CONTINUALLY LEARNING A FUNCTIONAL MEMORY FROM TASK EXPERIENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Trained low-rank adapters (LoRAs) record what a model has learned from task experience. We ask how a stream of these adaptations can form a shared memory that preserves past solutions and supports new ones. The core question is which weight directions each adapter should contribute. We find that projecting an adapter into existing directions can reduce its update magnitude, and restoring that magnitude often recovers much of the lost task performance. A small number of task-dependent residual additions can recover further behaviour. Retained weight energy alone therefore does not determine behavioural recovery. These findings motivate **F**unctional **M**emory of **L**ow-rank subspaces (**FML**), which uses adapter predictions on unlabelled task inputs to guide the growth of a shared basis after correcting update scale. The admitted task is stored through coefficients and a global scale, allowing the original adapter and task inputs to be discarded. As the shared basis expands, previously admitted reconstructed updates remain unchanged. We evaluate continual retention and basis growth over a large SNI adapter stream and several ViT image-classification benchmarks, and further demonstrate downstream reuse through target-data-free LoRA initialization for subsequent fine-tuning and description-conditioned LoRA generation and adaptation within the functional basis, with evaluations on unseen SNI tasks and seven language benchmarks. Our findings demonstrate how structure retained from completed learning can support subsequent adaptation.

1 INTRODUCTION

What happens when a child who has learned to grasp a rattle is handed an unfamiliar cup? Its shape and weight call for adjustments, but prior grasping experience gives the child a head start in learning to handle it. In this paper, we ask how pretrained models can similarly draw on past learning experience when adapting to new tasks. These models acquire task-specific capabilities through training; with Low-Rank Adaptations (LoRA) (Hu et al., 2022), the resulting parameter changes are stored in separate adapters. Each adapter thus records the outcome of a completed learning experience. We study how a stream of independently trained adapters can form a shared functional memory that retains past solutions and provides reusable structure for future adaptation.

Building this memory requires deciding what to retain from each adapter. Since removing weight directions can also reduce the magnitude of the update, we first examine directional reduction while accounting for scale. On Super-NaturalInstructions (SNI) tasks with Mistral-7B, rescaling a substantially reduced update often recovers most of the original adapter's performance. We also find that the retained directions are reusable across tasks: directions accumulated from earlier adapters support later arrivals, while small residual additions recover further task-dependent behaviour. These findings motivate a functional criterion for memory construction: select directions by the behaviour they recover, and use the remaining behavioural discrepancy to guide what each new adapter contributes.

We introduce **Functional Memory of Low-rank subspaces (FML)**, an *optimization-free* method that continually consolidates a stream of independently trained LoRAs into a shared functional memory, illustrated in Figure 1. A small set of unlabelled task inputs characterises the behaviour of each incoming adapter, while the accumulated memory can later support adaptation to unseen tasks. Each incoming LoRA represents the outcome of a completed task adaptation. FML determines how much of an incoming adapter's behaviour is already reproduced by directions accumulated from previous

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.

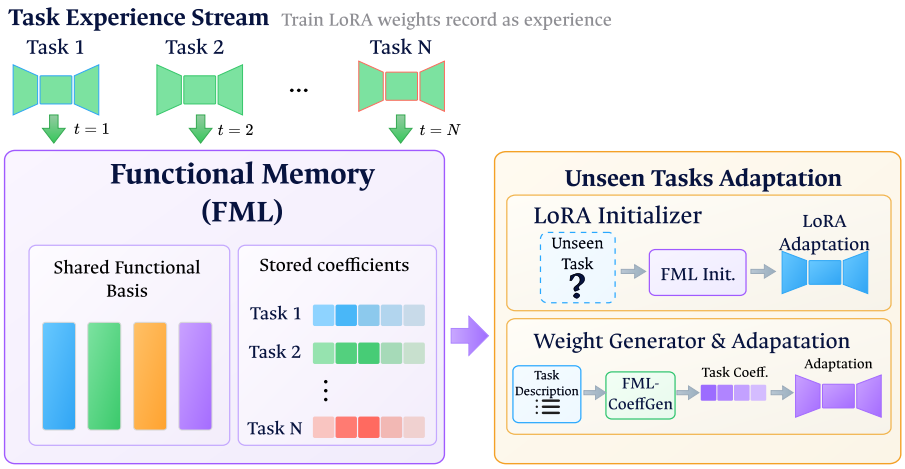


Figure 1: **FML turns accumulated task experience into a reusable functional memory.** The shared functional basis and stored coefficients support both target-data-free LoRA initialization and description-conditioned coefficient generation for unseen tasks.

tasks, applies global norm matching to that reconstruction, and admits only the residual candidates needed to recover the remaining behaviour. Once admitted, the task is represented by coefficients and a global scale in the evolving memory, allowing its original LoRA and task inputs to be discarded. This creates a memory that progressively accumulates shared structure from past experience while preserving the reconstructed updates of previously admitted tasks.

We study this memory through three complementary questions. First, can task experience be continually consolidated into a compact shared functional basis? Second, can previously admitted solutions remain preserved as further adaptations are incorporated? Finally, can the experience consolidated in this memory support adaptation to previously unseen tasks? These questions trace the role of the functional memory from consolidating task experience, to preserving previously learned solutions, and ultimately to reusing the resulting structure for new tasks.

Across a stream of 440 SNI adapters, structures accumulated from previous tasks increasingly explains newly arriving adaptations, allowing shared directions to be reused while limiting basis growth. Previously admitted solutions remain preserved as further adaptations expand the memory, which we verify across continual language and vision benchmarks. Most importantly, the resulting functional basis exhibits utility beyond consolidating seen tasks, providing useful structure for adaptation to unseen tasks outside the stream. Its final basis improves LoRA initialization on held-out SNI tasks, showing that directions from past experience provide a useful starting point for learning new adaptations. The same memory also provides an expressive space for description-conditioned LoRA generation: task descriptions can be mapped directly to coefficients in the frozen functional basis, showing competitive performance across seven downstream language benchmarks and supporting further adaptation within the functional basis span. These results show that completed task adaptations can be consolidated into a functional memory that preserves past solutions while providing reusable structure for future adaptation.

Our main contributions are:

- We show that task experience encoded in independently trained LoRAs contains shared functional structure. Much of an incoming adapter's behaviour can be explained by directions accumulated from previous tasks, with only a small amount of task-specific directions needed to be added.
- We introduce FML, an optimization-free continual method for consolidating a stream of trained LoRAs into a shared functional memory. FML expands the memory only when existing directions are insufficient to recover an incoming adapter's behaviour, while preserving the representations of previously admitted tasks as the basis grows.
- We demonstrate that the resulting memory is useful beyond preserving past adaptations. Its accumulated directions improve target-data-free LoRA initialization on unseen SNI tasks, while the same frozen functional basis supports description-conditioned coefficient generation and subsequent adaptation on seven downstream language benchmarks.

2 Related Work

Shared LoRA subspaces and Weight-space learning. Recent work suggests that independently adapted models reuse low-dimensional structure in weight space. UWSH identifies shared spectral subspaces across trained models and LoRA adapters (Kaushik et al., 2025a), while EigenLoRAx extracts a principal adapter subspace and learns only coefficients for new tasks (Kaushik et al., 2025b). SHARE extends this idea to continual learning by repeatedly updating SVD-based LoRA bases and analytically re-expressing previous tasks in the evolving space (Kaushik et al., 2026). These methods identify reusable structure through parameter-space geometry and reconstruction. FML instead expands its basis according to whether the current space reproduces an incoming adapter's *behavior*, admitting residual directions only when functional capture is insufficient.

LoRA initialization and generation. Several methods improve adaptation through informative LoRA parameterisations. PiSSA initializes from principal singular components of the pretrained weights (Meng et al., 2024), LoRA-GA uses target-task gradients to align the initial low-rank update with full fine-tuning (Wang et al., 2024), and EVA derives initial directions and rank allocation from target-task activation variance (Paischer et al., 2025). Our downstream initialization instead reuses functionally selected directions accumulated from previously trained adapters and requires no target inputs at initialization. Hypernetwork methods such as Hypter (Ye & Ren, 2021) and HINT (Ivison et al., 2023) generate adaptation parameters from task information, while Text-to-LoRA directly generates LoRA weights from natural-language task descriptions (Charakorn et al., 2025). Our coefficient generator instead predicts only coefficients in the functional basis discovered by FML and is trained on stored coefficient representations rather than full LoRA weights.

Functional sensitivity of parameter directions. Several methods use functional information to assess the importance of parameter directions. AdaLoRA uses loss sensitivity to allocate rank during adaptation (Zhang et al., 2023). TARA-Merging measures the task-loss influence of individual LoRA directions for reweighting during merging (Jeong et al., 2026). Matveev et al. (2026) studies functionally important directions through behavioural curvature. These approaches primarily quantify how strongly individual directions affect an objective. To our knowledge, FML is the first to characterise candidate LoRA directions by their task-conditioned effects on model outputs and use this functional structure to discover reusable directions across independently trained adapters.

3 Functional Memory of Low-rank subspaces

FML converts a stream of trained LoRAs into a *functional memory* by determining, for each incoming adapter, which of its behaviour is already represented by the current basis and which residual directions must be added. Figure 2 illustrates this process. We project the incoming LoRA into the current basis and norm-match the resulting update before evaluating its *functional capture*. If capture is insufficient, the remaining directions are represented by task-conditioned functional sketches and ranked by unexplained functional variation using column pivoting QR factorisation (CPQR) (Chan, 1987), where only as many directions as needed to meet the functional admission criterion are appended. The admitted task is then stored through its coefficients in the shared basis, allowing its full LoRA and task inputs to be discarded.

3.1 Motivation

Our method is motivated by a simple question: how much of a trained LoRA is actually required to preserve its task behavior? We find that much of an adapter's function can remain even after discarding the majority of its weight directions, provided the projected update is restored to an appropriate scale. After this correction, the remaining functional difference is often explained by only a small number of task-specific directions, although which directions are required varies across tasks. This suggests that preserving a LoRA does not require reconstructing all of its weights equally well – much of its behavior can be represented through shared directions, while only the functionally necessary residual directions need to be retained. These findings directly motivate FML, where we represent each LoRA using the shared basis, rescales the resulting update, and adds only the residual directions needed to recover its behavior. The experiments on the motivation is provided in Appendix A

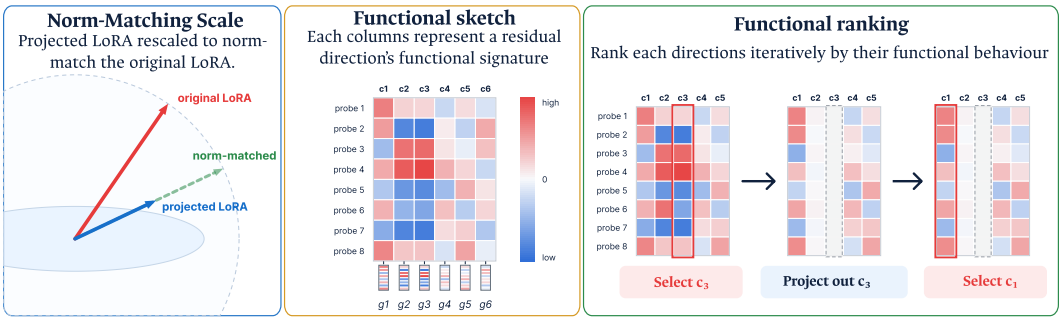


Figure 2: **Core components of FML.** The figure illustrates norm-matched projection, construction of functional sketches for residual directions, and CPQR-based functional ranking.

3.2 Continual Setting

At step t, a LoRA arrives with a small set of unlabelled task inputs $\mathcal{X}_t$. All LoRAs share the same base model. For each adapted module m, FML maintains input and output bases $U_{A,m} \in \mathbb{R}^{d_{\text{in}} \times K_{A,m}}$ and $U_{B,m} \in \mathbb{R}^{d_{\text{out}} \times K_{B,m}}$. Previous adapters and their task data are discarded after admission.

3.3 Functional Capture

We first project the incoming LoRA $\Delta W_{t,m} = B_{t,m}A_{t,m}$ into the current shared basis. This projection can be written in low-rank form as $\widetilde{\Delta W}_{t,m} = \widetilde{B}_{t,m}\widetilde{A}_{t,m}$, where $\widetilde{B}_{t,m} = U_{B,m}U_{B,m}^{\top}B_{t,m}$ and $\widetilde{A}_{t,m} = A_{t,m}U_{A,m}U_{A,m}^{\top}$. Equivalently, $\widetilde{\Delta W}_{t,m} = U_{B,m}U_{B,m}^{\top}\Delta W_{t,m}U_{A,m}U_{A,m}^{\top}$. As motivated in Section 3.1, projection can preserve useful directions while reducing the magnitude of the update. As such, we restore its magnitude using a global norm-matching factor α_t and evaluate the reconstruction $\widehat{\Delta W}_{t,m} = \alpha_t \widetilde{\Delta W}_{t,m}$, as illustrated in Figure 2.

We measure how much of the incoming LoRA's behaviour is preserved by this reconstruction using *functional capture*. We use Kullback–Leibler (KL) divergence between the predictive distributions of the incoming LoRA and its reconstruction as a modality-independent measure of behavioural discrepancy:

$$C_t = 1 - \frac{\mathbb{E}_{x \in \mathcal{X}_t} D_{\text{KL}}(p_t(\cdot \mid x) \,\|\, \widehat{p}_t(\cdot \mid x))}{\mathbb{E}_{x \in \mathcal{X}_t} D_{\text{KL}}(p_t(\cdot \mid x) \,\|\, p_0(\cdot \mid x))}.$$

Here p_t, $\widehat{p}_t$, and p_0 denote predictions from the incoming LoRA, the norm-matched reconstruction, and the frozen base model, respectively. Thus $C_t = 1$ denotes perfect functional recovery, while $C_t = 0$ corresponds to the base-model reference. If the projected reconstruction already satisfies the functional admission criterion, no residual directions are added. Otherwise, the remaining discrepancy is addressed through the functionally ranked residual directions described in Section 3.4.

After admission, let $\overline{\Delta W}_{t,m}$ denote the reconstruction for task t. The task is retained through coefficient factors $Z_{B,t,m}, Z_{A,t,m}$ and its scale α_t, with $G_{t,m} = Z_{B,t,m}Z_{A,t,m}^{\top} = U_{B,m}^{\top}\overline{\Delta W}_{t,m}U_{A,m}$. Together with the shared bases, these coefficients and scales form the *functional memory*.

