	0.08154 ± 0.00654
PointMaze	0.04210 ± 0.00297	0.20839 ± 0.00166	0.04663 ± 0.00496	0.04286 ± 0.00554
Wall	0.02795 ± 0.00119	0.08365 ± 0.00090	0.03668 ± 0.00121	0.03177 ± 0.00120
Rope	0.26568 ± 0.00446	0.69788 ± 0.00371	0.28527 ± 0.00546	0.27676 ± 0.00542

Table 2: NanoWM LPIPS-Hmacro over four-client federations (mean ± sample SD, three training seeds; lower is better). FedWM and FedAvg differ by at most four total training steps. FedAvg-S results appear in Appendix D.

Environment	Centralized Pooled reference	FedAvg Shared model	FedPer-style I/O-only	FedWM I/O + internal
PushT	0.04939 ± 0.00113	0.10266 ± 0.00500	0.05917 ± 0.00016	0.05540 ± 0.00068
PointMaze	0.01594 ± 0.00098	0.03228 ± 0.00256	0.01603 ± 0.00014	0.01561 ± 0.00005
Wall	0.00601 ± 0.00023	0.00821 ± 0.00031	0.00758 ± 0.00013	0.00690 ± 0.00009
Rope	0.05716 ± 0.00083	0.07523 ± 0.00261	0.07277 ± 0.00191	0.06978 ± 0.00139

0.081536 ± 0.006541. Table 1 uses this extended reference; Appendix J compares the training checkpoints.

Qualitative coverage across domains and backbones. Figure 2 shows how prediction errors develop over H1–H3 in all four environments with both backbones. Within each row, methods receive the same context and recorded actions, making spatial and appearance differences directly visible.

A common visualization decoder renders DINO-WM patch predictions within each environment; NanoWM uses its native diffusion and variational autoencoder (VAE) outputs. The DINO-WM Rope row uses a separate 30-step replay at the original test seed and settings, with frozen model and decoder weights. Appendix K specifies sample selection, replay and decoder fitting.

4.3 Internal capacity and residual placement

Controlled variants. The prediction comparison adds both internal adaptation and private parameters. To distinguish these effects, we compare four variants on DINO-WM PushT. I/O-only uses two width-32 interface adapters (51,284 parameters). Wide I/O increases their width to 132 (209,084 parameters). FedWM keeps the width-32 interfaces and adds six width-32 FFN branches (209,036 parameters). Attention moves those six branches to attention sites with the same parameter count. Figure 3(a) summarizes the configurations.

Shared protocol. All variants use the same three data/shared-core blocks, each consisting of a fixed data partition and shared checkpoint. Within a block, the four clients share the same private optimizer and 20-epoch schedule, and the core remains frozen. The prespecified primary rule selects each client's best one-step validation checkpoint from epochs 11–20. Test error then averages equally over clients and horizons 1, 3 and 5. This controlled study uses a different checkpoint rule from the main PushT comparison and is reported separately in Appendix E.

Capacity and placement. FedWM lowers mean test error by 1.04% relative to I/O-only, with improvements in all three blocks. It also lowers mean error by 0.43% relative to Wide I/O, using 48 fewer private parameters and improving in two of three blocks. This near-capacity-matched comparison favors allocating some private capacity inside the predictor. Placement also matters: Attention has a slightly lower overall mean than FedWM, although FFN performs better in two of their three block-wise comparisons.

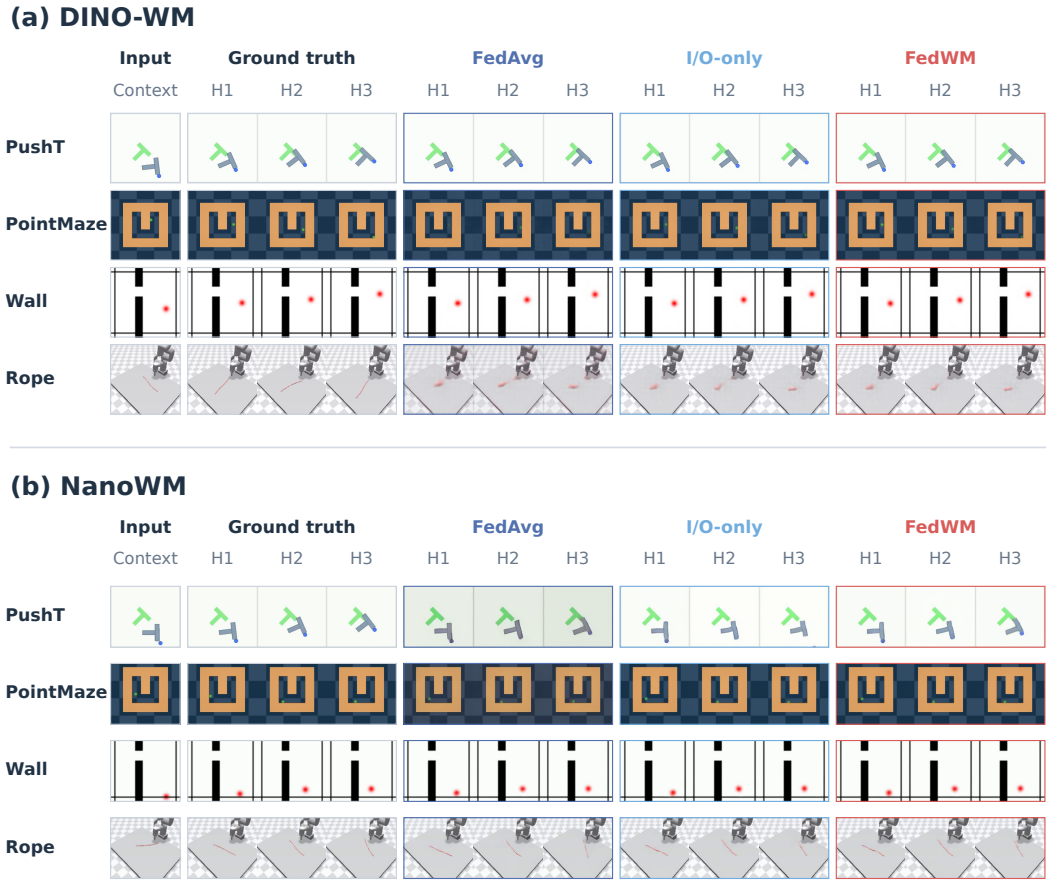


Figure 2: Visual prediction across four environments with (a) DINO-WM and (b) NanoWM. Each row shows the input, ground truth and three methods' predictions at H1–H3, using common context and actions. DINO-WM uses a shared visualization decoder per environment; its Rope sequence is a separate 30-step replay. Backbone-specific sampling and decoder protocols are detailed in Appendix K.

Stability and layer layouts. To examine the placement result across private training seeds, we run three additional initialization/shuffle seeds within each fixed data/shared-core block. Attention has lower error in seven of the nine pairs and in all three block means, averaging 0.081262 versus 0.081460 for FFN. A separate validation-only comparison of eight six-layer layouts ranks all-attention first. Appendix E reports these studies. The main FedWM results use the prespecified FFN implementation throughout.

4.4 Client-count expansion and new-client adaptation

More clients with the same data. To study smaller local datasets, we split each original DINO-WM profile into two clients, increasing K from four to eight while preserving the total training, validation and test data. FedWM maintains lower error than FedAvg and FedPer-style in all four eight-client environments (Figure 4). Its own errors increase by 32.35% on Wall and 16.67% on Rope, indicating sensitivity to smaller local partitions despite the relative advantage. Appendix F reports the full single-seed comparison.

Adapting to new clients. We next evaluate adaptation beyond the clients used for shared training. Three new PushT clients cover interpolation, extrapolation and a new visual condition, each with nested subsets of 5%, 10% and 20% of its training trajectories. The three adaptation strategies yield 27 runs, with target-side steps and data exposure matched within each client/fraction combination. FedWM-Adapt has lower error than FedAvg-FT in seven of nine combinations and I/O-only adaptation in eight (Figure 5). FedAvg-FT leads on the visual client at 10% and 20%; I/O-only leads on the

(a) Personalization configurations

Configuration	I/O width	FFN branches	Attention branches
I/O-only	32	-	-
Wide I/O	132	-	-
FedWM	32	6 x width 32	-
Attention	32	-	6 x width 32

Matched within each block: frozen core, data, 20 private epochs and selection rule.

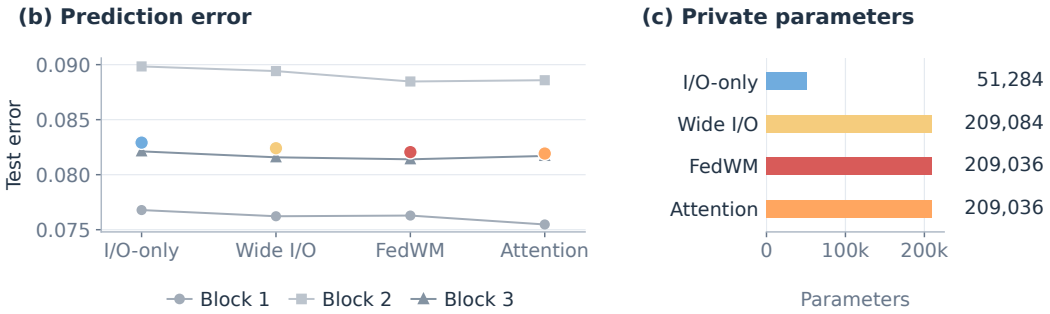


Figure 3: Internal personalization on DINO-WM PushT. (a) Adapter configurations: I/O denotes input/output adapters, width denotes bottleneck dimension, and dashes indicate absent branches. (b) Test prediction error: gray lines connect the three matched data/shared-core blocks, and colored circles indicate method means. (c) Private parameter counts. Variants use the same frozen core within each block, private training schedule and validation rule.

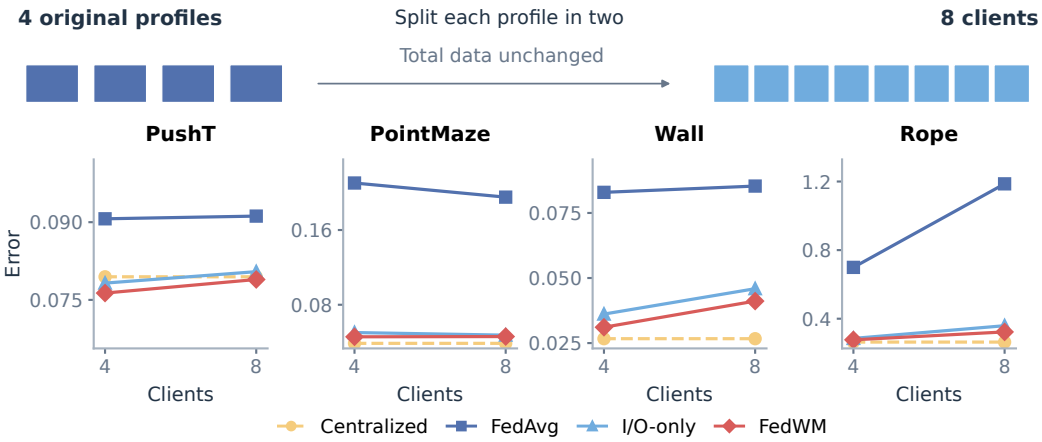


Figure 4: DINO-WM prediction error at four and eight clients with fixed total data. Each original profile is split into two clients. Results use one training seed and the same pooled-data Centralized reference at both client counts. Each environment retains its native metric.

extrapolation client at 5%. Both adapter methods start from the same source core, while FedAvg-FT uses its own source checkpoint. Appendix G details this single-seed study.