3.4 Functional Ranking of Residual Directions

If the current reconstruction does not satisfy the functional admission criterion, the current basis does not yet reproduce the incoming adapter sufficiently well. We consider only the part of the LoRA that lies outside the current subspace. After removing the existing projection, the residual update is decomposed into rank-one candidates $\Delta W_{t,m}^{\text{res}} = \sum_{j=1}^{J_m} \sigma_j u_j v_j^{\top}$, where each candidate can independently expand the functional basis.

Rather than ranking these candidates by σ_j or their weight energy, we rank them by their task-conditioned effect on the model outputs. We use random output projections to efficiently sketch these effects without materialising the full output Jacobian (Varga et al., 2018; Hoffman et al., 2019). Let $z_t(x; c)$ denote the model logits when residual candidate j is multiplied by $\exp(c_j)$.

Each probe q is associated with an unlabelled task input x_q. For random probe r_q, let D_{x_q} denote the number of output-logit coordinates included in the projection. We define

$$\phi_q(c) = \frac{r_q^\top z_t(x_q; c)}{\sqrt{D_{x_q}}}, \qquad g_{qj} = \left.\frac{\partial \phi_q(c)}{\partial c_j}\right|_{c=0},$$

where g_{qj} measures the sensitivity of probe q to residual candidate j. Collecting Q probe measurements over the task inputs gives the *functional sketch*, illustrated in Figure 2,

$$H_t \in \mathbb{R}^{Q \times \sum_m J_m},$$

whose columns $h_j = (g_{1j}, \ldots, g_{Qj})^\top$ are the task-conditioned functional signatures of the residual candidates. Candidates with similar signatures induce similar changes in the model outputs, and a single vector–Jacobian product computes one row of H_t for all candidates simultaneously.

We apply CPQR to H_t to obtain a rank-revealing ordering illustrated in Figure 2 of residual candidates (Chan, 1987). Directions whose functional signatures are already explained by previously selected ones are treated as redundant, while those contributing large unexplained variation are ranked higher. The resulting ordering determines *which* directions are considered for admission, while functional capture determines how many are required.

3.5 Functional Threshold and Admission

The functional ranking determines the order of residual directions but not how many should be admitted. Starting from the highest-ranked candidates, FML progressively enlarges the reconstruction. At each candidate set size k, we globally rescale the reconstructed adapter to match the Frobenius norm of the original LoRA and evaluate its functional capture $C_t(k)$ in the full model.

Directions are added monotonically until the functional admission criterion is met. We require both high functional capture and sufficient remaining KL discrepancy. The full admission criterion and its numerical thresholds are given in Appendix B.6. The selected directions are orthogonalized against the existing functional bases and appended during the stream. Previously admitted basis directions are left unchanged, so later arrivals introduce no additional representation error for earlier tasks.

After the stream, we apply *functional re-ranking* using the stored task coefficients. We reconstruct the accumulated task representations, canonicalise them by SVD, and order directions by their cross-task usage, placing the most reusable directions first. Figure 8 shows that this produces substantially more concentrated shared structure than the weight-based subspace. The re-ranked bases together with the stored task coefficients and scales form the *functional memory*.

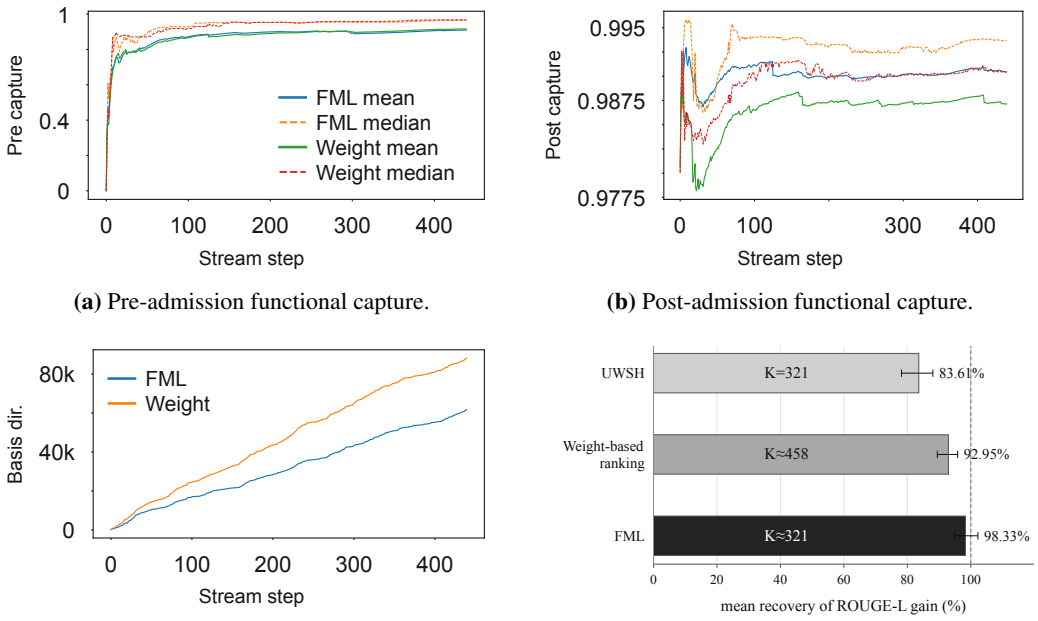
4 Experiments

Model and task streams. For language, we use Mistral-7B-Instruct-v0.2 (Jiang et al., 2023) and rank-16 Lots-of-LoRAs (LoL) adapters (Brüel-Gabrielsson et al., 2024) trained on SNI (Wang et al., 2022). Construction uses a 440-adapter stream, while temporal retention follows SHARE's 22-task stream (Kaushik et al., 2026). For vision, we follow SHARE's ViT-B/16 (Dosovitskiy et al., 2021) continual classification protocol on CIFAR-100 (Krizhevsky, 2009), Caltech-100 (Fei-Fei et al., 2007), and Flowers-100 (Nilsback & Zisserman, 2008).

Comparisons and measurements. We compare FML with a weight-ranked admission variant using the same norm matching and KL criterion, and with offline UWSH (Kaushik et al., 2025a) constructed from the LoL collection at matched capacity. We report functional capture, total basis width, and downstream task performance. The 440-task SNI comparison uses 32 examples per task.

4.1 Discovering a shared functional basis

We first examine whether the functional directions can be discovered continually across tasks. Figure 3a shows that earlier adapters quickly provide useful directions for later tasks: the running median pre-admission capture reaches 0.929 after 100 LoRAs and 0.964 over the complete stream. Since capture is measured before adding the incoming adapter's residual directions, this demonstrates reuse of previously discovered structure rather than recovery through task-specific expansion.



(a) Pre-admission functional capture.

(b) Post-admission functional capture.

(c) Cumulative basis growth.

(d) Reconstruction vs baselines.

Figure 3: Continual construction of the shared functional basis over the 440-task SNI adapter stream. We report pre- and post-admission functional capture, cumulative basis growth, and reconstruction performance against weight-ranked and offline UWSH subspaces.

Table 1: Continual retention on vision. CL methods results are reported from SHARE. At $\tau = 0.98$, FML recovers 98–100% of full-LoRA accuracy.

Method	# Params	CIFAR-100		Caltech-100		Flowers-100	
		Acc.	Forg.↓	Acc.	Forg.↓	Acc.	Forg.↓
LoRA (r=8)	3.0M	95.45	–	99.41	–	99.88	–
EWC	86M	59.60 ±1.3	23.73 ±3.0	57.96 ±1.8	29.26 ±1.6	69.79 ±1.8	24.65 ±0.1
LwF	86M	68.22 ±1.6	25.14 ±2.4	63.26 ±1.4	33.42 ±1.9	71.78 ±2.0	15.44 ±1.5
L2P	86M	83.05 ±1.0	2.12 ±0.1	89.34 ±1.8	6.93 ±0.4	94.53 ±1.2	7.21 ±0.2
DualPrompt	0.35M	84.77 ±0.7	1.40 ±0.3	91.52 ±0.9	4.84 ±0.1	95.25 ±0.8	5.60 ±0.4
CODA-P	0.42M	86.25 ±0.7	1.16 ±0.2	91.44 ±0.3	4.03 ±2.1	97.02 ±0.4	4.67 ±0.3
DAP	0.19M	94.05 ±1.2	0.41 ±0.1	97.23 ±0.3	2.52 ±0.8	96.49 ±0.1	2.28 ±1.0
EASE	0.28M	87.81 ±0.8	0.52 ±0.0	96.54 ±0.5	3.48 ±1.5	97.53 ±0.2	5.40 ±1.0
SHARE	0.10M	94.20 ±0.9	0.40 ±0.4	97.70 ±1.1	2.18 ±2.4	97.90 ±0.7	2.33 ±1.1
FML (Ours)	0.44–0.66M	**94.32** ±2.4	**0.00** ±0.0	**98.84** ±1.3	**0.00** ±0.0	**99.79** ±0.2	**0.00** ±0.0

By selecting residual directions according to their functional contribution, FML requires 29.8% fewer directions than weight-ranked construction while attaining slightly higher mean post-admission capture (Figure 3b, 3c). The stored representations also exhibit more concentrated cross-task usage after re-ranking (Figure 8). These results show that functional admission discovers a smaller shared basis that already explains much of the behaviour of subsequent adapters, leaving only the remaining task-specific information to be added.

We also compare task-performance recovery with the offline UWSH subspace constructed from all 440 LoRAs jointly (Figure 3d). At the same basis capacity, offline UWSH recovers 83.61% of the original LoRA's ROUGE-L, compared with 98.33% for FML. The weight-ranked admission baseline reaches 92.95% while growing to a larger basis under the same functional admission criterion.

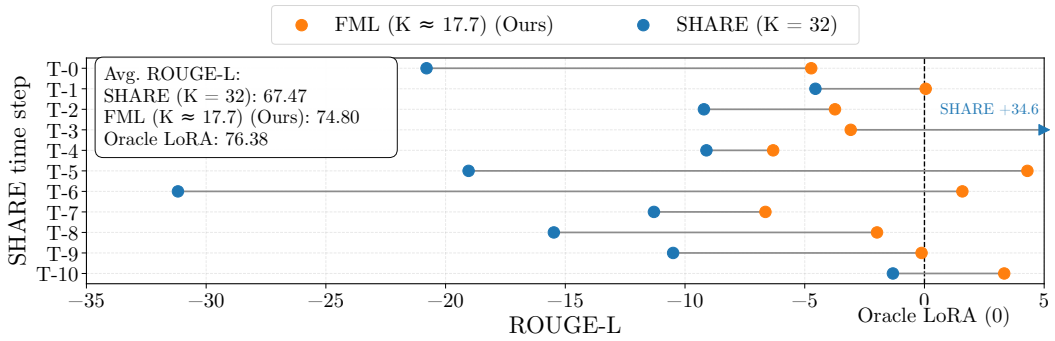


Figure 4: **FML better preserves task-specific LoRA performance than SHARE.** Across 22 tasks over 11 time steps, FML retains substantially more Oracle-LoRA performance than SHARE.

Table 2: Target-data-free LoRA initialization. Final test ROUGE-L after 256 optimizer steps (two epochs over 1,024 examples) on five held-out SNI tasks, over three seeds. All methods use total rank $\sum_m r_m = 384$ ($r = 4$). **Bold**/underlined values mark the best/second-best task average scores.

Method	Target inputs at init.	T039	T290	T391	T442	T1598	Avg. ↑
Fully-trained LoRA ($r = 16$)	0	58.8	93.7	93.4	67.8	51.6	73.1
Uniform Rank							
LoRA ($r = 4$)	0	68.4±1.7	90.5±0.3	85.7±3.9	65.6±0.3	48.8±0.9	71.8±1.0
PiSSA	0	57.6±3.9	90.4±0.0	55.6±48.2	64.9±0.7	48.7±1.3	63.4±9.2
UWSH (EigenLoRAx-Init)	0	69.4±2.8	90.6±0.2	83.9±1.8	**66.3**±0.6	**49.8**±0.6	72.0±0.9
FML-Init	0	**71.0**±3.3	90.4±0.0	84.9±2.9	64.3±0.6	49.4±1.0	72.0±1.1
Adaptive Rank							
EVA	64	55.5±1.6	90.1±0.4	83.3±0.1	63.6±1.7	48.6±1.1	68.2±0.3
FML-Init (global-rank)	0	69.5±4.7	**90.9**±1.1	**91.2**±1.6	65.5±0.2	**49.8**±0.6	**73.4**±1.0

4.2 FML AS A CONTINUAL LEARNER

We next test whether the growing functional basis preserves admitted task representations as the memory expands. On SHARE's original 22-task language stream, FML retains an average ROUGE-L of 74.80 versus 76.38 for the independently trained task-specific LoRAs, while using substantially fewer basis directions than SHARE and introducing no post-admission forgetting (Figure 4). We additionally test the order-sensitivity of the stream in Appendix D.

We observe the same preservation behaviour on the vision benchmarks. Across CIFAR-100, Caltech-100, and Flowers-100, FML retains 98.8–99.9% of the independently trained LoRA accuracy while maintaining zero forgetting (Table 1). This shows that the continual functional basis preserves admitted solutions across both language generation and image classification.

4.3 ADAPTATION TO UNSEEN TASKS

Having established that the functional basis continually preserves previously seen solutions, we next ask whether the same accumulated structure can serve as a prior for *new* tasks. We consider two strategies of using the functional memory for an initialization to unseen tasks. The first directly reuses the top directions that were useful across the training stream to initialize a new LoRA. The second learns a lightweight description-conditioned coefficient generator that maps task descriptions to coefficients within the functional basis. These two settings test different levels of prior knowledge, the former captures *task-agnostic* directions that transfer broadly across tasks, whereas the latter attempts to select a *task-specific* point in the same functional space before observing any task inputs.

FML as a target-data-free LoRA initialization. We first consider supervised adaptation to unseen tasks, where labelled examples become available after initialization. Rather than initializing

Table 3: Zero-shot performance (%) on seven downstream language benchmarks following the Text-to-LoRA evaluation protocol (Charakorn et al., 2025). We also compare the inputs required to train each methods: training task's data, trained task LoRA weights, and access to the base model.

		Training inputs			Downstream accuracy (%)							
Method	Optim. steps	Task data	LoRA weights	Base model	ArcC	ArcE	BQ	HS	OQA	PIQA	WG	Avg. ↑
Mistral-7B-Instruct	–	–	–	–	65.4	77.8	71.6	49.7	54.2	72.8	45.0	62.4
Prepending task desc.	–	–	–	–	72.0	85.8	67.6	58.9	63.4	77.9	59.0	69.2
3-shot ICL	–	–	–	–	72.1	85.9	71.7	59.0	66.2	76.2	58.0	69.9
Average LoRA	–	✓	✓	✓	70.7	84.4	75.4	59.9	59.0	78.0	54.3	68.8
Multi-task LoRA	–	✓	×	✓	76.2	88.3	85.5	65.2	68.0	81.8	62.4	75.3
Arrow Routing	–	✓	✓	✓	60.9	86.2	**87.6**	**80.8**	48.6	**83.0**	**68.5**	73.7
Hyperdecoders	–	✓	×	✓	76.6	88.5	83.9	65.2	**76.6**	81.3	64.9	76.7
T2L-L (SFT)	1M	✓	×	✓	**77.5**	**88.9**	85.0	66.5	75.5	82.1	64.2	**77.1**
FML-CoeffGen (Recon)	**10k**	×	×	×	72.2	87.2	86.2	60.7	71.6	80.0	56.9	73.5

a new LoRA randomly, we initialize its factors using directions from the final functional basis and then optimize the LoRA normally. Under the same total rank budget $\sum_m r_m = 384$, we use the final reranked functional basis to initialize the new LoRA. For *uniform-rank* initialization, we take the top-ranked directions within each module using a fixed rank $r_m = 4$. For *adaptive-rank* initialization, we instead take the top-ranked directions globally across the model, allowing the reranked functional basis to determine how the rank budget is distributed across modules.

As shown in Table 2, the globally allocated FML initialization achieves the strongest average performance among the initialization methods, reaching 73.4 ROUGE-L under the same adaptation budget. Unlike activation-aware initialization methods such as EVA, FML-Init, both the directions and their allocation are obtained entirely from previous task experience, without requiring any preprocessing or statistics computed from target-task inputs before the fine-tuning step.

Description-conditioned generation and within basis adaptation. The previous experiment uses the same task-agnostic prior for every unseen task. We next ask whether task descriptions can identify a more targeted point in the functional basis. We observe that related task categories occupy correlated regions of coefficient space (Figure 5), suggesting that task-specific structure is retained in the stored coefficients. This motivates FML-CoeffGen, a hypernetwork that predicts the coefficients of an unseen task directly from its natural-language description. In contrast to prior works such as T2L, Hyperdecoders and Arrow, our generator requires no storage of the original LoRAs, task inputs, or passes through the base model during generator training. Its training set is constructed entirely from information already retained by FML – the functional basis, stored task coefficients, and their associated task descriptions, with a simple objective of reconstructing the task coefficient pattern and scale. We find that predicting these two components separately is substantially more effective than directly predicting the scaled coefficients, allowing the generator to preserve the predicted functional direction while recovering an appropriate update magnitude. This makes FML-CoeffGen particularly well suited to a continual-learning setting – as new tasks are incorporated into the functional memory, their coefficients and descriptions can simply be added to the generator-training set, allowing the hypernetwork to be cheaply fine-tuned or retrained without replaying the original task data or loading the base model.

Despite its lightweight training, FML-CoeffGen achieves competitive zero-shot performance relative to substantially more optimization-intensive description-conditioned generators such as T2L (Table 3). This reflects the structured coefficient space learned by FML, which provides a constrained target for the prediction while retaining task- specific variation. The generated coefficients also provide an effective starting point for subsequent specialization within this space. Directly tuning only the coefficients in the frozen functional basis yields immediate gains from limited supervision (Table 4), while full adaptation converges within 288–800 coefficient updates for all tasks, within the first epoch. The fully adapted solutions reach 85.0 average accuracy and outperform independently trained task-specific LoRAs on 5/7 benchmarks despite introducing no new basis di-

Table 4: Direct coefficient adaptation within the functional basis. All FML rows are initialized from FML-CoeffGen predictions and update only coefficients. Few-shot rows show adaptation from limited target supervision, while the fully adapted row reports the validation-selected endpoint.

Method	ArcC	ArcE	BQ	HS	OQA	PIQA	WG	Avg. ↑
Efficient adaptation from coefficients generated by FML-CoeffGen								
FML-CoeffGen	72.2	87.2	86.2	60.7	71.6	80.0	56.9	73.5
FML-CoeffGen + 8-shot	72.5	87.5	86.2	62.7	73.0	81.1	56.8	74.3
FML-CoeffGen + 64-shot	71.8	87.3	86.0	67.2	69.0	82.3	60.8	74.9
FML-CoeffGen + 128-shot	74.5	88.4	86.2	71.6	78.6	80.6	58.6	76.9
FML-CoeffGen + 256-shot	77.1	88.9	85.7	78.0	75.6	82.8	62.5	78.6
FML-CoeffGen + 512-shot	76.1	87.6	86.9	81.3	78.6	83.7	66.0	80.0
Full adaptation								
FML-CoeffGen + fully adapted	**77.1**	**90.5**	**89.5**	90.9	82.6	**86.5**	**77.8**	**85.0**
Task-specific LoRAs (Oracle)	76.6	89.9	89.4	**92.6**	**85.0**	69.9	51.1	79.2

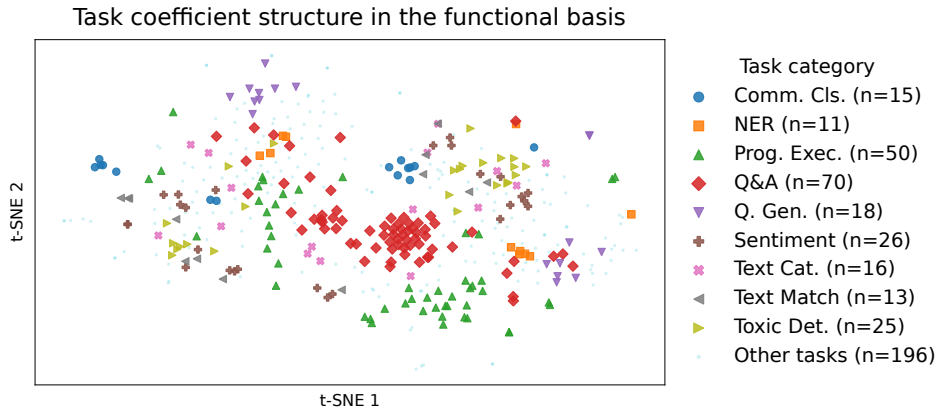


Figure 5: *Task structure emerges in the functional basis.* This t-SNE visualization of pairwise cosine similarities between vectorized SNI task coefficients shows coherent clustering by category, demonstrating that the shared basis preserves task-specific information.

rections. These results show that the functional basis supports more than preservation of the adapters used to construct it, allowing new task solutions to be predicted and refined within the same space.

5 Conclusion

We introduced FML, an optimization-free method that consolidates a stream of trained LoRAs into a shared functional memory. By combining analytic rescaling with function-guided admission, FML reuses structure accumulated from past adaptations while adding only the functional directions that remain missing. On the 440-adapter SNI stream, FML reduces basis growth relative to weight-ranked admission while maintaining strong functional recovery. Separate language and vision experiments show that previously admitted solutions remain preserved as the memory expands. The resulting basis also supports new tasks, improving LoRA initialization on held-out SNI tasks and enabling description-conditioned coefficient generation and subsequent adaptation within the frozen basis. With full coefficient adaptation, these solutions reach 85.0 average accuracy across 7 downstream language benchmarks, surpassing the independently trained task-specific LoRA on average. This suggests that task experience consolidated from completed adaptations can be carried forward both to preserve past solutions and support incoming unseen tasks.

AI DISCLOSURE

We used large language models (LLMs) to assist with text editing, code debugging, and the generation of boilerplate code and dataset retrieval scripts. Following our manual literature review, we also employed LLMs as an additional check for potential plagiarism. No LLMs were used for research ideation, formulation, experiment design, data analysis or result interpretation at any stage of this work.

REPRODUCIBILITY STATEMENT

We provide the methodological and experimental details required to reproduce our results in the appendix. Appendix B gives the complete functional admission criterion, numerical edge cases, and algorithms for FML construction, functional ranking, and admission. Appendix C reports the experimental protocols, hyperparameters, data splits, evaluation procedures, and sources of comparison results. Appendix D provides the main ablations and sensitivity analyses, and Appendix G gives the complete ordering of the 440 SNI adapters used for functional-memory construction. We have provided the code for the core components as anonymous supplementary material. If the paper is accepted, all code and associated resources will be made publicly available in a GitHub repository.

REFERENCES

Rickard Brüel-Gabrielsson, Jiacheng Zhu, Onkar Bhardwaj, Leshem Choshen, Kristjan Greenewald, Mikhail Yurochkin, and Justin Solomon. Compress then serve: Serving thousands of LoRA adapters with little overhead, 2024.

Tony F. Chan. Rank revealing QR factorizations. *Linear Algebra and its Applications*, 88–89:67–82, 1987. doi: 10.1016/0024-3795(87)90103-0.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7485–7514. PMLR, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007. doi: 10.1016/j.cviu.2005.09.012.

Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. In *Conference on the Mathematical Theory of Deep Learning*, 2019.

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

Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew Peters. HINT: Hypernetwork instruction tuning for efficient zero- and few-shot generalisation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11272–11288. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.631.

Wooseong Jeong, Wonyoung Lee, and Kuk-Jin Yoon. Preference-aligned LoRA merging: Preserving subspace coverage and addressing directional anisotropy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 823–835, 2026.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

Prakhar Kaushik, Shravan Chaudhari, Ankit Vaidya, Rama Chellappa, and Alan Yuille. The universal weight subspace hypothesis, 2025a.

Prakhar Kaushik, Ankit Vaidya, Shravan Chaudhari, and Alan Yuille. EigenLoRAx: Recycling adapters to find principal subspaces for resource-efficient adaptation and inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 649–659, 2025b.

Prakhar Kaushik, Ankit Vaidya, Shravan Chaudhari, Rama Chellappa, and Alan Yuille. Shared LoRA subspaces for almost strict continual learning, 2026. Accepted at ECCV 2026.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Maria Matveev, Pascal Esser, Ayush Bharadwaj, Lucius Bushnaq, and Gitta Kutyniok. Functional degeneracy in neural networks: Measurement and pruning, 2026.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3846.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

Fabian Paischer, Lukas Hauzenberger, Thomas Schmied, Benedikt Alkin, Marc Deisenroth, and Sepp Hochreiter. Parameter efficient fine-tuning via explained variance adaptation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1539.

Dániel Varga, Adrián Csiszárik, and Zsolt Zombori. Gradient regularization improves accuracy of discriminative models. *Schedae Informaticae*, 27:31–45, 2018. doi: 10.4467/20838476SI.18.003.10408.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1741.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.340.

Qinyuan Ye and Xiang Ren. Learning to generate task-specific adapters from task description. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 646–653. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-short.82.

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

A Additional motivation experiments

We provide the two experiments underlying the observations in Section 3.1. The first examines whether performance lost by restricting a trained LoRA to a small set of directions can be recovered by correcting its scale. The second examines how much additional task-specific structure remains after this correction.

Scale correction. Figure 6 shows the effect of scaling the leading canonical rank-one component of six trained SNI adapters. Despite retaining only a single direction, a substantial fraction of the trained adapter's performance can often be recovered by increasing its scale. In particular, the Frobenius-norm-matched scale frequently lies near the high-performing region. This motivates correcting the magnitude of the projected update before deciding whether additional directions are needed.

Task-dependent residual directions. We next optimize sparse gates over the remaining adapter directions using teacher–student KL divergence on unlabelled task inputs. Table 5 shows that the number of directions required to recover the remaining behaviour varies substantially across tasks. In several cases, performance close to the trained adapter is recovered from only a small fraction of the $3{,}072$ available directions. This motivates allowing the shared basis to grow by a task-dependent amount rather than adding a fixed number of directions for every incoming adapter.

From sparse optimization to direct functional selection. The sparse-gating experiment also highlights a practical limitation of using the regularised objective directly for continual construction. The number of retained directions depends strongly on the penalty λ, and the appropriate trade-off between sparsity and functional recovery differs across tasks. Using this procedure online would require both gradient-based optimization for every incoming adapter and a choice of penalty that indirectly controls how much capacity is added. FML instead uses the same functional intuition without fitting sparse gates. Random-probe sensitivities and CPQR provide an optimization-free ordering of the residual directions, while functional capture directly determines how much of that ordering is admitted. This replaces penalty-controlled sparsity with an explicit behavioural recovery criterion.

B Detailed Methodological Analysis

This section provides complementary details and theoretical analysis for FML, extending the description in Section 3.

B.1 Why Norm Matching Precedes Functional Admission

Section 3 evaluates an incoming LoRA only after its projection into the current basis has been rescaled. Here we make this calculation explicit.