4.5 Closed-loop control and prediction–planning alignment

Lower rollout error is useful for control only if the model can guide action selection. We evaluate closed-loop PushT with checkpoints trained under a separate equal-total-step schedule. All methods use the same fixed action bounds and planner: a horizon decreasing from five to one, one-step receding-horizon execution, and CEM with 300 samples, 30 elites and 30 updates.

The validation bank contains four source tasks evaluated in four client profiles and with three planner seeds, giving 48 episodes per method. FedWM succeeds in 38 episodes and FedAvg in 32. Mean final position error is 15.384 versus 21.895 pixels, a paired difference of -6.511 pixels with a source-cluster bootstrap 95% interval of $[-13.017, -0.005]$. An episode-level McNemar diagnostic gives

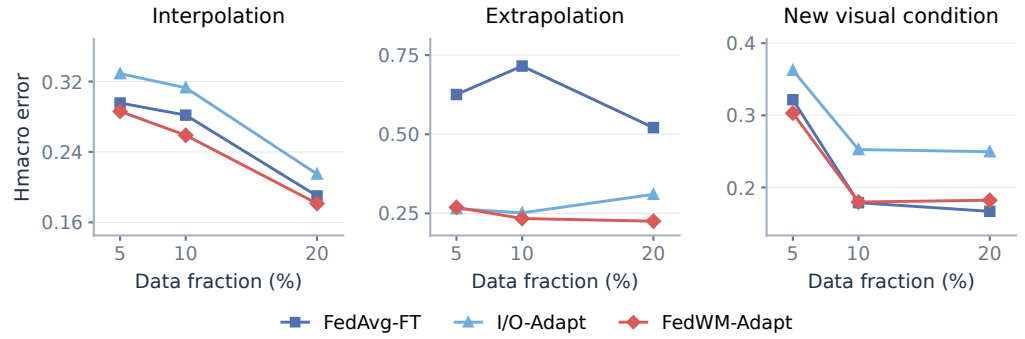


Figure 5: Adaptation to three new PushT clients using 10, 20 and 40 training trajectories (5%, 10% and 20%). Each point is one run with matched target-side steps and fixed-final-epoch evaluation. FedWM-Adapt and I/O-Adapt share a source core; FedAvg-FT uses its own source checkpoint.



Figure 6: Closed-loop PushT control. (a) Success rates on validation (four sources, four profiles, three planner seeds; 48 episodes per method) and frozen test (13 sources, four profiles; 52 episodes). (b) Execution traces for all four validation sources under D1V1 and planner seed 0. Traces connect block centers at successive replanning steps; hollow markers indicate final states and stars mark goals. Axes use image coordinates in pixels with equal aspect. S/F denotes success/failure under the joint position-and-angle criterion.

$p = 0.180$. Profiles and planner seeds are repeated measurements within only four source tasks, so success counts are reported descriptively.

The frozen test bank gives a different ordering: FedWM succeeds in 37 of 52 client–task units and FedAvg in 38, with FedWM's mean position error 2.459 pixels higher. Figure 6 pairs the results from both banks with execution traces for all four validation sources under a fixed profile and planner seed. Successes and failures occur in both directions, indicating that the control advantage depends on the task bank.

Prediction–planning alignment. We compare predicted costs with real-environment outcomes for common candidate action sequences on the frozen test bank. The analysis covers 208 method–

task combinations, 832 replay sequences and 3,328 model scores. The diagnostic rule assigns 26 of FedWM's 52 cases to model/latent-objective misranking and one to search failure, compared with 19 and zero for FedAvg. Mean rank correlations are high (0.9214 and 0.9234, respectively), yet errors among the candidates selected for execution can still affect control. The comparison therefore identifies a gap between average ranking agreement and decision-relevant accuracy within the evaluated candidate set (Appendix H).

4.6 Training budget and resource cost

FedWM's private DINO-WM PushT modules contain 209,036 parameters, or 1.04% of the 20,122,600-parameter shared predictor, compared with 51,284 for I/O-only. NanoWM uses 350,432 private adapter parameters, compared with 50,528 for I/O-only. These counts describe private adaptation; collaborative training also updates the shared model.

The equal-step PushT control protocol assigns each method 96,000 optimizer steps across four clients and matches total training-window exposure at 767,040. FedAvg uses all steps for full-model training. FedWM and FedPer-style divide them into 32,000 shared and 64,000 private steps. We account for optimizer steps and data use separately from floating-point operations (FLOPs) and elapsed time.

Private residuals require no communication during adaptation. Shared-stage traffic depends on the training schedule: the equal-step PushT comparison uses 60 shared rounds for FedAvg and 20 for the personalized methods, whereas NanoWM uses 100 for both. We report logical tensor traffic, excluding network overhead. Complete-run timing is unavailable for resumed NanoWM jobs because the process timer resets on resumption. Appendix I provides definitions and the complete 93-record resource ledger.

5 Discussion and conclusion

FedWM separates collaborative dynamics learning from client-specific residual adaptation, providing a common training procedure for predictive and diffusion-based world models. Across the eight backbone–environment settings, internal residuals improve prediction over interface-only adaptation at matched training steps. The near-capacity-matched DINO-WM PushT comparison also favors internal residuals over wider interfaces, suggesting that the allocation of private capacity matters alongside its size.

The strongest placement results on DINO-WM PushT come from attention residuals. Attention performs slightly better on average than the fixed FFN implementation and ranks first in the eight-layout validation study. This supports internal personalization without establishing FFN as the best insertion site. The client extensions address a different source of variation: FedWM retains its relative advantage with eight clients, but smaller local partitions increase absolute error in some environments. Further federation seeds and client profiles are needed to characterize this dependence and the low-data adaptation results.

Closed-loop control exposes a gap between accurate rollout prediction and reliable action selection. The validation bank favors FedWM, whereas the frozen test bank slightly favors FedAvg. Common-candidate diagnostics associate failures with errors in the model or latent objective near the selected action, despite high average rank correlation. These results make action ranking a concrete target for improving the connection between personalized prediction and planning.

FedWM leaves each client with a shared core and a small private parameter set, whose adaptation requires no communication. Its total resource cost still depends on shared training. The closest personalized-federation methods require faithful adaptations for a direct comparison, and FedAvg requires total-budget-matched runs in the older DINO-WM environments. Keeping trajectories local also provides no formal privacy guarantee. The present evidence establishes consistent prediction gains over the implemented interface-only comparator; their extension to reliable control depends on the quality of action rankings as well as state predictions.

REPRODUCIBILITY STATEMENT

The appendices specify client partitions, parameter ownership, optimization, checkpoint selection, evaluation units and resource accounting. Quantitative results are derived from saved experiment artifacts. Qualitative comparisons use fixed held-out samples and recorded actions; Appendix K describes the visualization decoders, reconstruction controls and separate Rope replay. The accompanying LaTeX package includes all manuscript figures. Figure-generation scripts and source manifests are archived in the working project. Public release of training data and checkpoints is subject to their distribution terms.

AI USE STATEMENT

Generative AI tools assisted with experimental design, implementation and debugging, and result organization and interpretation. Quantitative results were extracted from saved experiment artifacts, with selected values recomputed from raw evaluation files. AI-assisted work was checked against these records and the implementation. The authors take responsibility for the final content.

REFERENCES

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated Learning with Personalization Layers. arXiv preprint arXiv:1912.00818, 2019. URL https://arxiv.org/abs/1912.00818.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. arXiv preprint arXiv:2407.01392, 2024. URL https://arxiv.org/abs/2407.01392.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. arXiv preprint arXiv:2303.04137, 2023. URL https://arxiv.org/abs/2303.04137.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting Shared Representations for Personalized Federated Learning. In *International Conference on Machine Learning*, pp. 2089–2099, 2021. URL https://proceedings.mlr.press/v139/collins21a.html.

Sajjad Ghiasvand, Mahnoosh Alizadeh, and Ramtin Pedarsani. pFedMMA: Personalized Federated Fine-Tuning with Multi-Modal Adapter for Vision-Language Models. arXiv preprint arXiv:2507.05394, 2025. URL https://arxiv.org/abs/2507.05394.

Yuchao Hou. FedWorld: Scope-Aware Federation of Agent World Models. arXiv preprint arXiv:2608.01561, 2026. URL https://arxiv.org/abs/2608.01561.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-Efficient Transfer Learning for NLP. In *International Conference on Machine Learning*, pp. 2790–2799, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

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. arXiv preprint arXiv:2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Siqiao Huang, Partha Kaushik, Michael Chen, Hengkai Pan, Kaiwen Geng, Omar Chehab, Fernando Moreno-Pino, and Max Simchowitz. Nano World Models: A Minimalist Implementation of Future Video Prediction. arXiv preprint arXiv:2605.23993, 2026. URL https://arxiv.org/abs/2605.23993.

JB Lanier, Kyungmin Kim, Armin Karamzade, Yifei Liu, Ankita Sinha, Kat He, Davide Corsi, and Roy Fox. Adapting World Models with Latent-State Dynamics Residuals. In *Annual Learning for Dynamics and Control Conference*, pp. 117–144, 2026. URL https://proceedings.mlr.press/v331/lanier26a.html.

Hongxia Li, Zhongyi Cai, Jingya Wang, Jiangnan Tang, Weiping Ding, Chin-Teng Lin, and Ye Shi. FedTP: Federated Learning by Transformer Personalization. arXiv preprint arXiv:2211.01572, 2022. URL https://arxiv.org/abs/2211.01572.

Yao Lu, Jiaxing QI, Zhongzhi Luan, Shaohan Huang, Carol Fung, Hailong Yang, and Depei Qian. FDLoRA: Personalized Federated Learning of Large Language Model via Dual LoRA Tuning. arXiv preprint arXiv:2406.07925, 2024. URL https://arxiv.org/abs/2406.07925.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Jaehoon Oh, Sangmook Kim, and Se-Young Yun. FedBABU: Towards Enhanced Representation for Federated Image Classification. arXiv preprint arXiv:2106.06042, 2021. URL https://arxiv.org/abs/2106.06042.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. arXiv preprint arXiv:2304.07193, 2023. URL https://arxiv.org/abs/2304.07193.

Guangyu Sun, Matias Mendieta, Jun Luo, Shandong Wu, and Chen Chen. FedPerfix: Towards Partial Model Personalization of Vision Transformers in Federated Learning. arXiv preprint arXiv:2308.09160, 2023. URL https://arxiv.org/abs/2308.09160.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. arXiv preprint arXiv:1801.03924, 2018. URL https://arxiv.org/abs/1801.03924.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. arXiv preprint arXiv:2411.04983, 2024. URL https://arxiv.org/abs/2411.04983.

A IMPLEMENTATION AND PARAMETER OWNERSHIP

Algorithm 1 FedWM: shared training and private residual adaptation

Require: Client datasets $\{\mathcal{D}_k\}_{k=1}^{K}$, shared initialization $\theta^{(0)}$, optional private encoders ψ_k, frozen training/selection protocol.

for $r = 0, \ldots, R_g - 1$ **do**
  for each client k **do**
    Load $\theta^{(r)}$ and the client's current ψ_k.
    Run the prescribed local shared-stage updates with the backbone loss.
    Return only the shared tensors $\theta_k^{(r+1)}$; retain ψ_k locally.
  end for
  Aggregate $\theta^{(r+1)} \leftarrow \sum_k p_k \theta_k^{(r+1)}$.
  Evaluate the shared candidate on the designated validation split.
end for
Select $\theta^\star$ and the corresponding local encoder states by the frozen validation rule.
for each client k **do**
  Freeze $\theta^\star$; initialize private interface and internal residuals with zero output.
  Optimize only ϕ_k and any trainable local ψ_k for the prescribed private schedule.
  Select the private checkpoint by validation, or use the fixed final epoch when that protocol specifies it.
end for
return Shared $\theta^\star$ and client-local payloads $\{\phi_k^\star, \psi_k^\star\}$.

DINO-WM implementation. The input residual modifies observation features while preserving action slots. After positional embedding and dropout, each block applies the shared attention residual, followed by the shared FFN and a parallel private bottleneck. Both feed-forward branches receive the same output of the shared pre-FFN layer normalization. The predictor's final normalization precedes the output adapter. Each bottleneck contains two affine maps and a GELU activation; its output weights and bias are initialized to zero. During private training, the shared predictor is kept in evaluation mode, and only private residuals and environment-specific local encoders are optimized.

Let u_ℓ be the attention-residual output and $n_\ell = \mathrm{LN}_\ell(u_\ell)$. The implemented DINO-WM block is

$$h_{\ell+1} = u_\ell + \mathrm{FFN}_\ell(n_\ell) + U_{k,\ell}\,\mathrm{GELU}(D_{k,\ell} n_\ell + b^D_{k,\ell}) + b^U_{k,\ell}. \tag{4}$$

Layer normalization remains part of the shared core, and the six private internal bottlenecks have width 32. Zero initialization of their output maps preserves the original block computation at the start of adaptation.

NanoWM implementation. The input and action embedder wrappers compute $b(x) + r_k(b(x))$, where b is the frozen base module. Internal MLP wrappers compute $b(h) + r_k(h)$, and the final projection is adapted at its output. All bottlenecks use SiLU; the final bottleneck's width is capped by its output dimension. The FedPer-style variant uses only boundary wrappers, whereas FedWM also adapts internal MLPs. The private-training parameter mask selects the adapters and freezes the base model. Matching base-state hashes before and after adaptation verify that shared parameters remain unchanged.

Ownership differences. DINO-WM retains client-specific action/proprioception encoders for PointMaze, Wall and Rope. These encoders are trained locally in both stages and excluded from aggregation. PushT instead uses frozen visual/action representations with a shared predictor. NanoWM averages the base model with equal client weights. Equation 1 applies to the backbone-specific set of shared tensors in each case.

Structural checks. Implementation checks cover equivalence to the shared core at zero initialization, preservation of action slots, causal masking, frozen shared tensors, private-parameter gradients and updates, and save/load equivalence. Reused checkpoints were verified against shared-core hashes and training/validation counts.

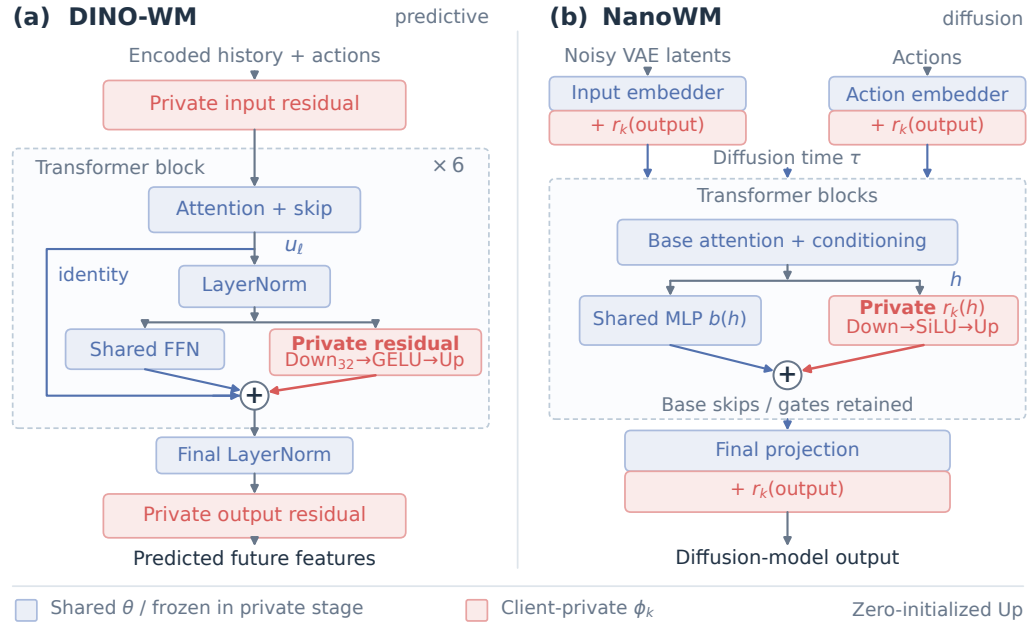


Figure 7: Backbone instantiations of FedWM. DINO-WM uses private input/output adapters and parallel GELU residuals at six FFN blocks. NanoWM uses SiLU residuals at its input/action embedders, internal MLPs and final projection. Shared modules retain the backbone's attention, conditioning and skip paths. Each backbone is trained in a separate federation.

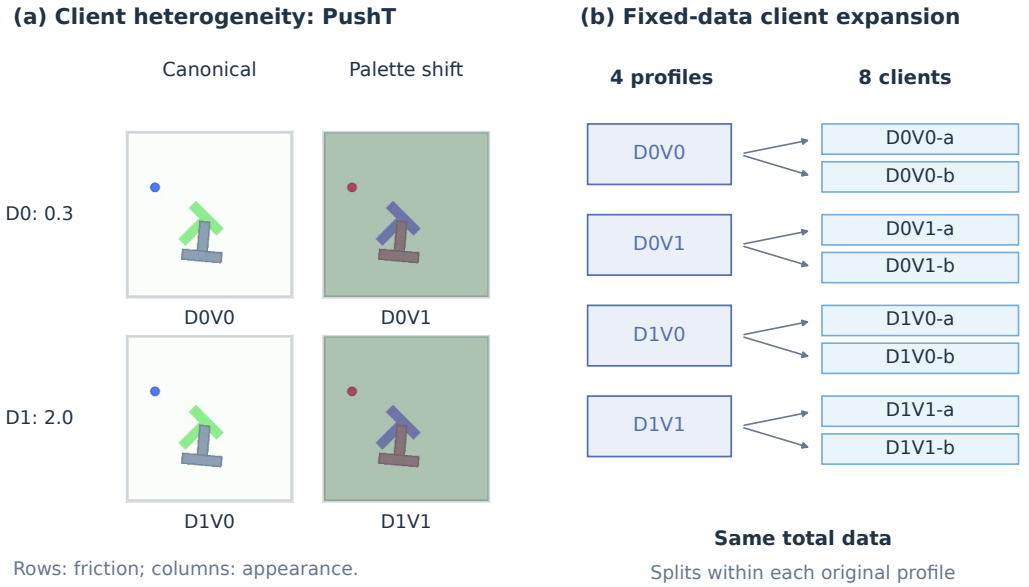


Figure 8: Client construction and fixed-data expansion. (a) PushT profiles cross contact friction and rendering, illustrated from a common initial configuration. (b) Each profile is split into two clients while preserving the total trajectory set.

B Data organization and experimental units

Each four-client federation crosses two factors. PushT varies contact friction and rendering; PointMaze varies damping and rendering; Rope varies stiffness and camera view. Wall varies wall and door locations. Subdivision preserves these parent profiles. Within each comparison, trajectory partitions, temporal windows and normalization statistics are fixed. Training and checkpoint selection use the designated training and validation splits, followed by held-out evaluation under the prespecified selection rule.

Table 3: Client profiles defined by two-by-two factor combinations. Wall groups use wall and door positions from trajectory metadata.

Environment	First factor	Second factor
PushT	Contact friction: 0.3 / 2.0	Canonical / palette-light transform
PointMaze	Damping: 1.0 / 1.35	Canonical / palette-light transform
Wall	Wall location: $\leq m_w$ / $> m_w$	Door location: $\leq m_o$ / $> m_o$
Rope	Stiffness: 0.6 / 0.9	Camera view: 1 / 2

For Wall, m_w and m_o are the medians of the corresponding trajectory metadata, and all four groups are balanced to the smallest group size. PointMaze's palette-light transform maps RGB channels to $(0.55B + 35,\ 0.70R + 18,\ 0.60G + 28)$ and clips the output to uint8. PushT varies contact friction while holding other physical settings fixed; its appearance intervention changes rendering. Rope fixes rope length at 2.5 and the configured z rotation at 10 degrees. Thus the physical and visual factors have environment-specific definitions.

Table 4: Federation-level trajectory counts. Four- and eight-client partitions use the same total training, validation and test data.

Environment	Train	Validation	Test	Eight-client construction
PushT	800	80	100	Two children per original profile
PointMaze	1,600	200	200	Two children per original profile
Wall	1,536	192	192	Two children per original profile
Rope	1,600	200	200	Two children per original profile

The client-count experiment uses training seed 260832. Splitting each profile into two clients reduces local data while preserving the total trajectory set and the four physical/visual profiles. Centralized training uses the pooled data once per environment; the same reference is plotted at $K = 4$ and $K = 8$.

The adaptation study uses three new clients with 200 training, 20 validation and 25 test trajectories each. Fractions of 5%, 10% and 20% form nested subsets of 10, 20 and 40 training trajectories. All methods are evaluated at the fixed final epoch, so target validation data are not used for checkpoint selection. Client differences also include changes in data collection and therefore represent distribution shifts rather than isolated single-factor interventions.

The main prediction study treats training seeds as independent repetitions. Capacity and placement studies use three fixed data/shared-core blocks, with private initialization and shuffle seeds nested within each block. In control, an episode is one method's execution on a client–task unit with a planner seed. Client profiles and planner repeats derived from the same source trajectory remain grouped in the source-cluster analysis.

C Training schedules, losses and checkpoint selection

DINO-WM. The main shared schedule uses 20 rounds, one local epoch per round and batch size eight. The predictor's learning rate is 10^{-5} during shared training and 10^{-4} for the private residuals, which are trained for 20 epochs. In PointMaze, Wall and Rope, this shared-plus-private schedule uses twice the total optimizer steps and sample exposure of the 20-round FedAvg schedule. FedWM and I/O-only use matched schedules. Centralized and Local references follow their environment-specific pooled and client-local schedules. PointMaze uses low-overlap training windows with stride five and 17 windows per trajectory; held-out rollouts use fixed starts. The resource ledger records the realized counts.

For PushT, validation compares the shared checkpoint at round 10 with the best checkpoint from rounds 11–20, then compares private epoch 10 with the best later checkpoint. The main FedWM results use epoch 10, selected by the validation client-macro rollout score. The main I/O-only results use epoch 10 for the first two seeds and the best later checkpoint for the third. The controlled capacity study instead selects each client's best one-step validation checkpoint from epochs 11–20 for every method. We report that study separately because its selection rule differs from the main PushT comparison.