Table 5: Selected residual directions and ROUGE-L as the admission penalty λ varies. Each task begins with $3{,}072$ candidates. A small, task-dependent subset is often sufficient to recover the behavior of the trained adapter.

	task 63		task 116		task 1156		task 1197		task 1283		task 1428	
λ	open	R-L	open	R-L	open	R-L	open	R-L	open	R-L	open	R-L
0	3037	0.926	2410	0.591	2910	0.905	2679	0.979	2940	0.896	2811	0.625
3×10^{-4}	227	0.915	1013	0.574	1397	0.905	614	0.979	959	0.875	736	0.613
1×10^{-3}	122	0.908	645	0.534	560	0.905	367	0.979	614	0.875	220	0.613
3×10^{-3}	16	0.873	274	0.655	193	0.881	92	0.979	225	0.875	54	0.604
1×10^{-2}	3	0.943	96	0.632	153	0.881	63	0.979	174	0.833	7	0.600
trained adapter		0.967		0.648		0.905		0.979		0.875		0.625
base model		0.503		0.068		0.068		0.051		0.066		0.198

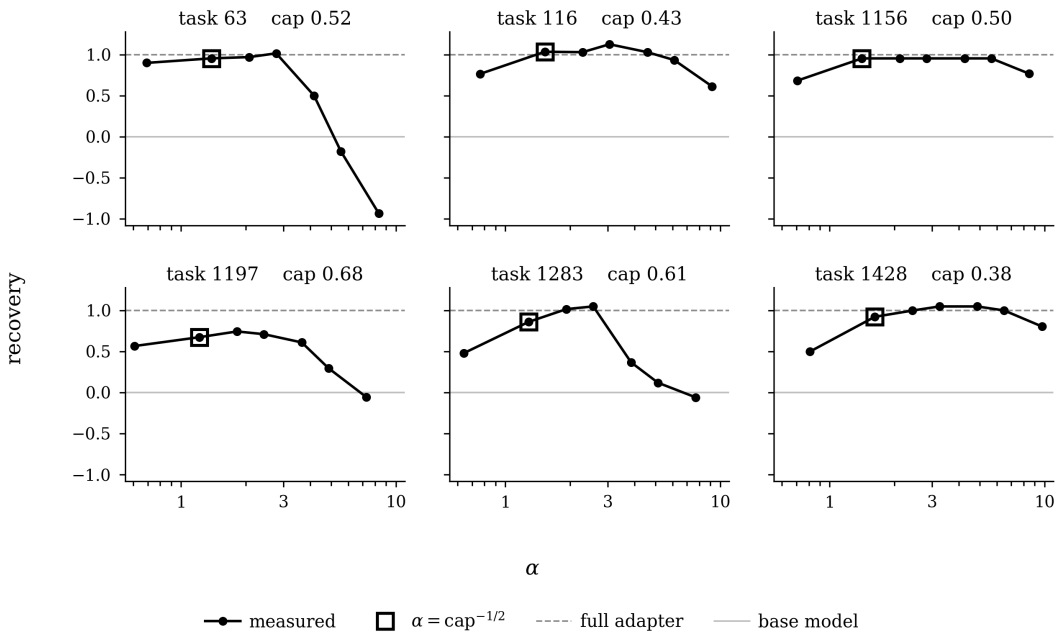


Figure 6: Functional recovery after scaling the leading canonical rank-one component of six task adapters. The square marks the scale that matches the Frobenius norm of the trained update while the dashed and solid horizontal lines denote the full adapter and the base model. Much of the task performance can be recovered by correcting the global scale of the retained direction.

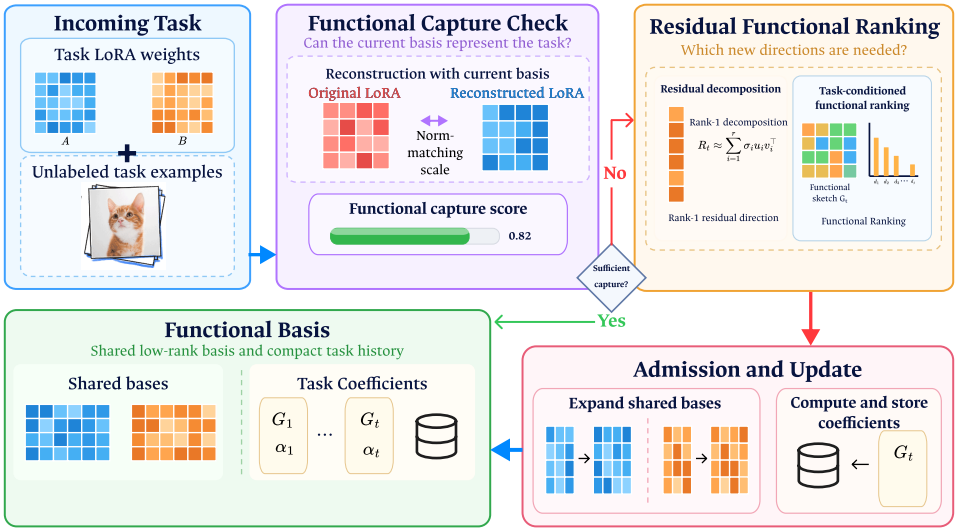


Figure 7: **Functional admission in FML.** An incoming LoRA is projected into the current shared basis and rescaled before measuring functional capture on unlabelled task inputs. If capture is insufficient, directions are ranked by their functional behaviour and admitted until the admission criterion is met. The task is retained only through its coefficients and scale in the shared basis.

For an incoming task t, let $\widetilde{\Delta W}_{t,m}$ denote the projection of module m into the current input and output bases, written as

$$\widetilde{\Delta W}_{t,m} = \widetilde{B}_{t,m}\widetilde{A}_{t,m},$$

where

$$\widetilde{B}_{t,m} = U_{B,m}U_{B,m}^{\top}B_{t,m}, \qquad \widetilde{A}_{t,m} = A_{t,m}U_{A,m}U_{A,m}^{\top}.$$

We use a single global scale,

$$\alpha_t = \sqrt{\frac{\sum_m \|\Delta W_{t,m}\|_F^2}{\sum_m \|\widetilde{\Delta W}_{t,m}\|_F^2}},$$

and evaluate the reconstructed adapter using $\alpha_t \widetilde{\Delta W}_{t,m}$ for every module m. If $\sum_m \|\widetilde{\Delta W}_{t,m}\|_F^2 = 0$, we set $\alpha_t = 1$, since the projected update is identically zero, this convention does not change the reconstruction.

The purpose of this correction is to separate two effects of projection. Discarding weight directions can change both the orientation of the update and its overall magnitude. A reconstruction may therefore retain useful directions but have a substantially smaller effect on the network simply because its norm has decreased. Measuring functional capture before correcting this scale may equate missing functional behaviour with missing directions and cause unnecessary basis expansion. The motivation experiments in Appendix A show that restoring the magnitude alone can recover a substantial fraction of the original task behaviour.

Functional capture then measures the remaining behavioural discrepancy,

$$C_t = 1 - \frac{D_t^{\text{rec}}}{D_t^{\text{base}}},$$

where

$$D_t^{\text{rec}} = \mathbb{E}_{x \in \mathcal{X}_t} D_{\text{KL}} \left(p_t(\cdot \mid x) \,\|\, \widehat{p}_t(\cdot \mid x) \right),$$

and

$$D_t^{\text{base}} = \mathbb{E}_{x \in \mathcal{X}_t} D_{\text{KL}} \left(p_t(\cdot \mid x) \,\|\, p_0(\cdot \mid x) \right).$$

We write $C_t(0) = C_t$ and $D_t(0) = D_t^{\text{rec}}$ for the pre-admission reconstruction. The denominator measures how far the incoming LoRA moves the model away from the frozen base model, while the numerator measures how much of that behaviour is still missing from the reconstruction. Hence $C_t = 1$ corresponds to exact functional recovery, while $C_t = 0$ corresponds to the base-model reference.

Language-generation tasks. Let an unlabelled task example x_i consist of an input prompt and a continuation of length L_i. For continuation position ℓ, let $z_{i,\ell}^t$, $\widehat{z}_{i,\ell}$, and $z_{i,\ell}^0$ denote the logits produced by the incoming LoRA, its reconstruction, and the frozen base model, respectively, under the same input–continuation context. We define

$$p_{i,\ell}^t = \text{softmax}(z_{i,\ell}^t), \qquad \widehat{p}_{i,\ell} = \text{softmax}(\widehat{z}_{i,\ell}), \qquad p_{i,\ell}^0 = \text{softmax}(z_{i,\ell}^0).$$

The teacher–student discrepancy for $q \in \{\widehat{p}, p^0\}$ is

$$\mathcal{L}_{\text{lang}}(p^t, q) = \frac{1}{|\mathcal{X}_t|} \sum_{i=1}^{|\mathcal{X}_t|} \frac{1}{L_i} \sum_{\ell=1}^{L_i} D_{\text{KL}} \left(p_{i,\ell}^t \,\|\, q_{i,\ell} \right),$$

where

$$D_{\text{KL}}(p \| q) = \sum_v p(v) \log \frac{p(v)}{q(v)}.$$

Thus the incoming LoRA provides the target token distribution, while the reconstruction and frozen base model are evaluated on the same unlabelled task examples and continuation positions.

The corresponding functional capture is

$$C_t^{\text{lang}} = 1 - \frac{\mathcal{L}_{\text{lang}}(p^t, \widehat{p})}{\mathcal{L}_{\text{lang}}(p^t, p^0)}.$$

Image-classification tasks. For an unlabelled task input x_i, let z_i^t, $\widehat{z}_i$, and z_i^0 denote the class logits produced by the incoming LoRA, its reconstruction, and the frozen base model using the same task-specific classifier head. With temperature T, define

$$p_i^t = \text{softmax}\left(\frac{z_i^t}{T}\right), \qquad \widehat{p}_i = \text{softmax}\left(\frac{\widehat{z}_i}{T}\right), \qquad p_i^0 = \text{softmax}\left(\frac{z_i^0}{T}\right).$$

We use

$$\mathcal{L}_{\text{vis}}(p^t, q) = \frac{T^2}{|\mathcal{X}_t|} \sum_{i=1}^{|\mathcal{X}_t|} D_{\text{KL}}\left(p_i^t \,\|\, q_i\right),$$

with $T = 1$ in all vision experiments. Functional capture is then

$$C_t^{\text{vis}} = 1 - \frac{\mathcal{L}_{\text{vis}}(p^t, \widehat{p})}{\mathcal{L}_{\text{vis}}(p^t, p^0)}.$$

Unified form. Both modalities therefore use the same normalised recovery measure,

$$C_t = 1 - \frac{\mathcal{L}(p_t, \widehat{p}_t)}{\mathcal{L}(p_t, p_0)},$$

where $\mathcal{L}$ denotes the appropriate task-level KL discrepancy.

B.2 Why Basis Expansion Does Not Forget Admitted Tasks

Throughout the paper, we state that FML introduces no post-admission forgetting. Once a task has been admitted into the functional memory, later tasks may expand the shared basis but do not change the representation already stored for that task. Here we show why this property holds by construction.

Suppose task t has been admitted with input and output bases $U_{A,m}$ and $U_{B,m}$ and coefficient factors $Z_{A,t,m}$ and $Z_{B,t,m}$ for module m. Their product defines the coordinate core

$$G_{t,m} = Z_{B,t,m} Z_{A,t,m}^\top.$$

Because all selected residual directions are appended before the task is stored, $\overline{\Delta W}_{t,m}$ lies in the spans of the admitted input and output bases. Therefore,

$$\overline{\Delta W}_{t,m} = U_{B,m} G_{t,m} U_{A,m}^\top,$$

and the scaled admitted update is

$$\alpha_t \overline{\Delta W}_{t,m} = \alpha_t U_{B,m} G_{t,m} U_{A,m}^\top,$$

where α_t is the global scale stored at admission.

When a later task is admitted, FML only appends new directions to the existing bases. Let

$$U'_{A,m} = [U_{A,m}, D_{A,m}], \qquad U'_{B,m} = [U_{B,m}, D_{B,m}]$$

denote the expanded bases. Since the previously admitted directions are retained unchanged, the earlier task can be represented in the enlarged bases by zero-padding its coefficient factors,

$$Z'_{A,t,m} = \begin{bmatrix} Z_{A,t,m} \\ 0 \end{bmatrix}, \qquad Z'_{B,t,m} = \begin{bmatrix} Z_{B,t,m} \\ 0 \end{bmatrix}.$$

Their product is therefore

$$G'_{t,m} = Z'_{B,t,m} Z'^\top_{A,t,m} = \begin{bmatrix} G_{t,m} & 0 \\ 0 & 0 \end{bmatrix}.$$

Its reconstruction after basis expansion is consequently

$$\begin{aligned} U'_{B,m} G'_{t,m} U'^\top_{A,m} &= [U_{B,m}, D_{B,m}] \begin{bmatrix} G_{t,m} & 0 \\ 0 & 0 \end{bmatrix} [U_{A,m}, D_{A,m}]^\top \\ &= U_{B,m} G_{t,m} U_{A,m}^\top. \end{aligned}$$

Because the stored scale α_t is also unchanged,

$$\alpha_t U'_{B,m} G'_{t,m} U'^\top_{A,m} = \alpha_t \overline{\Delta W}_{t,m}.$$

Because both the coefficient representation and the stored scale remain unchanged under later basis expansion, the reconstructed update for task t is identical before and after subsequent tasks are admitted. Any difference from the original LoRA is fixed when the task is first admitted according to the functional admission criterion, rather than accumulating as the memory continues to grow. This gives the zero post-admission forgetting behaviour for FML.

B.3 Functional Sketch from Random Probes

Section 3.4 assigns each residual candidate a *functional signature* describing how changing that direction affects the model outputs on the incoming task.

For probe q, let x_q denote its associated unlabelled task input. We parameterise the multiplier of residual candidate j as $\exp(c_j)$, so that $c = 0$ assigns unit strength to every residual candidate and recovers the complete incoming adapter. We define

$$a_j(x_q) = \left.\frac{\partial z_t(x_q; c)}{\partial c_j}\right|_{c=0},$$

where $a_j(x_q)$ is the local change in the output logits associated with residual candidate j. This vector is high-dimensional, particularly for language models where it spans continuation positions and the output vocabulary.

Rather than storing $a_j(x_q)$ directly, we observe it through a random output projection r_q,

$$g_{qj} = \left.\frac{\partial \phi_q(c)}{\partial c_j}\right|_{c=0} = \frac{1}{\sqrt{D_{x_q}}} r_q^\top a_j(x_q),$$

where D_{x_q} is the number of output-logit coordinates included in the projection. Each g_{qj} therefore gives one scalar view of how candidate j changes the model output.

Collecting the Q probe measurements gives

$$h_j = (g_{1j}, \ldots, g_{Qj})^\top,$$

which we use as the functional signature of candidate j. Collecting all candidate signatures forms

$$H_t \in \mathbb{R}^{Q \times \sum_m J_m}.$$

Candidates with similar columns in H_t have similar effects on the model outputs over the sampled task examples, while candidates with different columns induce different functional changes. The resulting sketch is used only to order residual directions, while the number eventually admitted is still determined by functional capture in the full model.

B.4 CPQR Ordering of Residual Directions

The functional sketch

$$H_t \in \mathbb{R}^{Q \times \sum_m J_m}$$

contains one column for each residual candidate, with h_j describing its effect on the model outputs over the random probes and unlabelled task examples. We use QR factorisation with column pivoting (CPQR) to turn these signatures into an ordering of residual directions.

Given H_t, CPQR computes

$$H_t \Pi = QR,$$

where Π permutes the columns of H_t. The corresponding pivot order π defines the order in which residual candidates are considered for admission.