NanoWM. NanoWM uses the S/2 configuration, deterministic VAE posterior-mode representations, batch size eight and 100 federated rounds. FedAvg receives 20 effective data passes. FedWM and I/O-only each receive ten shared and ten private passes. The learning rate is 10^{-4} and weight decay is 0.01. The action-input dimension is ten for PushT, PointMaze and Wall, and four for Rope. Shared and private checkpoints are selected by their validation criteria and evaluated with the fixed sampling configuration. Training retains the native diffusion prediction objective; LPIPS measures the resulting visual predictions.

Controlled structural comparisons. The ablations use AdamW with learning rate 10^{-4}, weight decay 0.01, batch size eight and 20 private epochs. Methods within each block share the same frozen core, data manifests and paired initialization/shuffle rule. Width-132 I/O adapters contain 209,084 private parameters, compared with 209,036 for six width-32 internal residuals plus the original interfaces. Moving the internal residuals from FFN to attention sites preserves this parameter count. Table 7 reports both best-later and epoch-10 checkpoints on validation and test data.

Metric aggregation. DINO-WM errors are first aggregated within client/horizon combinations using the environment's metric, then over clients and training seeds. The capacity study weights horizons 1, 3 and 5 and the four clients equally. NanoWM averages LPIPS over evaluated future horizons and clients. For three seed-level values e_1, e_2, e_3, sample dispersion is $s = \sqrt{\sum_i (e_i - \bar{e})^2/(3-1)}$. A paired-seed mean relative improvement is computed as $\frac{1}{3}\sum_i (b_i - e_i)/b_i$; this differs from the reduction between mean errors, $(\bar{b} - \bar{e})/\bar{b}$.

D Complete prediction results

Table 5 includes DINO-WM's client-local reference and NanoWM's shared-checkpoint reference, FedAvg-S. Figures 9 and 10 plot the main comparisons on their respective metric scales. The PushT Centralized reference uses validation-selected checkpoints from the 20-epoch run; Appendix J compares it with the earlier five-epoch result.

Table 5: Complete prediction results. Means and sample SDs summarize the three seeds listed in protocol-specific order. PushT uses its protocol-specific validation-selected checkpoints.

Backbone	Environment	Method	Mean ± SD	Seed 1	Seed 2	Seed 3
DINO	PushT	Centralized	0.080977 ± 0.004401	0.077043	0.085730	0.080159
DINO	PushT	Local	0.083956 ± 0.008580	0.077542	0.093703	0.080624
DINO	PushT	FedAvg	0.104947 ± 0.008375	0.098384	0.114380	0.102077
DINO	PushT	FedPer-style	0.083018 ± 0.006532	0.076983	0.089953	0.082118
DINO	PushT	FedWM	0.081536 ± 0.006541	0.075641	0.088572	0.080394
DINO	PointMaze	Centralized	0.042104 ± 0.002969	0.038698	0.043464	0.044149
DINO	PointMaze	Local	0.045419 ± 0.001542	0.043658	0.046071	0.046529
DINO	PointMaze	FedAvg	0.208387 ± 0.001664	0.210307	0.207388	0.207465
DINO	PointMaze	FedPer-style	0.046630 ± 0.004958	0.050263	0.040982	0.048645
DINO	PointMaze	FedWM	0.042862 ± 0.005543	0.045699	0.036475	0.046414
DINO	Wall	Centralized	0.027950 ± 0.001186	0.026626	0.028914	0.028311
DINO	Wall	Local	0.043291 ± 0.002109	0.042783	0.045609	0.041483
DINO	Wall	FedAvg	0.083645 ± 0.000903	0.083344	0.084660	0.082932
DINO	Wall	FedPer-style	0.036676 ± 0.001208	0.036136	0.038061	0.035832
DINO	Wall	FedWM	0.031773 ± 0.001198	0.031401	0.033113	0.030805
DINO	Rope	Centralized	0.265676 ± 0.004459	0.263684	0.270784	0.262560
DINO	Rope	Local	0.706855 ± 0.003737	0.706978	0.710529	0.703058
DINO	Rope	FedAvg	0.697878 ± 0.003713	0.699322	0.700654	0.693660
DINO	Rope	FedPer-style	0.285266 ± 0.005463	0.285099	0.290810	0.279888
DINO	Rope	FedWM	0.276758 ± 0.005415	0.276797	0.282154	0.271324
Nano	PushT	Centralized	0.049387 ± 0.001131	0.049203	0.048359	0.050598
Nano	PushT	FedWM	0.055401 ± 0.000683	0.056189	0.055036	0.054978
Nano	PushT	FedAvg	0.102661 ± 0.004997	0.101851	0.098119	0.108014
Nano	PushT	FedAvg-S	0.081428 ± 0.005228	0.086794	0.076350	0.081139
Nano	PushT	FedPer-style	0.059172 ± 0.000162	0.059277	0.058985	0.059253
Nano	PointMaze	Centralized	0.015944 ± 0.000980	0.017071	0.015470	0.015290
Nano	PointMaze	FedWM	0.015613 ± 0.000049	0.015590	0.015670	0.015579
Nano	PointMaze	FedAvg	0.032283 ± 0.002564	0.035243	0.030820	0.030786
Nano	PointMaze	FedAvg-S	0.029327 ± 0.001450	0.030956	0.028849	0.028177

Backbone	Environment	Method	Mean ± SD	Seed 1	Seed 2	Seed 3
Nano	PointMaze	FedPer-style	0.016029 ± 0.000141	0.015963	0.016191	0.015934
Nano	Wall	Centralized	0.006011 ± 0.000228	0.005751	0.006099	0.006182
Nano	Wall	FedWM	0.006897 ± 0.000093	0.006808	0.006887	0.006995
Nano	Wall	FedAvg	0.008206 ± 0.000305	0.008523	0.008178	0.007915
Nano	Wall	FedAvg-S	0.009495 ± 0.000131	0.009376	0.009473	0.009636
Nano	Wall	FedPer-style	0.007583 ± 0.000131	0.007433	0.007636	0.007679
Nano	Rope	Centralized	0.057164 ± 0.000833	0.057220	0.057967	0.056304
Nano	Rope	FedWM	0.069781 ± 0.001387	0.070657	0.070505	0.068181
Nano	Rope	FedAvg	0.075228 ± 0.002608	0.072274	0.076200	0.077211
Nano	Rope	FedAvg-S	0.079972 ± 0.000286	0.079692	0.079960	0.080263
Nano	Rope	FedPer-style	0.072769 ± 0.001911	0.073223	0.074411	0.070672

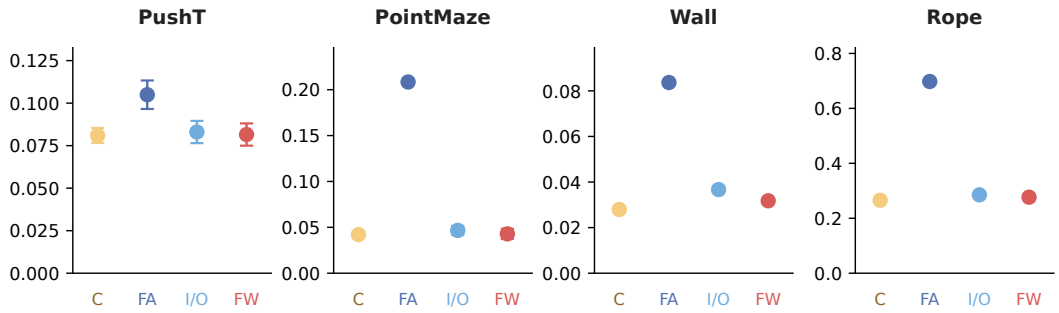


Figure 9: DINO-WM prediction error, reported as mean ± sample SD over three training seeds. C: Centralized; FA: FedAvg; I/O: FedPer-style; FW: FedWM. Environment-specific metrics and training budgets are defined in Section 4.

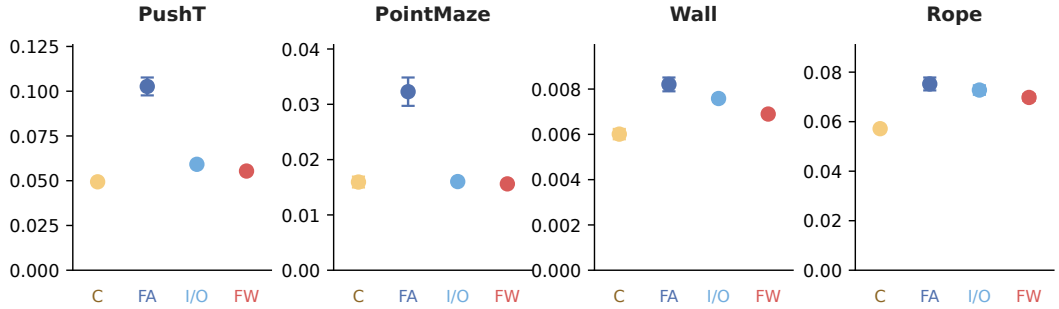


Figure 10: NanoWM LPIPS-Hmacro, reported as mean ± sample SD over three training seeds. Method abbreviations follow Figure 9.

E Capacity, placement and repeated private training

What I/O-only personalizes. I/O-only denotes the FedPer-style structural comparator in the figures. In DINO-WM PushT, its two private adapters act on non-action feature slots at the predictor boundaries, leaving action slots and Transformer blocks unchanged. In NanoWM, the boundary adapters wrap the visual/action embedders and final projection. The shared core is frozen during adaptation in both cases. Any local encoders used by the cross-environment implementation follow the same ownership rule in I/O-only and FedWM.

Four controlled structures. The DINO-WM PushT capacity study compares I/O-only (two width-32 boundary adapters; 51,284 parameters), Wide I/O (two width-132 adapters; 209,084 parameters), FedWM (width-32 boundaries and six width-32 FFN residuals; 209,036 parameters), and Attention (the same boundaries and six attention residuals; 209,036 parameters). FedWM versus I/O-only tests the combined addition of internal adaptation and capacity. Wide I/O tests whether approximately the same capacity is as effective at the boundaries, while Attention isolates internal placement at equal parameter count (Figure 3).

Matched training and the primary endpoint. Each of the three data/shared-core blocks supplies the same fixed core and four client datasets to all variants. Private training uses AdamW with learning rate 10^{-4}, weight decay 0.01, batch size eight and 20 epochs under the paired initialization/shuffle rule. The core remains frozen. Checkpoint selection minimizes each client's one-step validation loss over epochs 11–20; the primary test endpoint averages equally over four clients and horizons 1, 3 and 5. Updates and data exposure are matched, while FLOPs can differ with the adapted computation. Appendix A describes the implementation checks.

Capacity control. Wide I/O differs from FedWM by 48 parameters (0.02296%). Under the prespecified best-later rule, FedWM has lower mean test error and improves in two of three paired blocks. The comparison shows a modest advantage for internal adaptation at near-matched capacity on DINO-WM PushT.

Table 6: DINO-WM PushT capacity and placement controls (mean ± sample SD over three fixed data/shared-core blocks). Errors use the primary best-later selection protocol defined in Appendix E.

Method	Private parameters	Test error
I/O-only (FedPer-style)	51,284	0.082914 ± 0.006563
FedWM (FFN)	209,036	0.082051 ± 0.006118
Wide I/O	209,084	0.082406 ± 0.006635
Attention residual	209,036	0.081921 ± 0.006557

Table 7: Prespecified ablation analyses. Later denotes per-client validation selection among epochs 11–20; later-test is the primary endpoint.

Selection/split	Method	Mean ± SD	Block 1	Block 2	Block 3
later-validation	I/O-only	0.083748 ± 0.004869	0.080176	0.089294	0.081775
later-validation	FedWM	0.083878 ± 0.005730	0.078976	0.090177	0.082482
later-validation	Wide I/O	0.083153 ± 0.005246	0.078582	0.088881	0.081996
later-validation	Attention	0.083312 ± 0.005255	0.079046	0.089182	0.081710
later-test	I/O-only	0.082914 ± 0.006563	0.076786	0.089838	0.082118
later-test	FedWM	0.082051 ± 0.006118	0.076286	0.088470	0.081398
later-test	Wide I/O	0.082406 ± 0.006635	0.076223	0.089415	0.081579
later-test	Attention	0.081921 ± 0.006557	0.075476	0.088585	0.081702
epoch10-validation	I/O-only	0.083675 ± 0.004676	0.079568	0.088764	0.082694
epoch10-validation	FedWM	0.083102 ± 0.005582	0.078360	0.089254	0.081691
epoch10-validation	Wide I/O	0.083431 ± 0.004987	0.079917	0.089139	0.081238
epoch10-validation	Attention	0.082720 ± 0.004965	0.078517	0.088198	0.081444
epoch10-test	I/O-only	0.082989 ± 0.006538	0.076983	0.089953	0.082032
epoch10-test	FedWM	0.081536 ± 0.006541	0.075641	0.088572	0.080394
epoch10-test	Wide I/O	0.082508 ± 0.006187	0.076922	0.089158	0.081443
epoch10-test	Attention	0.081258 ± 0.006292	0.075646	0.088060	0.080067

Effect sizes and seed directions. Relative reduction here is $(\bar{e}_{\text{reference}} - \bar{e}_{\text{FedWM}})/\bar{e}_{\text{reference}}$. FedWM reduces mean error by 1.0408% relative to I/O-only and 0.4303% relative to Wide I/O. Attention has a slightly lower mean than FFN, despite FFN performing better in two of the three original blocks. Their approximate FFN-minus-Attention differences are $+0.000809$, -0.000116 and -0.000304: the first difference outweighs the other two. Sample SD summarizes variation across the three fixed blocks; these effect sizes are descriptive.

Selection sensitivity. Table 7 reports all four prespecified combinations of checkpoint rule and evaluation split. Best-later test error is the primary endpoint; epoch-10 and validation results are supporting analyses. With best-later selection, validation favors Wide I/O on mean error, while held-out test favors FFN. Each result remains attached to its original selection rule and evaluation split.

Private-seed repeats. The follow-up trains both placements with three additional private initialization/shuffle seeds in each block, yielding 18 runs and nine paired test comparisons. Training and stopping rules were fixed before execution. Table 8 lists all nine primary pairs. Across these repeats

and the three original pairs, FFN has lower error in four pairs and higher error in eight. This 12-pair summary describes repeated adaptation within three fixed blocks, not 12 independent federations.

Table 8: Nine paired private-seed repeats nested within three fixed data/shared-core blocks. Positive FFN-minus-Attention test-error differences favor Attention.

Block	Private repeat	FFN	Attention	FFN − Attention
1	1	0.0755764	0.0753739	+0.0002026
1	2	0.0752619	0.0750126	+0.0002493
1	3	0.0753891	0.0751568	+0.0002322
2	1	0.0887641	0.0878680	+0.0008961
2	2	0.0873252	0.0877774	-0.0004521
2	3	0.0897603	0.0891701	+0.0005902
3	1	0.0805370	0.0802721	+0.0002649
3	2	0.0801361	0.0800753	+0.0000608
3	3	0.0803921	0.0806561	-0.0002639

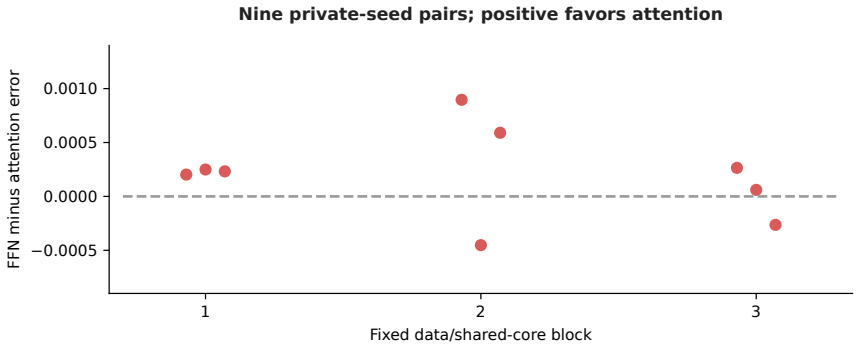


Figure 11: Nine paired private-seed repeats across three fixed data/shared-core blocks. Positive FFN-minus-Attention test-error differences favor Attention. Horizontal offsets distinguish the three private seeds within each block.

Eight-layout selection. We compare two uniform layouts with six prespecified mixed layouts. Every layer contains one width-32 internal residual, giving 209,036 private parameters and a 20-epoch private schedule in all cases. The six uniform-layout block results are reused, and 18 mixed-layout results are newly trained. Selection uses validation data only. A mixed layout would qualify for a separate confirmation if it improved mean error over the better uniform layout by at least 1%, improved in at least two of three blocks, and kept worst-client error within 105% of the reference in every block. All-attention ranks first, so no mixed layout meets this rule. The candidate set was fixed throughout, and no additional test evaluation was performed.

Table 9: Validation comparison of eight six-layer residual layouts. A and F denote attention and feed-forward sites in input-to-output order. Each configuration has one internal residual per layer and 209,036 private parameters.

Rank	Layout	Mean	Sample SD	Block 1	Block 2	Block 3
1	AAAAAA	0.0833123	0.0052546	0.0790455	0.0891815	0.0817098
2	AFAFAF	0.0833767	0.0054231	0.0786551	0.0892997	0.0821754
3	AAAFFF	0.0834594	0.0056377	0.0786873	0.0896800	0.0820107
4	FFAAFF	0.0835535	0.0059445	0.0784987	0.0901025	0.0820592
5	AAFFAA	0.0836757	0.0054580	0.0790423	0.0896922	0.0822926
6	FAFAFA	0.0837085	0.0056491	0.0789135	0.0899359	0.0822761
7	FFFAAA	0.0837607	0.0053509	0.0791842	0.0896439	0.0824541
8	FFFFFF	0.0838780	0.0057296	0.0789756	0.0901766	0.0824818

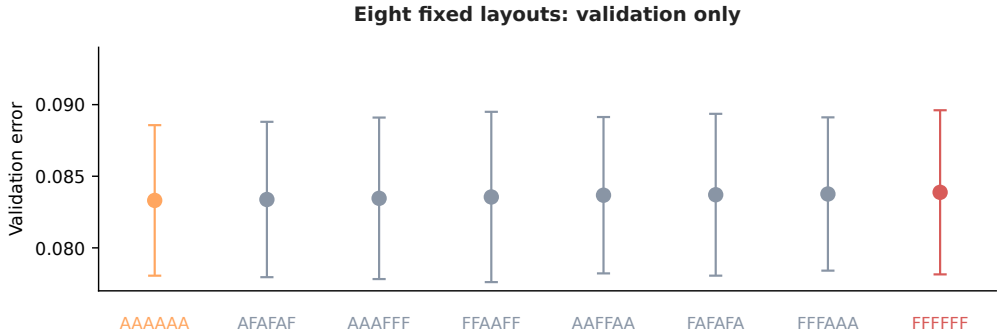


Figure 12: Validation errors for eight prespecified six-layer layouts. Points and error bars show the mean and sample SD over three fixed data/shared-core blocks.

Scope of the ablation evidence. The capacity, placement, repeat and layout studies cover DINO-WM PushT. In the other settings, the I/O-only comparison evaluates internal residuals together with their added parameters; NanoWM has no corresponding near-capacity-matched control. Input-only, output-only and internal-only variants, bottleneck width, and freezing versus joint core updates remain untested design choices.

F FIXED-DATA CLIENT-COUNT EXPERIMENT

Table 10 reports the complete DINO-WM client-count comparison. Subdivision preserves the pooled data and parent profiles under fixed training and checkpoint rules. The $K = 4$ checkpoints belong to this single-seed extension, separately from the three-seed main comparison. FedWM maintains its relative ranking at eight clients, although absolute error increases particularly in Wall and Rope.

Table 10: DINO-WM client-count comparison with fixed total data and training seed 260832. The Centralized run is shared across client counts. Relative change is $(E_8/E_4 - 1) \times 100$.

Environment	Method	$K = 4$	$K = 8$	Change
PushT	Centralized	0.079458	0.079458	+0.00%
PushT	FedAvg	0.090653	0.091174	+0.57%
PushT	FedPer-style	0.078239	0.080449	+2.82%
PushT	FedWM	0.076317	0.078939	+3.43%
PointMaze	Centralized	0.038698	0.038698	+0.00%
PointMaze	FedAvg	0.210307	0.195179	-7.19%
PointMaze	FedPer-style	0.050263	0.047467	-5.56%
PointMaze	FedWM	0.045699	0.045965	+0.58%
Wall	Centralized	0.026683	0.026683	+0.00%
Wall	FedAvg	0.082923	0.085330	+2.90%
Wall	FedPer-style	0.036136	0.045896	+27.01%
Wall	FedWM	0.031080	0.041133	+32.35%
Rope	Centralized	0.263684	0.263684	+0.00%
Rope	FedAvg	0.699322	1.186429	+69.65%
Rope	FedPer-style	0.285099	0.359147	+25.97%
Rope	FedWM	0.276797	0.322933	+16.67%

G NEW-CLIENT ADAPTATION

The new-client study uses one source-model training seed and 10 private adaptation epochs. The three data fractions require 220, 430 and 850 optimizer steps and expose the model to 1,700, 3,400 and 6,800 clips, respectively. These quantities are matched across methods within each client/fraction combination. FedAvg-FT updates 20,122,600 parameters, I/O-only adaptation updates 51,284, and FedWM-Adapt updates 209,036. Both adapter strategies use the same frozen source core; FedAvg-FT starts from its own checkpoint. Budget matching therefore applies to target adaptation, not source pretraining.

Table 11: New-client adaptation across nine comparison cells and 27 runs. Values are held-out Hmacro errors from one training seed, with target-side optimization steps matched within each cell.

Client	Fraction	Steps	FedAvg-FT	I/O-Adapt	FedWM-Adapt
Interpolation	5%	220	0.295624	0.328975	0.286109
Interpolation	10%	430	0.281830	0.312958	0.258945
Interpolation	20%	850	0.190000	0.214913	0.181444
Extrapolation	5%	220	0.625409	0.264701	0.268867
Extrapolation	10%	430	0.715330	0.251600	0.233646
Extrapolation	20%	850	0.520894	0.310014	0.225274
Visual	5%	220	0.321585	0.362340	0.303037
Visual	10%	430	0.178951	0.252555	0.179886
Visual	20%	850	0.166993	0.249531	0.182387

FedAvg-FT outperforms FedWM-Adapt on the visual client at 10% and 20%; I/O-only outperforms it on the extrapolation client at 5%. These single-seed comparisons assess adaptation from the available source models; a from-scratch baseline was not evaluated. The complete 27-run source table also reports training time, peak memory, serialized private checkpoint size and supported-operator FLOP estimates as separate resource measures.