The ordering can be understood as a greedy selection procedure. At the first step, CPQR selects a candidate with a large functional signature. After a candidate has been selected, its contribution is removed from the remaining columns, and the next pivot is chosen from the functional variation that is still unexplained. More explicitly, if $\mathcal{S}_k$ denotes the candidates selected after k steps and Q_k is an orthonormal basis for their signatures, the residual signature of an unselected candidate j is

$$h_j^\perp = \left(I - Q_k Q_k^\top\right) h_j.$$

The next candidate is chosen as

$$\pi_{k+1} = \arg\max_{j \notin \mathcal{S}_k} \left\| h_j^\perp \right\|_2.$$

Thus, once one functional effect is already represented by an earlier candidate, another candidate with a similar signature receives a small residual norm and is moved later in the ordering. Candidates whose signatures contain new functional variation are instead promoted.

B.5 FUNCTIONAL RE-RANKING

During continual construction, basis directions are ordered by when they are admitted rather than by how broadly they are used across tasks. After the stream, we can re-express the final basis according to the accumulated stored task representations.

For module m, let

$$G_{t,m} = Z_{B,t,m} Z_{A,t,m}^\top \in \mathbb{R}^{K_{B,m} \times K_{A,m}}$$

denote the coordinate core of task t in the final basis, with the coefficient factors zero-padded where necessary for tasks admitted before the basis reached its final size.

We accumulate input- and output-side usage as

$$S_{A,m} = \sum_t \alpha_t^2 G_{t,m}^\top G_{t,m}, \qquad S_{B,m} = \sum_t \alpha_t^2 G_{t,m} G_{t,m}^\top.$$

The stored scale α_t is included so that usage reflects the magnitude of the reconstructed task update.

We then compute

$$S_{A,m} = V_{A,m} \Lambda_{A,m} V_{A,m}^\top, \qquad S_{B,m} = V_{B,m} \Lambda_{B,m} V_{B,m}^\top,$$

with eigenvalues ordered from largest to smallest. Equivalently, this canonicalises the accumulated task representations along their principal cross-task usage directions. The final bases are rotated as

$$U_{A,m}^* = U_{A,m} V_{A,m}, \qquad U_{B,m}^* = U_{B,m} V_{B,m},$$

and the corresponding coefficient factors are transformed as

$$Z_{A,t,m}^* = V_{A,m}^\top Z_{A,t,m}, \qquad Z_{B,t,m}^* = V_{B,m}^\top Z_{B,t,m}.$$

The task coordinate core is therefore

$$\begin{aligned} G_{t,m}^* &= Z_{B,t,m}^* Z_{A,t,m}^{*\top} \\ &= V_{B,m}^\top G_{t,m} V_{A,m}. \end{aligned}$$

This transformation may be applied at any point during the stream. For previously admitted tasks, transforming the stored coefficient factors together with the bases leaves their reconstructed updates unchanged, as shown above. For future tasks, the projection operators are also unchanged because

$$U_{A,m}^* U_{A,m}^{*\top} = U_{A,m} V_{A,m} V_{A,m}^\top U_{A,m}^\top = U_{A,m} U_{A,m}^\top,$$

and analogously

$$U_{B,m}^* U_{B,m}^{*\top} = U_{B,m} U_{B,m}^\top.$$

Therefore an incoming LoRA has exactly the same projection and residual before and after re-ranking. Functional re-ranking can consequently be performed during or after the stream without changing previously stored task representations or the admission of future tasks.

The operation changes only the coordinate system and ordering within the existing functional basis. Directions with greater accumulated cross-task usage are placed earlier, while the represented subspace and stored task updates remain unchanged. The corresponding eigenvalues in $\Lambda_{A,m}$ and $\Lambda_{B,m}$ provide the cross-task usage ordering used in our analysis and downstream initialization.

B.6 FUNCTIONAL ADMISSION CRITERION AND NUMERICAL EDGE CASES

At each candidate set size k, we reconstruct the incoming adapter using the current basis together with the first k residual directions in the CPQR ordering, recompute the global norm-matching factor, and evaluate functional capture in the full model. CPQR therefore determines which residual directions are considered, while admission determines how many are added.

Language generation. For the SNI construction, we use a functional-capture threshold

$$\tau = 0.98.$$

We evaluate candidate set sizes

$$k \in \{8, 16, 24, \ldots, 256\}.$$

For a tested set size k, let

$$D_t(k) = \mathcal{L}_{\text{lang}}\left(p_t, \widehat{p}_t^{(k)}\right)$$

denote its teacher–student KL discrepancy. A candidate set is admitted once

$$C_t(k) \geq 0.98$$

and either

$$D_t(k) \leq 0.05,$$

or further expansion has ceased to meaningfully reduce the discrepancy. For two consecutive tested set sizes $k^- < k$, we measure the relative KL improvement as

$$\rho_t(k) = \frac{D_t(k^-) - D_t(k)}{\max(D_t(k^-), \epsilon)}.$$

We treat the discrepancy as saturated when

$$\rho_t(k) < 0.05$$

for two consecutive expansions after the functional-capture target has been reached. If none of the tested candidate sets satisfies the full criterion, we retain the tested set with the highest functional capture.

We additionally handle two degenerate cases. If the incoming LoRA is nearly indistinguishable from the frozen base model on the unlabelled task examples,

$$\mathcal{L}_{\text{lang}}(p_t, p_0) \leq 10^{-6},$$

the normalised capture ratio is not informative and no expansion is performed. Similarly, if the existing reconstruction already has effectively perfect capture,

$$C_t(0) \geq 1 - 10^{-4},$$

we retain the existing basis without forcing the minimum candidate-set size. If no non-negligible residual directions remain after projection, no new directions are added.

Image classification. For image classification, we use the same functional admission criterion as in the language setting. The only difference is that we do not impose a maximum candidate-set size, since the vision experiments have a substantially smaller residual candidate set and can therefore consider all ranked directions when necessary.

B.7 Complete FML Algorithm

Algorithm 1 summarises the complete continual construction of FML. The admission criterion and numerical edge cases are given in Section B.6.

We denote the pre-admission functional capture by $C_t(0)$, with $C_t(k)$ denoting capture after adding the first k ranked residual candidates. $\mathrm{Admit}_t(k; \tau)$ denotes the complete admission criterion in Section B.6, including its discrepancy history and numerical edge cases. $\mathrm{Factorise}(G)$ returns coefficient factors Z_B, Z_A satisfying $G = Z_B Z_A^\top$ exactly.

C Experimental Details

C.1 Experimental Settings and Hyperparameters

This section provides the experimental settings, additional context and implementation details.

Algorithm 1 Functional Memory of Low-rank subspaces (FML)

Require: LoRA stream $\{\mathcal{L}_t = \{\Delta W_{t,m} = B_{t,m}A_{t,m}\}_m\}_{t=1}^T$, unlabelled task inputs $\{\mathcal{X}_t\}_{t=1}^T$, ranking inputs $\{\mathcal{X}_t^{\text{rank}}\}_{t=1}^T$, functional-capture threshold τ, random probes Q
Ensure: Functional bases $\{U_{A,m}, U_{B,m}\}_m$, task coefficient factors $\{Z_{A,t,m}, Z_{B,t,m}\}_{t,m}$, scales $\{\alpha_t\}_{t=1}^T$

$U_{A,m}, U_{B,m} \leftarrow \varnothing$ for all m
for $t = 1$ to T **do**
  ▷ Project into existing memory
  for each module m **do**
    $\widetilde{B}_{t,m} \leftarrow U_{B,m}U_{B,m}^\top B_{t,m}$
    $\widetilde{A}_{t,m} \leftarrow A_{t,m}U_{A,m}U_{A,m}^\top$
    $\widetilde{\Delta W}_{t,m} \leftarrow \widetilde{B}_{t,m}\widetilde{A}_{t,m}$
  end for
  $\alpha_t \leftarrow \text{NormMatch}(\{\Delta W_{t,m}\}_m, \{\widetilde{\Delta W}_{t,m}\}_m)$
  $C_t(0), D_t(0) \leftarrow \text{Evaluate}(\mathcal{L}_t, \alpha_t\widetilde{\mathcal{L}}_t, \mathcal{X}_t)$
  $\overline{\mathcal{L}}_t \leftarrow \widetilde{\mathcal{L}}_t$
  if $\neg\,\text{Admit}_t(0;\tau)$ **then**
    $\mathcal{O}_t \leftarrow$ FUNCTIONALRANKING$(\mathcal{L}_t, \widetilde{\mathcal{L}}_t, \mathcal{X}_t^{\text{rank}}, Q)$ ▷ Algorithm 2
    $(\mathcal{S}_t, \alpha_t, \overline{\mathcal{L}}_t) \leftarrow$ FUNCTIONALADMISSION$(\mathcal{L}_t, \widetilde{\mathcal{L}}_t, \mathcal{O}_t, \mathcal{X}_t, \tau)$ ▷ Algorithm 3
    $\{U_{A,m}, U_{B,m}\}_m \leftarrow \text{Append}(\{U_{A,m}, U_{B,m}\}_m, \mathcal{S}_t)$
    Zero-pad previously stored coefficient factors
  end if
  ▷ Store task representation
  for each module m **do**
    $G_{t,m} \leftarrow U_{B,m}^\top \overline{\Delta W}_{t,m} U_{A,m}$
    $(Z_{B,t,m}, Z_{A,t,m}) \leftarrow \text{Factorise}(G_{t,m})$
  end for
  Store $\{Z_{A,t,m}, Z_{B,t,m}\}_m, \alpha_t$
  Discard $\mathcal{L}_t, \mathcal{X}_t, \mathcal{X}_t^{\text{rank}}$
end for
$\{U_{A,m}, U_{B,m}, Z_{A,t,m}, Z_{B,t,m}\} \leftarrow \text{ReRank}(\{U_{A,m}, U_{B,m}, Z_{A,t,m}, Z_{B,t,m}, \alpha_t\}_{t,m})$
return $\{U_{A,m}, U_{B,m}\}_m, \{\{Z_{A,t,m}, Z_{B,t,m}\}_m, \alpha_t\}_{t=1}^T$

Functional-memory construction. For language, $\mathcal{X}_t^{\text{rank}}$ contains 40 unlabelled teacher–continuation pairs for functional ranking, while $\mathcal{X}_t$ uses its first 8 pairs for full-model admission evaluation. The 32-example evaluation reported for the 440-task comparison is used only for the post-admission performance audit and does not influence basis construction.

For controlled comparisons, the weight-ranked variant uses the same incoming adapters, residual candidate set, norm matching, and admission criterion as FML, only the ordering of residual directions is changed. The offline UWSH comparison is constructed from the complete adapter collection and evaluated at matched total basis capacity.

Algorithm 2 FUNCTIONALRANKING: functional ranking of residual directions

Require: Incoming LoRA $\mathcal{L}_t$, current reconstruction $\widetilde{\mathcal{L}}_t$, ranking inputs $\mathcal{X}_t^{\mathrm{rank}}$, random probes Q
Ensure: Ordered residual candidates $\mathcal{O}_t$

$\mathcal{O}_t \leftarrow \varnothing$
for each module m **do**
  $\Delta W_{t,m}^{\mathrm{res}} \leftarrow \Delta W_{t,m} - \widetilde{\Delta W}_{t,m}$
  Append the non-negligible rank-one components of $\mathrm{SVD}(\Delta W_{t,m}^{\mathrm{res}})$, tagged by module m, to $\mathcal{O}_t$
end for
if $\mathcal{O}_t = \varnothing$ **then**
  return $\mathcal{O}_t$
end if
Distribute Q probe indices approximately evenly across $\mathcal{X}_t^{\mathrm{rank}}$
Parameterise residual candidate j by multiplier $\exp(c_j)$
for each $x_i \in \mathcal{X}_t^{\mathrm{rank}}$ **do**
  for each probe q assigned to x_i **do**
    Sample random probe r_q
    $g_{qj} \leftarrow \left.\dfrac{\partial \phi_q(c)}{\partial c_j}\right|_{c=0}$ for all candidates j
  end for
end for
Construct $H_t \in \mathbb{R}^{Q \times |\mathcal{O}_t|}$ from candidate signatures $h_j = (g_{1j}, \ldots, g_{Qj})^\top$
$\pi \leftarrow \mathrm{CPQR}(H_t)$
Reorder $\mathcal{O}_t$ according to π
return $\mathcal{O}_t$

Algorithm 3 FUNCTIONALADMISSION: functional admission

Require: Incoming LoRA $\mathcal{L}_t$, current reconstruction $\widetilde{\mathcal{L}}_t$, ordered residual candidates $\mathcal{O}_t$, unlabelled inputs $\mathcal{X}_t$, functional-capture threshold τ
Ensure: Selected candidates $\mathcal{S}_t$, scale α_t, accepted reconstruction $\overline{\mathcal{L}}_t$

if $\mathcal{O}_t = \varnothing$ **then**
  $\alpha_t \leftarrow \mathrm{NormMatch}(\mathcal{L}_t, \widetilde{\mathcal{L}}_t)$
  return $\varnothing, \alpha_t, \widetilde{\mathcal{L}}_t$
end if
for each tested candidate set size k **do**
  $\mathcal{S}_k \leftarrow \{\mathcal{O}_{t,1}, \ldots, \mathcal{O}_{t,k}\}$
  $\overline{\mathcal{L}}_t^{(k)} \leftarrow \mathrm{Reconstruct}(\widetilde{\mathcal{L}}_t, \mathcal{S}_k)$
  $\alpha_t^{(k)} \leftarrow \mathrm{NormMatch}(\mathcal{L}_t, \overline{\mathcal{L}}_t^{(k)})$
  $C_t(k), D_t(k) \leftarrow \mathrm{Evaluate}(\mathcal{L}_t, \alpha_t^{(k)}\overline{\mathcal{L}}_t^{(k)}, \mathcal{X}_t)$
  if $\mathrm{Admit}_t(k; \tau)$ **then**
    return $\mathcal{S}_k, \alpha_t^{(k)}, \overline{\mathcal{L}}_t^{(k)}$
  end if
end for
$k^\star \leftarrow \arg\max_k C_t(k)$
return $\mathcal{S}_{k^\star}, \alpha_t^{(k^\star)}, \overline{\mathcal{L}}_t^{(k^\star)}$

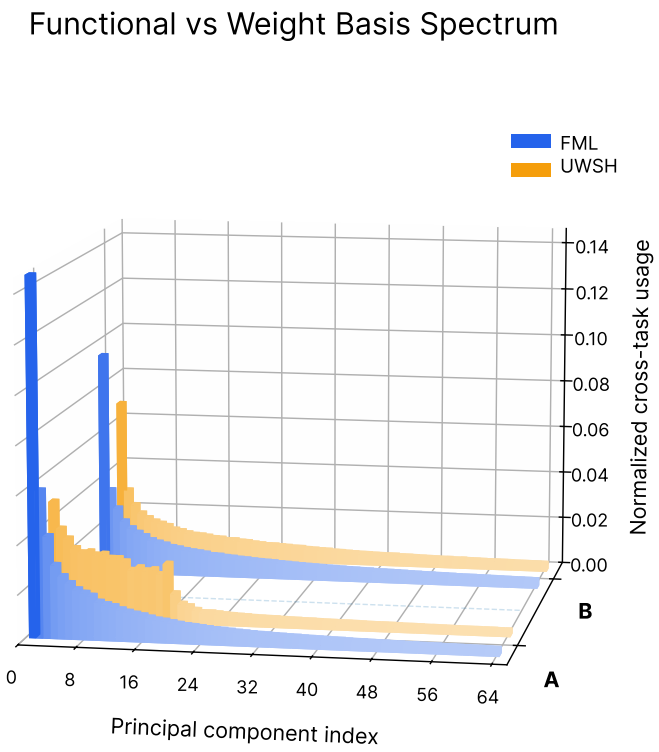


Figure 8: *Functional reranking concentrates cross-task shared structure.* Cross-task spectra of the input (A) and output (B) LoRA bases for FML and the weight-based UWSH subspace (Kaushik et al., 2025a) After end-of-stream reranking using the stored task representations, FML places substantially more shared energy in the leading directions, indicating a more concentrated reusable basis.