H Closed-loop protocols and diagnostics

Task construction and planning. The validation bank selects four source episodes at the centers of motion-score quartiles, excluding the two sources used in the initial minimal validation. Selection uses ground-truth motion, without inspecting model outputs. Four client profiles and three planner seeds give 16 client–task units and 48 episodes per method, or 96 episodes across FedAvg and FedWM. The history and goal offsets correspond to ten and 25 primitive actions. Position error is the Euclidean norm of the concatenated agent and block position differences. Success requires this error below 20 pixels and circular block-angle error below $\pi/9$ radians. All methods use per-coordinate action bounds fixed to the observed training-split minima and maxima. Each CEM call uses 300 samples, 30 elites and 30 updates, with a horizon decreasing from five to one and one-step receding-horizon execution.

The validation episode/offset pairs are $(18243, 10)$, $(1835, 29)$, $(14999, 33)$ and $(15943, 12)$, representing low, mid-low, mid-high and high motion strata. Profiles and planner repeats are grouped by these four source trajectories. The frozen test bank contains 13 source tasks evaluated across four profiles, giving 52 client–task units per method.

Table 12: Closed-loop evaluation protocols, showing source tasks, client profiles, planner seeds and episode counts separately.

Protocol	FedAvg	FedPer-style	FedWM
Minimum validation, four units	1/4	2/4	3/4
Same units, three planner seeds	5/12	6/12	7/12
New validation, 16 units, three seeds	32/48	–	38/48
Frozen formal test, 52 units	38/52	36/52	37/52

Centralized succeeds in 36/52 frozen test episodes. The initial minimal validation additionally evaluated an upstream DINO-WM positive control (2/4) and random actions (0/80). These controls use the initial protocol, separately from the expanded validation bank.

Paired statistics. On the expanded validation bank, FedWM alone succeeds in ten episode pairs, FedAvg alone in four, both in 28, and neither in six. The exact McNemar value of 0.180 treats episodes as paired observations and does not adjust for source clustering. For position error, the bootstrap resamples the four source groups, preserving their associated client profiles and planner repeats. Figure 13 reports results by planner seed.

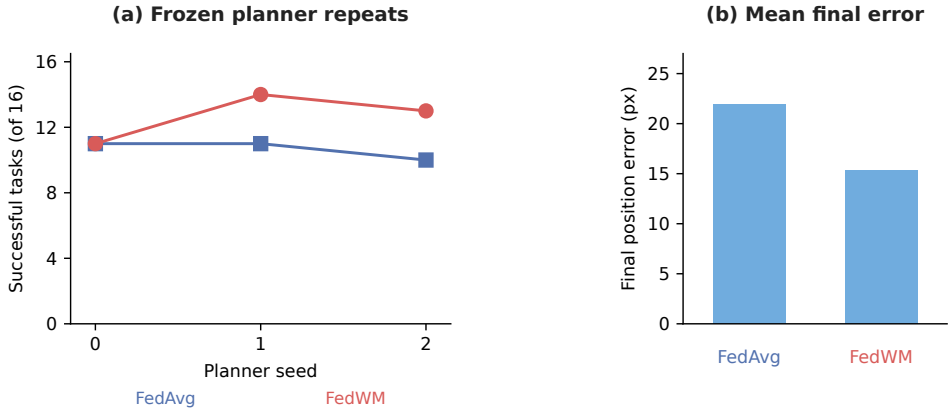


Figure 13: PushT validation performance. (a) Success counts by planner seed, each covering four sources across four client profiles. (b) Mean final position error across all 48 episodes per method. The source-cluster interval for the paired difference is reported in the text.

Formal-test diagnostics. On the frozen test bank, the FedWM-minus-FedAvg position-error difference has mean $+2.459$ pixels, median $+0.422$, and 10% trimmed mean $+1.037$. The source-cluster bootstrap interval for the mean is $[-3.171, 8.379]$ pixels. Leave-one-source-out means range from $+0.820$ to $+4.062$, so the direction of the mean difference persists when any one source is omitted. The test results are reported under the fixed model and planner selections.

For common candidate action sequences, we compare predicted costs with errors from real-environment replay. Model/latent-objective misranking denotes disagreement between predicted and realized candidate ordering; search failure is defined relative to the available candidates. These assignments are conditional on the candidate set and planning objective. Figure 14 reports the category counts for each method.

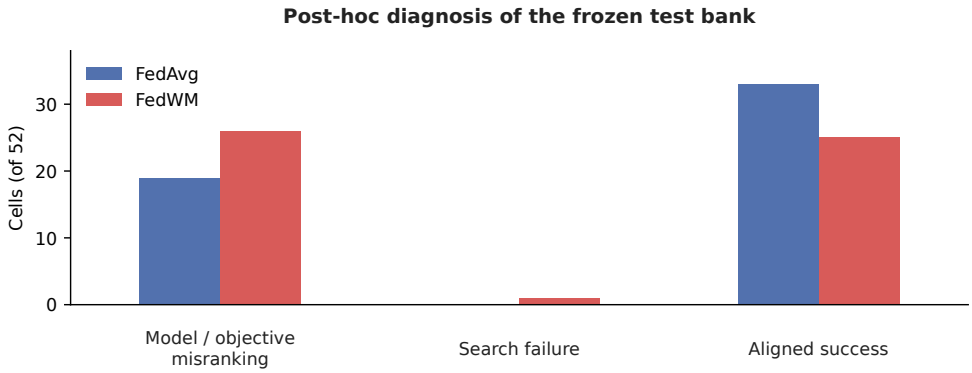


Figure 14: Failure-mode assignments from the common-candidate analysis of the frozen test bank, with 52 client–task units per method. Categories follow the diagnostic procedure in Appendix H.

Earlier PointMaze control study. An earlier PointMaze study evaluated Centralized, Local, FedAvg and FedWM on 12 client–task units each, totaling 48 episodes. Their success counts were 7, 6, 4 and 2, respectively; a separate random controller succeeded in 10/12. Under that protocol, FedWM performed worse than the learned-model comparators and random control.

I Resource accounting and complete budget ledger

The resource ledger reports optimizer steps, data exposure, trainable parameters, deployment storage, logical communication, elapsed time and supported-operator FLOPs separately. For a serialized shared tensor payload of B_s bytes, K participating clients and R rounds, bidirectional logical traffic is

$$\text{Traffic} = 2KRB_s. \tag{5}$$

This count covers shared-tensor uploads and broadcasts, excluding networking and secure-aggregation overhead. Private residual parameters incur no communication during adaptation.

Deployment storage includes the shared tensors and one client's private tensors, excluding fixed visual encoders/decoders, optimizer states and Python serialization overhead. DINO-WM's private counts include local action/proprioception encoders where used: 209,196 parameters for PointMaze, 209,176 for Wall and 209,266 for Rope, compared with 51,444, 51,424 and 51,514 for their I/O-only counterparts.

Elapsed time includes validation and varies with machine load. NanoWM resets its process timer on resumption, making the last invocation insufficient to recover cumulative training time. Profiler FLOP estimates cover supported operators. The ledger separately records executed training, including checkpoint evaluation, and the effective optimization steps represented by an earlier selected checkpoint.

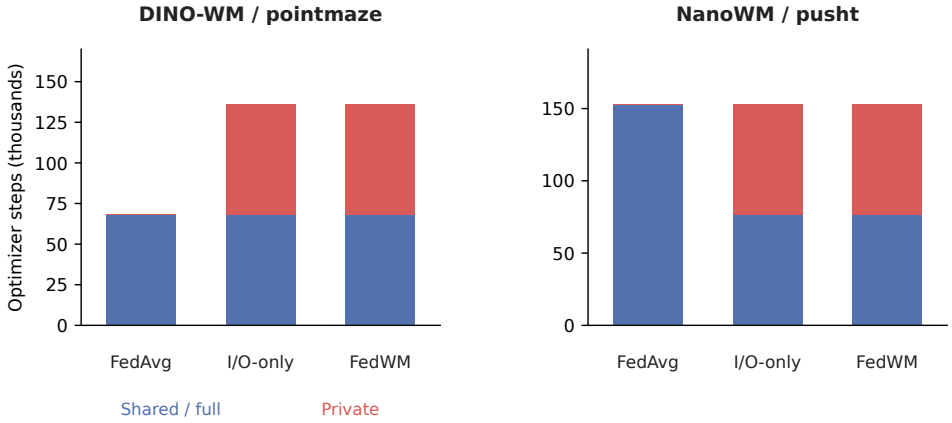


Figure 15: Shared/full-model and private optimization steps for seed 260832. DINO-WM PointMaze adds private adaptation after shared training; NanoWM PushT divides its near-matched total budget between the two stages.

Table 13: Cross-environment resource accounting (84 records). S/F and P denote shared/full-model and private optimizer steps. Exposure counts sampled windows/clips; traffic is logical bidirectional tensor payload in GiB. Seed suffixes 832/833/834 abbreviate 260832/260833/260834.

Backbone/environment	Seed	Method	S/F steps	P steps	Exposure	GiB
DINO/PointMaze	832	Centralized	68,000	0	544,000	0.00
DINO/PointMaze	832	FedAvg	68,000	0	544,000	11.99
DINO/PointMaze	832	I/O-only	68,000	68,000	1,088,000	11.99
DINO/PointMaze	832	FedWM	68,000	68,000	1,088,000	11.99
DINO/PointMaze	833	Centralized	68,000	0	544,000	0.00
DINO/PointMaze	833	FedAvg	68,000	0	544,000	11.99
DINO/PointMaze	833	I/O-only	68,000	68,000	1,088,000	11.99
DINO/PointMaze	833	FedWM	68,000	68,000	1,088,000	11.99
DINO/PointMaze	834	Centralized	68,000	0	544,000	0.00
DINO/PointMaze	834	FedAvg	68,000	0	544,000	11.99
DINO/PointMaze	834	I/O-only	68,000	68,000	1,088,000	11.99
DINO/PointMaze	834	FedWM	68,000	68,000	1,088,000	11.99
DINO/Wall	832	Centralized	34,560	0	276,480	0.00
DINO/Wall	832	FedAvg	34,560	0	276,480	11.90
DINO/Wall	832	I/O-only	34,560	34,560	552,960	11.90
DINO/Wall	832	FedWM	34,560	34,560	552,960	11.90
DINO/Wall	833	Centralized	34,560	0	276,480	0.00
DINO/Wall	833	FedAvg	34,560	0	276,480	11.90
DINO/Wall	833	I/O-only	34,560	34,560	552,960	11.90
DINO/Wall	833	FedWM	34,560	34,560	552,960	11.90
DINO/Wall	834	Centralized	34,560	0	276,480	0.00
DINO/Wall	834	FedAvg	34,560	0	276,480	11.90
DINO/Wall	834	I/O-only	34,560	34,560	552,960	11.90