Language retention and task ordering. Order sensitivity is evaluated on the same 22-task set used in the SHARE comparison. Each order begins from an empty functional memory and observes each completed task LoRA exactly once. We evaluate SHARE's original order and two fixed random permutations, while keeping the task examples and task-specific randomisation fixed across runs. No basis directions or stored task coefficients are transferred between orders.

Vision task adapters. Unlike the 440-task SNI stream, the vision benchmarks do not provide a corresponding collection of pretrained task-specific LoRAs. We follow SHARE's task-splitting protocol and independently train a LoRA for each task using the settings reported in Table 7. Each task adaptation starts from the same pretrained ViT-B/16 backbone rather than from the model obtained after the preceding task. The resulting trained LoRA and its task-specific classifier head are then treated as the completed task experience presented to FML.

For CIFAR-100 we use the standard train/test split. Flowers uses the official train and validation partitions for training and the official test partition for evaluation. Since Caltech-101 does not provide a standard train/test partition and the exact split used by SHARE is not available, we use a fixed stratified 70/30 split. After every admission, all tasks observed so far are re-evaluated from their stored representations, giving the complete accuracy matrix from which forgetting is measured.

Target-data-free initialization. The five evaluation tasks are excluded from functional-memory construction and from the historical statistics used for global rank allocation. For each seed, all initialization methods use the same target-task training and evaluation splits and are subsequently fine-tuned under the same optimization budget.

The uniform FML initialization assigns rank equally across adapted modules, while the global variant allocates the same total rank budget using cross-task functional usage accumulated from source tasks only. No target-task inputs are used by either FML variant at initialization. EVA is the only comparison in this experiment that uses target inputs before fine-tuning, using 64 prompt-only examples. After initialization, FML does not constrain optimization to remain within the functional basis – the new rank-r LoRA is free to adapt in its ordinary parameterisation.

Coefficient generation and adaptation. For each source task, natural-language description paraphrases are split deterministically at the task level, with 20 descriptions held out for validation. We use the same hypernetwork architecture as T2L, but modify its output space and training objective to predict task representations within the functional memory rather than full LoRA weights. Description embeddings are precomputed once and the generator is trained only from these embeddings and the stored functional-memory representations. The original task examples, trained LoRA weights, and base language model are not accessed during generator training.

Rather than predicting the scaled task coefficients directly, we separate their *direction* from their overall magnitude. The generator predicts the coefficient pattern within the functional basis using a cosine loss against the stored task representation, while a separate head predicts the logarithm of the task-update magnitude. This avoids requiring the coefficient predictor to model differences in both orientation and scale simultaneously. We additionally include a small reconstruction term after applying the predicted magnitude. The resulting objective is

Let

$$G_{t,m} = Z_{B,t,m} Z_{A,t,m}^{\top}, \qquad c_t = \mathrm{concat}_m \, \mathrm{vec}(G_{t,m}),$$

denote the complete coefficient representation of task t.

$$\mathcal{L}_{\mathrm{CoeffGen}} = (1 - \cos(\widehat{c}_t, c_t)) + 0.25\, \mathcal{L}_{\mathrm{mag}} + 0.10\, \mathcal{L}_{\mathrm{recon}},$$

where $\mathcal{L}_{\mathrm{mag}}$ measures error in the log update magnitude and $\mathcal{L}_{\mathrm{recon}}$ measures the relative reconstruction error after rescaling.

For downstream coefficient adaptation, the functional basis remains frozen throughout and only the generated task coefficients are optimized. Validation is performed periodically on the target benchmark and is used to select the reported fully adapted endpoint.

Table 6: Hyperparameters used for functional-memory construction in language and vision. The vision candidate set is sufficiently small that all ranked residual directions can be considered when necessary.

Setting	Language	Vision
Backbone	Mistral-7B-Instruct-v0.2	ViT-B/16
Incoming LoRA rank	16	8
Adapted modules	96 k/q/v modules	24 q/v modules
Construction stream	440 SNI adapters	10 tasks $\times$ 10 classes per dataset
Functional-capture threshold τ	0.98	0.98
Unlabelled inputs for functional ranking	40	8
Functional-evaluation inputs	8	16
Minimum candidate-set size	8 directions	8 directions
Candidate-set increment	8 directions	8 directions
Maximum candidate-set size	256 directions	all available candidates

Table 7: Training settings for the incoming task LoRAs in the vision continual-learning experiments.

Setting	Value
Backbone	ViT-B/16
Pretrained checkpoint	`google/vit-base-patch16-224-in21k`
Retained classes	100
Tasks per dataset	10
Classes per task	10
Task split seed	42
LoRA rank	8
LoRA scaling	$16r$
LoRA dropout	0
Task-training epochs	10
LoRA learning rate	5×10^{-4}
Classifier learning rate	1×10^{-3}
Batch size	128
Weight decay	1×10^{-6}
Learning-rate schedule	cosine
Training precision	BF16
Caltech train/test split	stratified 70/30
Flowers split	official train+val / test

C.2 Sources of Comparison Results

We summarise the sources of comparison results reported in the paper. Other experiments not stated here are completed and obtained from our own experiments.

Vision continual-learning comparison. For the vision continual-learning experiments in Table 1, the results for EWC, LwF, L2P, DualPrompt, CODA-P, DAP, EASE, and SHARE are taken directly from the evaluation reported by SHARE (Kaushik et al., 2026). The FML results are obtained from our own experiments using the task construction and evaluation procedure described in Appendix C. The independently trained LoRA reference is also obtained from our vision task adaptations.

22-task language continual-learning comparison. For the 22-task language experiment in Figure 4, the SHARE and task-specific Oracle LoRA results are taken from the corresponding experiment reported by SHARE (Kaushik et al., 2026). We evaluate FML on the same set of tasks and use SHARE's original task order for the main comparison. The additional order-sensitivity experiments reported in Appendix D are performed by us using the same task set under two additional fixed permutations.

Table 8: Hyperparameters for LoRA initialization and subsequent fine-tuning on the five held-out SNI tasks.

Setting	Value
Held-out tasks	T039, T290, T391, T442, T1598
LoRA rank	$r = 4$
Total rank budget	$\sum_m r_m = 384$
Training examples	1,024
Epochs	2
Batch size	8
optimizer steps	256
optimizer	AdamW
Learning rate	1×10^{-3}
Gradient clipping	1.0
Seeds	0, 1, 2
ROUGE-L evaluation	full test set
Beam size	4
Maximum generated tokens	32

Table 9: Hyperparameters for description-conditioned coefficient generation and subsequent coefficient-only adaptation.

Setting	FML-CoeffGen training	Coefficient adaptation
Description encoder	`Alibaba-NLP/gte-large-en-v1.5`	–
Description pooling	CLS	–
Embedding dimension	1024	–
Embedding normalisation	$\ell_2 \times \sqrt{1024}$	–
Predicted factor rank	16	–
Hidden dimension	512	–
Dropout	0.05	–
Training epochs	180	up to 5
Task batch size	24	8
Learning rate	3×10^{-4}	3×10^{-4}
Weight decay	1×10^{-3}	–
Gradient clipping	1.0	1.0
Validation interval	5 epochs	32 updates
Held-out descriptions/source task	20	–
Pattern-loss weight	1	–
Magnitude-loss weight	0.25	–
Reconstruction-loss weight	0.10	–
Early-stopping patience	–	10 validations

Target-data-free LoRA initialization. For Table 2, the fully trained rank-16 LoRA reference is taken from the EigenLoRAx evaluation (Kaushik et al., 2025b). The remaining initialization results, including LoRA, PiSSA, UWSH (EigenLoRAx-Init), EVA, FML-Init, and FML-Init with global rank allocation, are obtained from our own experiments under the common fine-tuning protocol described in Appendix C. All methods are evaluated on the same five held-out SNI tasks.

Description-conditioned generation. For Table 3, the comparison results for Mistral-7B-Instruct, prepending the task description, 3-shot ICL, Average LoRA, Multi-task LoRA, Arrow Routing, Hyperdecoders, and T2L-L are taken from the Text-to-LoRA evaluation reported by Charakorn et al. (2025). The FML-CoeffGen result is obtained from our own experiment on the same seven downstream language benchmarks and under the same evaluation setting.

Coefficient adaptation. For Table 4, the task-specific LoRA Oracle results are taken from the Text-to-LoRA evaluation reported by Charakorn et al. (2025). All FML rows, including the

zero-shot generated coefficients, few-shot coefficient-adaptation settings, and the fully adapted coefficient-only result, are obtained from our own experiments.

D Ablation Studies

We examine the main design choices in FML and the sensitivity of its functional-admission procedure. In these experiments, we ablate on different components of FML using 64 SNI tasks stream.

Contribution of scale correction and the admission criterion. Figure 9 isolates two components of the construction. Removing norm matching substantially increases basis growth, from approximately 11.4K to 17.3K directions, while also increasing the residual teacher–student divergence and slightly reducing final ROUGE-L. Using functional capture alone as the stopping condition has the opposite effect. The resulting memory is smaller, at approximately 8.5K directions, but retains a substantially larger residual KL discrepancy and reduces average ROUGE-L from 65.8 to 62.7.

Functional-capture threshold. The functional threshold τ directly controls the trade-off between memory growth and fidelity. Figure 10 varies $\tau \in \{0.95, 0.98, 0.99\}$ while holding the remaining construction settings fixed. Increasing the threshold produces a progressively larger basis together with higher ROUGE-L and recovery. The improvement from 0.98 to 0.99 comes with a noticeably larger increase in memory than the change from 0.95 to 0.98. We use $\tau = 0.98$ throughout the main experiments as an intermediate operating point that retains high functional recovery without requiring the more conservative growth rate.

Number of functional probes. Figure 11 varies the number of random output probes used to construct the functional signatures. Increasing Q from 64 to 256 leads to progressively more conservative basis growth and improves both downstream ROUGE-L and recovery. A larger sketch provides more measurements of the candidate directions' effects on the model outputs, allowing the admission ordering to retain more functionally useful structure. We use $Q = 256$ in the main language construction, prioritising functional recovery while remaining far smaller than the output space whose behaviour is being sketched.

Number of unlabelled task inputs. We next vary the number of unlabelled task inputs used to characterise each incoming adapter. As shown in Figure 12, using 8, 16, or 40 inputs produces very similar final memory sizes and recovery, showing our method's robustness to small set of unlabelled task inputs.

Does additional task experience improve downstream adaptation? The main experiments use the functional memory after observing the complete 440-adapter stream. We additionally evaluate FML-Init using memories available after 64, 128, 256, and 440 adapters, while keeping the downstream rank budget and fine-tuning protocol fixed. At each checkpoint, functional re-ranking uses only the tasks that have already been observed.

Figure 13 shows that downstream ROUGE-L improves as the memory grows from 64 to 256 observed adapters. Performance then remains similar at 440 adapters rather than continuing to increase monotonically. This suggest that reusable downstream structure emerges progressively from accumulated task experience and begins to saturate before the complete stream has been observed.

Order Sensitivity Because FML constructs its memory sequentially, the directions available to an incoming adapter depend on the tasks observed previously. We therefore repeat the 22-task language construction using SHARE's original order and two fixed random permutations.

Table 10 shows that both final performance and memory size are stable across the three orders. Average ROUGE-L varies from 74.67 to 74.93, with a mean of 74.80 ± 0.13, while the final basis size ranges from $3{,}296$ to $3{,}504$ directions. Figure 14 shows the corresponding task-level variation. The median difference between the maximum and minimum ROUGE-L obtained for an individual task is only 0.34 points across orders, although a small number of tasks are more order-sensitive. This indicates that the sequence of completed task adaptations can affect individual admissions, but has limited effect on the overall retention and capacity of the resulting functional memory.

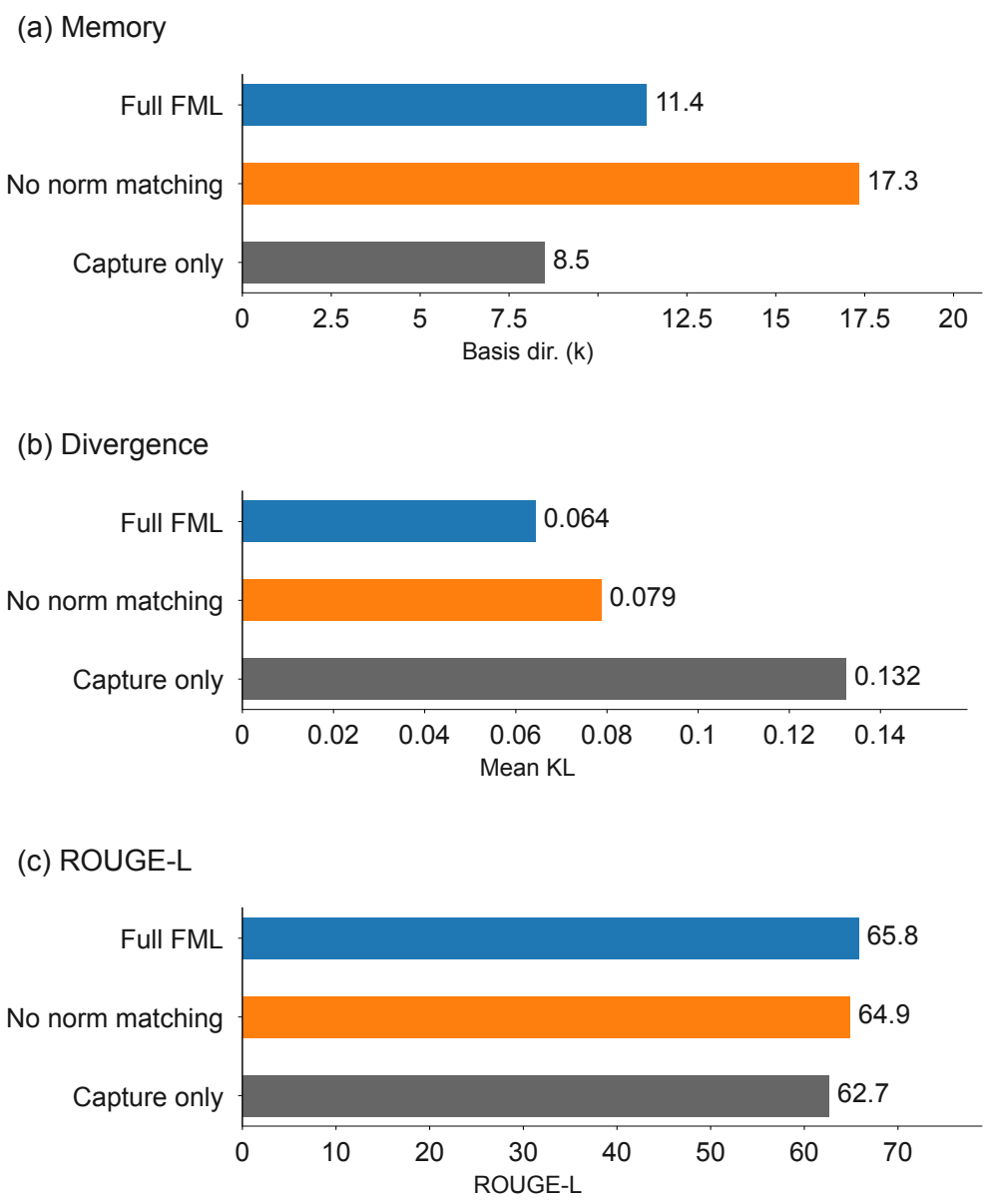


Figure 9: **Component ablations on the fixed 64-task SNI stream.** We compare full FML with construction without global norm matching and with admission based only on the functional-capture threshold. We report final basis size, mean residual teacher–student KL divergence, and average ROUGE-L over the 64 admitted tasks.

E Clarifications on Experimental Scope

We clarify several aspects of the experimental setting that may otherwise be ambiguous from the condensed presentation in the main text.

Why do we compare against conventional continual-learning methods on the vision benchmarks? We follow the evaluation protocol reported by SHARE so that our results can be placed in the same benchmark context. The conventional continual-learning baselines reported in Table 1 are taken directly from SHARE and do not operate under exactly the same learning interface as FML.

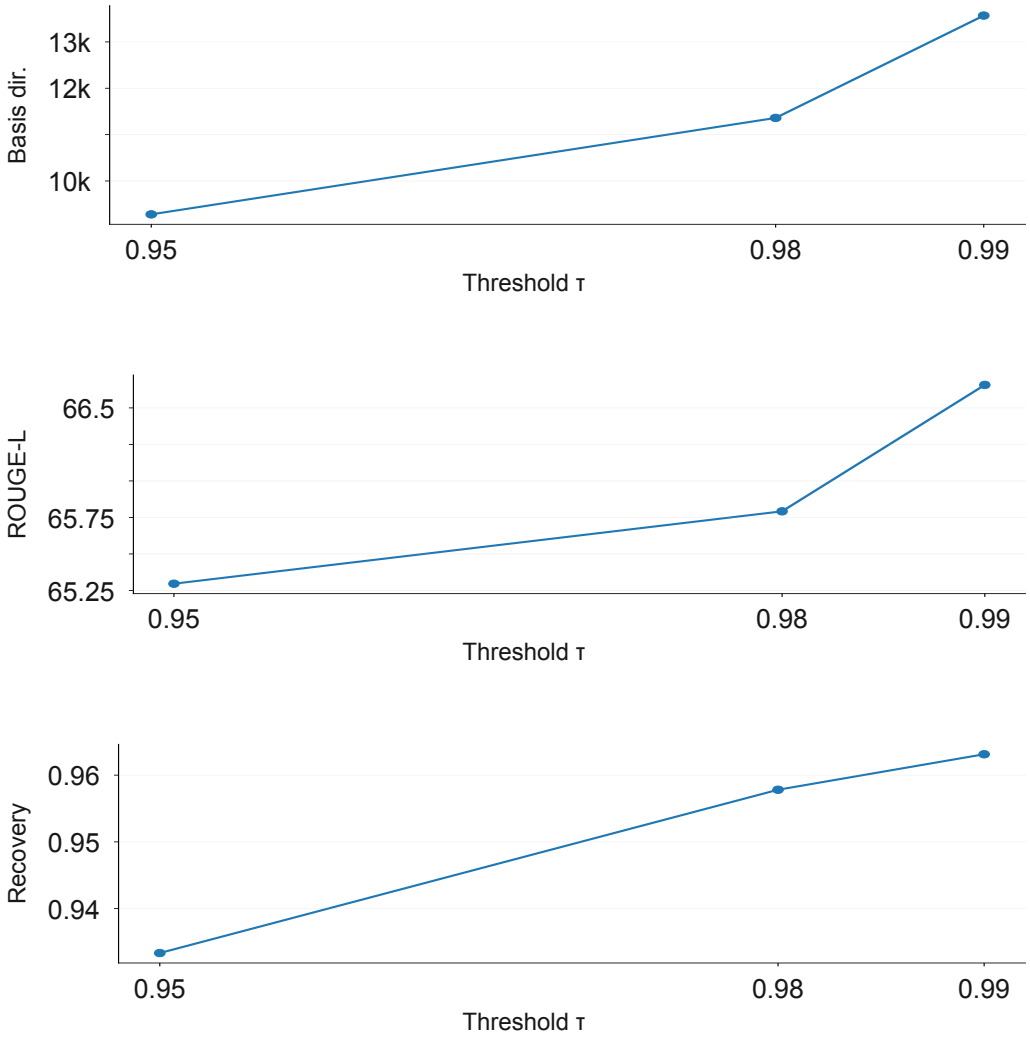


Figure 10: **Sensitivity to the functional-capture threshold.** Final basis size, average ROUGE-L, and recovery on the fixed 64-task stream for $\tau \in \{0.95, 0.98, 0.99\}$. Higher thresholds trade additional memory for more conservative functional recovery.

Table 10: Sensitivity to task order on the 22-task language stream. Each run begins from an empty functional memory and uses the same set of completed task adaptations.

Task order	Final ROUGE-L ↑	Final basis size ↓
SHARE original	74.80	3,328
Permutation 1	74.67	3,296
Permutation 2	74.93	3,504
Mean ± SD	74.80 ± 0.13	$3{,}376 \pm 112$

Methods such as EWC, LwF, and prompt-based continual learners modify or regularise the model as task data arrive, whereas SHARE and FML consolidate the outcome of completed task adaptations represented by trained LoRAs. SHARE is therefore the closest comparison in this table. We retain the broader continual-learning baselines to preserve comparability with the established benchmark and to contextualise performance on the same task sequences.

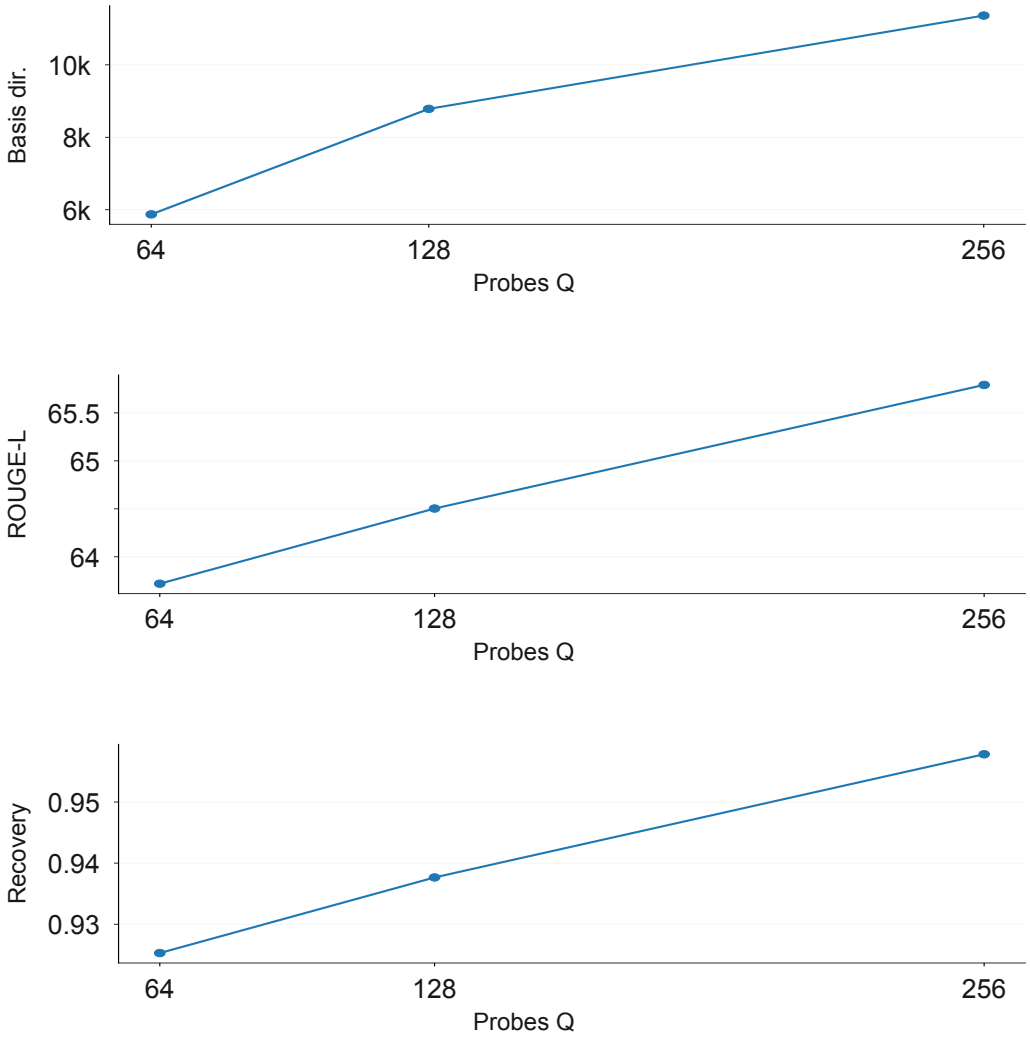


Figure 11: **Sensitivity to the number of random functional probes.** Basis size, average ROUGE-L, and recovery after construction with $Q \in \{64, 128, 256\}$ on the same 64-task stream.

What is matched in the global-rank initialization comparison? The comparison fixes the total rank budget $\sum_m r_m = 384$. The uniform methods distribute this budget evenly across modules, whereas the global functional variant redistributes rank according to historical cross-task usage.

Why compare FML with an offline UWSH subspace? UWSH provides a useful weight-space reference because it is constructed from the complete adapter collection jointly, whereas FML operates on a strict continual learning setting where it is only allowed to observe one adapter at a time. The comparison tests whether a continually constructed functional basis can approach the reconstruction quality of a subspace that has access to the full collection. It is not intended to imply that the two methods operate under the same problem settings.