Backbone/environment	Seed	Method	S/F steps	P steps	Exposure	GiB
DINO/Wall	834	FedWM	34,560	34,560	552,960	11.90
DINO/Rope	832	Centralized	4,000	0	32,000	0.00
DINO/Rope	832	FedAvg	4,000	0	32,000	11.99
DINO/Rope	832	I/O-only	4,000	4,000	64,000	11.99
DINO/Rope	832	FedWM	4,000	4,000	64,000	11.99
DINO/Rope	833	Centralized	4,000	0	32,000	0.00
DINO/Rope	833	FedAvg	4,000	0	32,000	11.99
DINO/Rope	833	I/O-only	4,000	4,000	64,000	11.99
DINO/Rope	833	FedWM	4,000	4,000	64,000	11.99
DINO/Rope	834	Centralized	4,000	0	32,000	0.00
DINO/Rope	834	FedAvg	4,000	0	32,000	11.99
DINO/Rope	834	I/O-only	4,000	4,000	64,000	11.99
DINO/Rope	834	FedWM	4,000	4,000	64,000	11.99
Nano/PushT	832	Centralized	152,880	0	1,223,040	0.00
Nano/PushT	832	FedAvg	152,880	0	1,223,040	118.56
Nano/PushT	832	I/O-only	76,440	76,440	1,223,040	118.56
Nano/PushT	832	FedWM	76,440	76,440	1,223,040	118.56
Nano/PushT	833	Centralized	151,700	0	1,213,600	0.00
Nano/PushT	833	FedAvg	151,700	0	1,213,600	118.56
Nano/PushT	833	I/O-only	75,852	75,852	1,213,632	118.56
Nano/PushT	833	FedWM	75,852	75,852	1,213,632	118.56
Nano/PushT	834	Centralized	152,480	0	1,219,840	0.00
Nano/PushT	834	FedAvg	152,480	0	1,219,840	118.56
Nano/PushT	834	I/O-only	76,240	76,240	1,219,840	118.56
Nano/PushT	834	FedWM	76,240	76,240	1,219,840	118.56
Nano/PointMaze	832	Centralized	324,000	0	2,592,000	0.00
Nano/PointMaze	832	FedAvg	324,000	0	2,592,000	118.56
Nano/PointMaze	832	I/O-only	162,000	162,000	2,592,000	118.56
Nano/PointMaze	832	FedWM	162,000	162,000	2,592,000	118.56
Nano/PointMaze	833	Centralized	324,000	0	2,592,000	0.00
Nano/PointMaze	833	FedAvg	324,000	0	2,592,000	118.56
Nano/PointMaze	833	I/O-only	162,000	162,000	2,592,000	118.56
Nano/PointMaze	833	FedWM	162,000	162,000	2,592,000	118.56
Nano/PointMaze	834	Centralized	324,000	0	2,592,000	0.00
Nano/PointMaze	834	FedAvg	324,000	0	2,592,000	118.56
Nano/PointMaze	834	I/O-only	162,000	162,000	2,592,000	118.56
Nano/PointMaze	834	FedWM	162,000	162,000	2,592,000	118.56
Nano/Wall	832	Centralized	119,040	0	952,320	0.00
Nano/Wall	832	FedAvg	119,040	0	952,320	118.56
Nano/Wall	832	I/O-only	59,520	59,520	952,320	118.56
Nano/Wall	832	FedWM	59,520	59,520	952,320	118.56
Nano/Wall	833	Centralized	119,040	0	952,320	0.00
Nano/Wall	833	FedAvg	119,040	0	952,320	118.56
Nano/Wall	833	I/O-only	59,520	59,520	952,320	118.56
Nano/Wall	833	FedWM	59,520	59,520	952,320	118.56
Nano/Wall	834	Centralized	119,040	0	952,320	0.00
Nano/Wall	834	FedAvg	119,040	0	952,320	118.56
Nano/Wall	834	I/O-only	59,520	59,520	952,320	118.56
Nano/Wall	834	FedWM	59,520	59,520	952,320	118.56
Nano/Rope	832	Centralized	68,000	0	544,000	0.00
Nano/Rope	832	FedAvg	68,000	0	544,000	118.56
Nano/Rope	832	I/O-only	34,000	34,000	544,000	118.56
Nano/Rope	832	FedWM	34,000	34,000	544,000	118.56
Nano/Rope	833	Centralized	68,000	0	544,000	0.00
Nano/Rope	833	FedAvg	68,000	0	544,000	118.56
Nano/Rope	833	I/O-only	34,000	34,000	544,000	118.56
Nano/Rope	833	FedWM	34,000	34,000	544,000	118.56
Nano/Rope	834	Centralized	68,000	0	544,000	0.00
Nano/Rope	834	FedAvg	68,000	0	544,000	118.56
Nano/Rope	834	I/O-only	34,000	34,000	544,000	118.56
Nano/Rope	834	FedWM	34,000	34,000	544,000	118.56

Table 14: PushT resources across four clients. B0 trains 20 private epochs; its main prediction checkpoint is epoch 10. B60 denotes control. Effective steps follow the intended schedule; overhead counts the unused B0 adapter job (dashes: unavailable).

Protocol	Run	Method	Shared	Private	Effective	Overhead	GiB
B0	1	FedWM	32000	32000	64000	1600	11.99
B0	1	I/O-only	32000	32000	64000	1600	11.99
B0	2	FedWM	31760	31760	63520	1588	11.99
B0	2	I/O-only	31760	31760	63520	1588	11.99
B0	3	FedWM	31920	31920	63840	1596	11.99
B0	3	I/O-only	31920	31920	63840	1596	11.99
WP-C	B60	FedAvg	96000	0	96000	–	35.98
WP-C	B60	I/O-only	32000	64000	96000	–	11.99
WP-C	B60	FedWM	32000	64000	96000	–	11.99

J TRAINING DURATION AND CENTRALIZED COMPARISONS

Extending PushT Centralized training from five to 20 epochs lowers mean test error from 0.084551 to 0.080977 at validation-selected checkpoints from epochs 18, 19 and 18. The epoch-10 mean is 0.082596. Figure 16 compares these checkpoints, showing the benefit of longer pooled training.

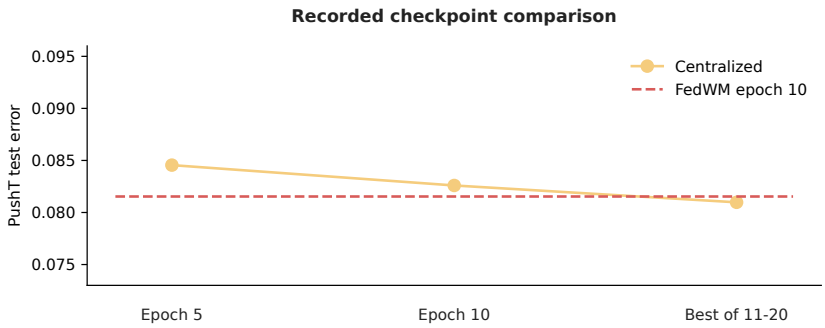


Figure 16: Centralized checkpoint comparison against the fixed FedWM main prediction reference. Test errors are evaluated at validation-selected checkpoints.

An earlier, smaller Rope dataset also illustrates the effect of training duration. Centralized initially used 800 steps, compared with 1,600 shared-plus-private steps for the personalized model. Extending Centralized to 1,600 steps produced mean error 0.471993 versus 0.612126 for the personalized model, reversing the comparison with the shorter run. The main Rope results use a separate full-data experiment.

K QUALITATIVE PREDICTION PROTOCOL AND DECODER CONTROLS

Figure 2 uses the sample selection, visualization decoders and temporal rollout procedures below. Figures 17, 19 and 20 show enlarged NanoWM sequences. Within each setting, all methods receive the same context and recorded actions.

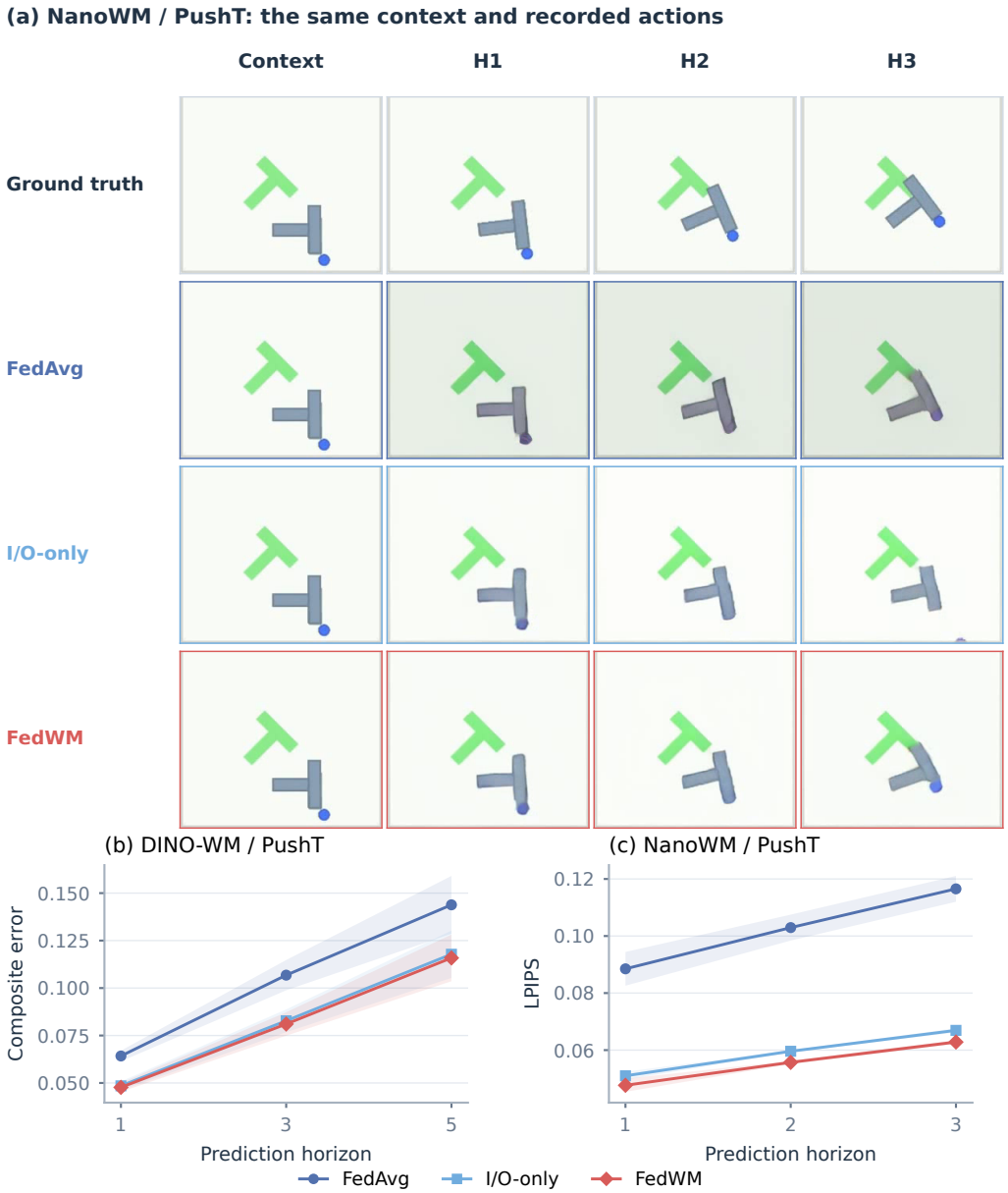


Figure 17: PushT temporal predictions and horizon-wise errors. (a) The NanoWM example from Figure 2, from shared context through H3. (b,c) DINO-WM and NanoWM prediction errors at their respective horizons. Lines and bands denote the mean and one sample SD over three training seeds, after client aggregation.