F Limitations and Future Work

Dependence on the functional admission rule. The final basis size depends on the functional-capture threshold and on the number of random probes used to estimate residual behaviour. These choices control the trade-off between compact growth and conservative recovery of an incoming

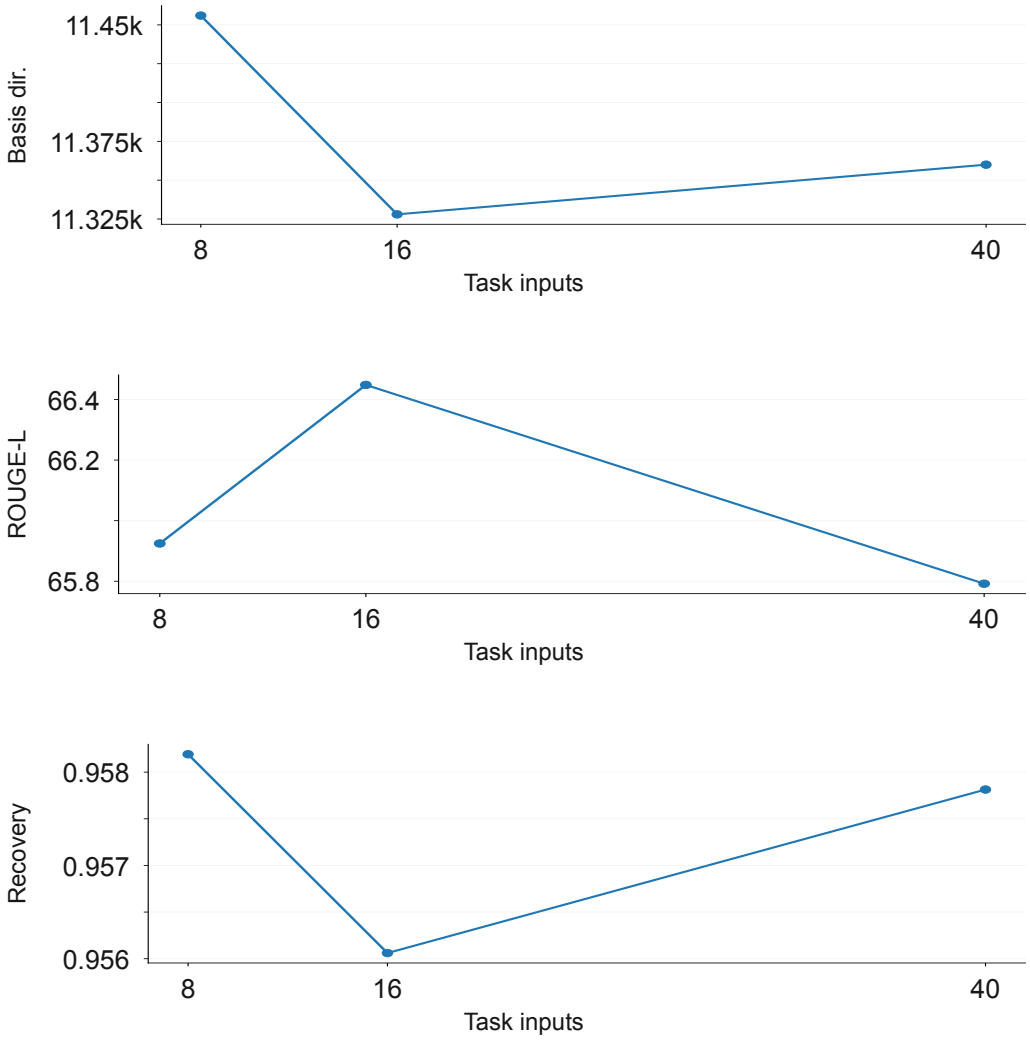


Figure 12: **Sensitivity to the amount of unlabelled task information.** Final basis size, average ROUGE-L, and recovery when functional ranking is constructed from 8, 16, or 40 unlabelled inputs per incoming task.

adapter. Unlike fixed-capacity constructions, however, FML can add different numbers of directions for different tasks according to the remaining functional discrepancy. The random-probe sketch also avoids fitting an additional optimization objective for ranking residual directions, since each probe provides sensitivities for all candidates through a vector–Jacobian product. Future work could explore more adaptive admission criteria, including rules that vary the functional threshold or the number of probes according to task difficulty or a fixed memory budget.

Dependence on representative unlabelled task inputs. Functional admission is estimated from the unlabelled task inputs available when the adapter arrives. If these inputs do not adequately represent the task distribution, the estimated functional capture may differ from the adapter's behaviour on the full task distribution, which could lead to insufficient or overexpansion of the functional basis. In our experiments, relatively small unlabelled sets were sufficient to retain strong task performance across language and vision. A broader study of robustness to mismatched or biased unlabelled task inputs is left for future work.

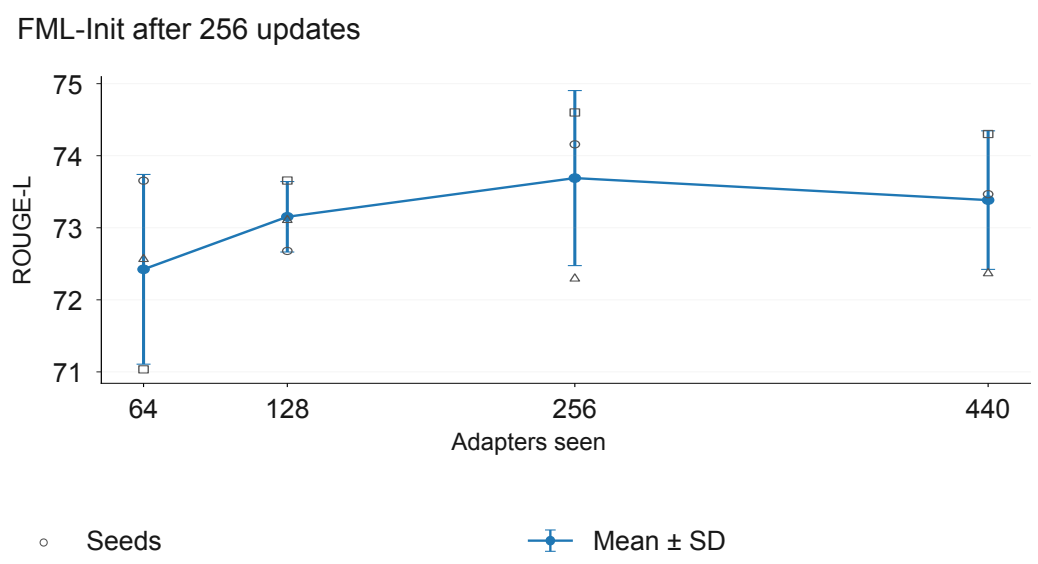


Figure 13: **Downstream utility as task experience accumulates.** Mean test ROUGE-L after 256 target-task updates when FML-Init is constructed from the functional memory available after 64, 128, 256, or 440 observed adapters. Markers show individual seeds and error bars show mean ± standard deviation.

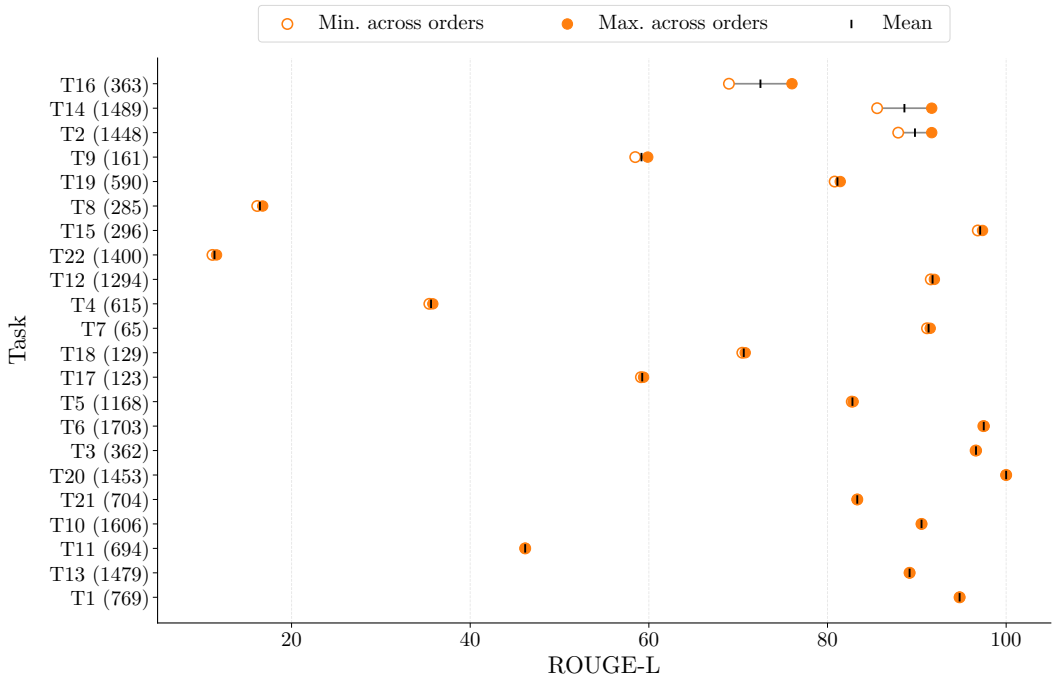


Figure 14: **Task-level sensitivity to stream order.** For each of the 22 language tasks, the interval shows the minimum and maximum final ROUGE-L across the three task orders, with the marker indicating the mean. Most tasks exhibit little variation across orders, while a small number are more sensitive to when their adaptation is encountered in the stream.

G Complete SNI task order

For completeness, Table 11 reports the exact order of the 440 SNI adapters used for functional-memory construction.

Table 11: Complete task order for the 440-adapter SNI construction stream. Tasks are listed in the order in which their trained LoRAs are presented to FML.

Pos.	Task	Pos.	Task	Pos.	Task	Pos.	Task	Pos.	Task	Pos.	Task	Pos.	Task	Pos.	Task
1	t0022	2	t0043	3	t0044	4	t0045	5	t0047	6	t0050	7	t0064	8	t0065
9	t0066	10	t0067	11	t0068	12	t0069	13	t0070	14	t0072	15	t0074	16	t0075
17	t0076	18	t0077	19	t0079	20	t0083	21	t0084	22	t0085	23	t0087	24	t0089
25	t0092	26	t0093	27	t0094	28	t0095	29	t0101	30	t0102	31	t0105	32	t0107
33	t0108	34	t1087	35	t1088	36	t1089	37	t0110	38	t0113	39	t1135	40	t1146
41	t1147	42	t1148	43	t1151	44	t1154	45	t1157	46	t1158	47	t1167	48	t1168
49	t0118	50	t1186	51	t1188	52	t1189	53	t0119	54	t1190	55	t1192	56	t1194
57	t1196	58	t1198	59	t1199	60	t1200	61	t1201	62	t1203	63	t1204	64	t1206
65	t1207	66	t1209	67	t0121	68	t1210	69	t1211	70	t1212	71	t1214	72	t1217
73	t0123	74	t0127	75	t0128	76	t1285	77	t1288	78	t0129	79	t1292	80	t1294
81	t0130	82	t1308	83	t0131	84	t1310	85	t1311	86	t1313	87	t1315	88	t1316
89	t1317	90	t1318	91	t1319	92	t1320	93	t1321	94	t1322	95	t1325	96	t1326
97	t1328	98	t1332	99	t1338	100	t1341	101	t1342	102	t1347	103	t1355	104	t0137
105	t1378	106	t1379	107	t0138	108	t1380	109	t1384	110	t1385	111	t1386	112	t1387
113	t1390	114	t1393	115	t1394	116	t1398	117	t0140	118	t1400	119	t1401	120	t1403
121	t1404	122	t1406	123	t1409	124	t1418	125	t1419	126	t1420	127	t1421	128	t1425
129	t1427	130	t1429	131	t1431	132	t1434	133	t0144	134	t1444	135	t1447	136	t1448
137	t1449	138	t0145	139	t1451	140	t1452	141	t1453	142	t0146	143	t0147	144	t1479
145	t0148	146	t1482	147	t1483	148	t1486	149	t1487	150	t1489	151	t0149	152	t1495
153	t1503	154	t1504	155	t1506	156	t1508	157	t1509	158	t0151	159	t1510	160	t1518
161	t0153	162	t1533	163	t1534	164	t1541	165	t0155	166	t1551	167	t1562	168	t1564
169	t1566	170	t1567	171	t1568	172	t0157	173	t1572	174	t1582	175	t1583	176	t1584
177	t1585	178	t1592	179	t1593	180	t1599	181	t1601	182	t1605	183	t1606	184	t1607
185	t1609	186	t0161	187	t0162	188	t1622	189	t0163	190	t1631	191	t1645	192	t1656
193	t1657	194	t1665	195	t1669	196	t0167	197	t1670	198	t1703	199	t1704	200	t1712
201	t1713	202	t1720	203	t1721	204	t1722	205	t1723	206	t1724	207	t1728	208	t1731
209	t0178	210	t0181	211	t0183	212	t0190	213	t0192	214	t0195	215	t0201	216	t0206
217	t0207	218	t0209	219	t0210	220	t0211	221	t0219	222	t0243	223	t0244	224	t0245
225	t0246	226	t0247	227	t0249	228	t0267	229	t0269	230	t0270	231	t0274	232	t0275
233	t0277	234	t0278	235	t0280	236	t0284	237	t0285	238	t0288	239	t0291	240	t0294
241	t0298	242	t0300	243	t0308	244	t0316	245	t0318	246	t0319	247	t0322	248	t0323
249	t0324	250	t0325	251	t0326	252	t0328	253	t0329	254	t0330	255	t0333	256	t0335
257	t0341	258	t0343	259	t0344	260	t0346	261	t0351	262	t0353	263	t0355	264	t0356
265	t0357	266	t0359	267	t0363	268	t0365	269	t0366	270	t0370	271	t0377	272	t0378
273	t0379	274	t0382	275	t0385	276	t0388	277	t0389	278	t0393	279	t0398	280	t0400
281	t0403	282	t0413	283	t0428	284	t0429	285	t0431	286	t0453	287	t0454	288	t0455
289	t0456	290	t0457	291	t0460	292	t0461	293	t0475	294	t0477	295	t0488	296	t0489
297	t0492	298	t0494	299	t0497	300	t0499	301	t0504	302	t0505	303	t0507	304	t0509
305	t0513	306	t0515	307	t0517	308	t0518	309	t0550	310	t0563	311	t0564	312	t0565
313	t0566	314	t0568	315	t0574	316	t0577	317	t0579	318	t0580	319	t0582	320	t0583
321	t0584	322	t0585	323	t0587	324	t0588	325	t0590	326	t0593	327	t0594	328	t0596
329	t0600	330	t0607	331	t0609	332	t0615	333	t0616	334	t0617	335	t0618	336	t0619
337	t0620	338	t0625	339	t0627	340	t0628	341	t0629	342	t0630	343	t0632	344	t0633
345	t0636	346	t0637	347	t0638	348	t0640	349	t0641	350	t0642	351	t0645	352	t0648
353	t0664	354	t0666	355	t0667	356	t0670	357	t0671	358	t0672	359	t0674	360	t0675
361	t0679	362	t0683	363	t0685	364	t0686	365	t0687	366	t0691	367	t0692	368	t0694
369	t0695	370	t0696	371	t0697	372	t0698	373	t0699	374	t0703	375	t0704	376	t0705
377	t0707	378	t0708	379	t0713	380	t0714	381	t0716	382	t0717	383	t0719	384	t0720
385	t0721	386	t0722	387	t0723	388	t0725	389	t0727	390	t0728	391	t0732	392	t0733
393	t0734	394	t0736	395	t0740	396	t0742	397	t0746	398	t0750	399	t0751	400	t0761
401	t0769	402	t0770	403	t0819	404	t0821	405	t0828	406	t0833	407	t0834	408	t0846
409	t0850	410	t0852	411	t0856	412	t0858	413	t0859	414	t0861	415	t0867	416	t0874
417	t0875	418	t0879	419	t0889	420	t0890	421	t0891	422	t0892	423	t0893	424	t0901
425	t0903	426	t0904	427	t0905	428	t0923	429	t0924	430	t0925	431	t0926	432	t0927
433	t0929	434	t0933	435	t0934	436	t0936	437	t0955	438	t0956	439	t0963	440	t0964