DINO-WM sample selection. PointMaze, Wall and Rope use training seed 260832 and the lowest numeric episode ID in the corresponding client's test manifest. PushT uses federation seed 2 and the first test branch. Its selected window is the first with a normalized RGB frame difference of at least 0.01, determined from ground truth alone. All samples are selected independently of method predictions. Each DINO-WM prediction step spans five raw frames.

DINO-WM multi-step export and Rope replay. The stacked figure uses frozen evaluation checkpoints and visualization decoders. PushT, PointMaze and Wall retain their original contexts and targets; their H1 latent features and decoded images were verified against the earlier exports. We then decode three successive autoregressive predictions conditioned on recorded actions. Rope's original test clip contains 20 observations. Its context indices are 0, 5 and 10, whereas H1–H3 require targets at 15, 20 and 25. We therefore record a separate 30-step clip at held-out seed 910002 and D0V0

settings with the unchanged simulator and action sampler. Simulator replay is not bitwise identical to the original clip: the entire Rope row, including input and H1, uses the new recording. All three methods receive its context and actions. The replay is used only for visualization; quantitative results, datasets, training and checkpoint selection remain unchanged. Source manifests record replay and checkpoint hashes.

Visualization-only decoding. One frozen decoder per environment maps DINO patch features to RGB and is shared across methods. PushT, PointMaze and Wall decoders are initialized from the original decoder and calibrated on training clips. Rope uses the same official decoder architecture with foreground-weighted reconstruction loss to reduce background dominance. All decoder fitting uses training clips and validation-based checkpoint selection. These decoders serve visualization only, outside world-model training and planning; their fitting cost is separate from the budgets in Section 4.6.

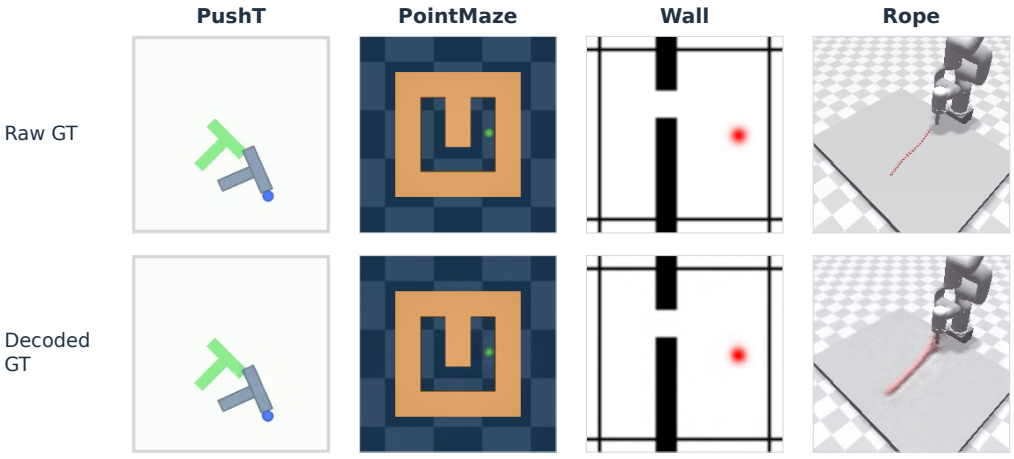


Figure 18: Decoder reconstructions of ground-truth features. Original images (top) are compared with reconstructions from their latent features (bottom), using each environment's frozen decoder. Rope uses the original short clip; Figure 2 uses a separate 30-step replay.

NanoWM sample selection. PushT uses seed 260833 and saved episode `fs02-b0220`. PointMaze, Wall and Rope use seed 260832 and the lexicographically first saved episode available for all three methods: `1019`, `112` and `D0V0_000002`, respectively. In the source videos, ground truth occupies the left half and prediction the right half. We remove the same 32-pixel label band from both halves for every method. Temporal panels extend these fixed samples from H1 through H3. The manifests record source hashes, frame hashes and ground-truth alignment.

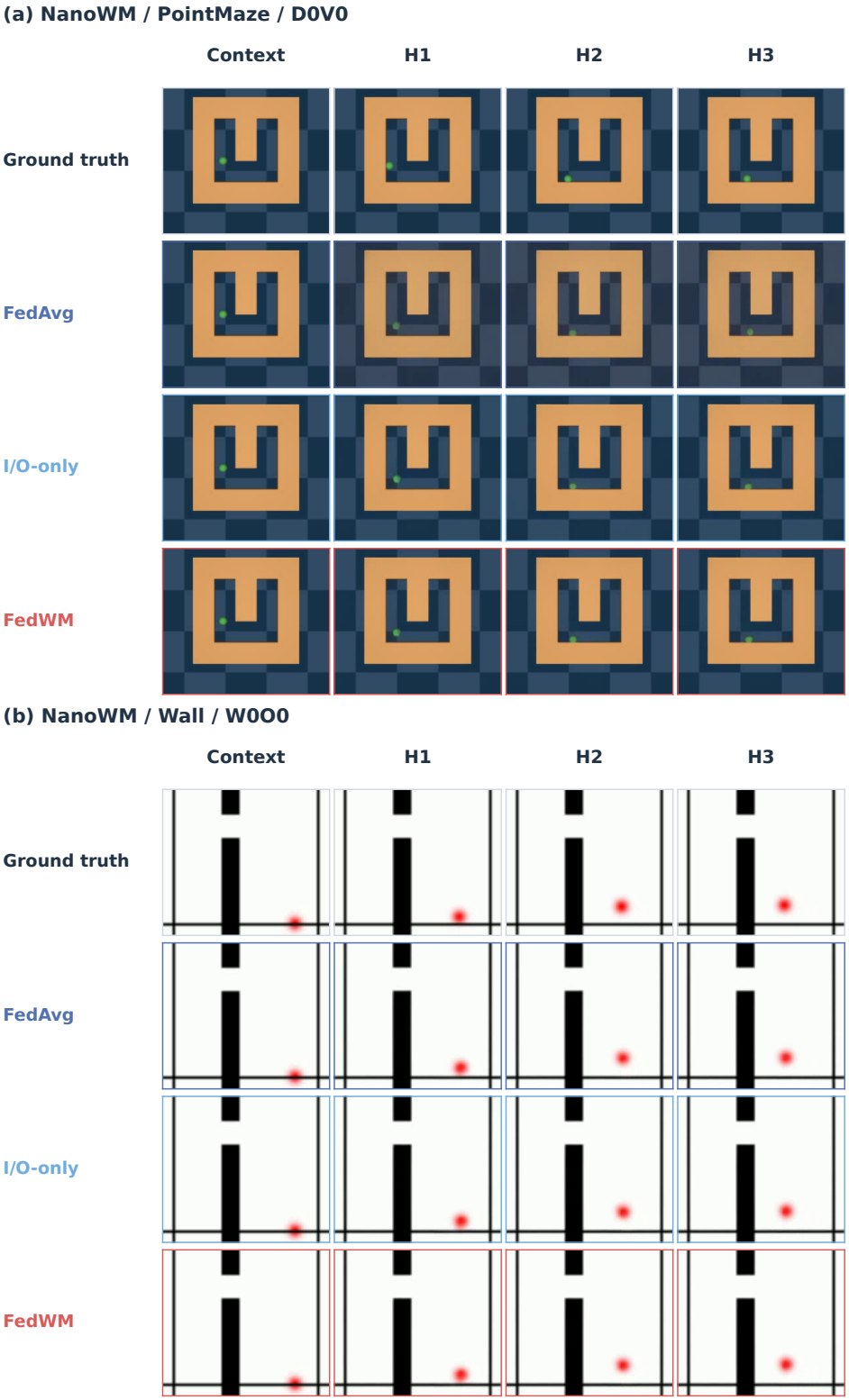


Figure 19: NanoWM rollouts in PointMaze and Wall. Rows compare ground truth, FedAvg, I/O-only and FedWM under common context and actions; columns show the context and H1–H3 predictions.

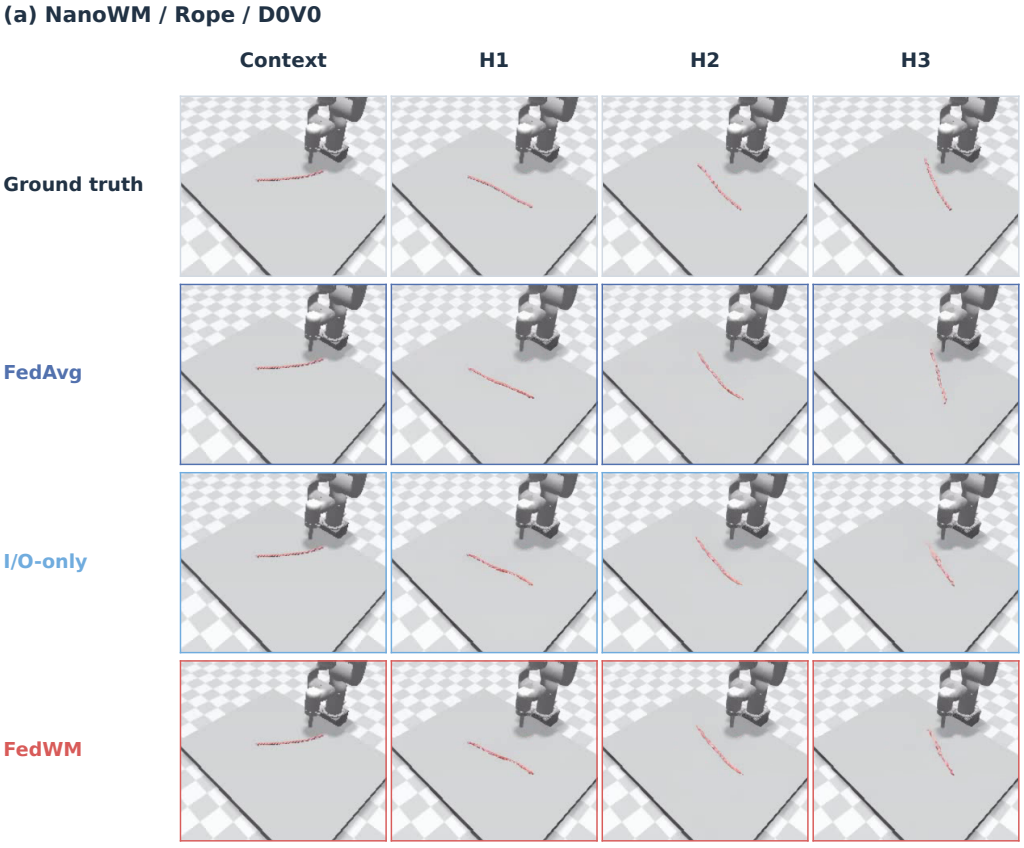


Figure 20: NanoWM rollouts in Rope. Ground truth and all methods share the same recorded episode, context and action sequence across H1–H3.

Horizon-wise summaries. Figure 17(b,c) summarizes three-seed PushT evaluations. DINO-WM averages within client/horizon combinations and then equally across clients; NanoWM uses client-macro LPIPS at H1, H2 and H3. Lines and bands show the mean and sample SD across training seeds. Each backbone uses its native prediction step. DINO-WM's H2/H3 visualizations come from separate recursive inference with frozen checkpoints.

Interpretation. The visualizations support comparisons within each environment and backbone, where inputs are matched. Across backbones, episodes are selected independently and prediction-step definitions and metrics differ. Aggregate performance is measured by the three-seed evaluations; the sequences illustrate how predictions evolve over time.