Lifelong Imitation Learning with Multimodal Latent Replay and Incremental Adjustment

Supplementary Material

This supplementary material includes extended implementation details of our method, as well as descriptions of the state-of-the-art methods used for comparison in Sec. A. Next, we present additional ablation studies that reflect and justify the design choices of our components and their parameter configurations in Sec. B. Sec. C offers further analyses on IFA and the choice of the task reference embedding, including an explanation of why angle-based computation can be more effective and how the adaptive parameter is designed. Finally, we report the computational efficiency of our method in Sec. D.

A. Extended Implementation Details

A.1. Our Framework

During multi-task pre-training, each task $\mathcal{T}'$ includes $N_j = 50$ expert demonstrations. In the lifelong learning stage, each new task $\mathcal{T}_k$ provides $N_k = 10$ demonstrations, following the setup used in prior state-of-the-art (SOTA) methods [23, 26]. Each rollout in the LIBERO suites consists of 70–300 time steps.

Our architecture follows the general design introduced in prior work [15], where each action is predicted from the previous $L = 8$ observations. Specifically, we employ CLIP-base [21] as both the vision encoder and the language encoder, each implemented as a 12-layer Transformer. A 6-layer GPT-2 decoder [20] serves as the temporal decoder, processing the visual and state tokens from the last eight steps and outputting the parameters of a 5-component Gaussian Mixture Model (GMM) over continuous actions. The use of a GMM policy head is a common practice in the LIBERO benchmarks [14]. Task instructions are injected via FiLM-based [18] conditioning of the visual and state representations within the modulation network. During pre-training, we train only LoRA adapters (with rank-8) in the CLIP backbones, together with the temporal decoder and the GMM policy head. In the lifelong learning stage, only the temporal decoder and policy head are updated for each new task, as depicted in Figure 2 in the main paper.

A.2. State of the Art

For the methods reported in Tab. 1 of the main paper, Sequential [14], Experience Replay (ER) [2, 14], BUDS [26, 33], and LOTUS [26] all use the same architectural setup and training configuration as LOTUS. Specifically, LOTUS employs a skill-discovery module to predict subgoal actions, which are then composed into a full action sequence to accomplish the task, and the numerical results for these methods are taken directly from the original LOTUS paper [26]. As for ISCIL [10], each skill is associated with its own adapter, and during evaluation the method retrieves the adapter corresponding to the skill most similar to the current input. For ISCIL, we use the implementation from [11], which has been adapted for the LIBERO setting. In this way, the same training setup as other methods is followed, ensuring fair comparisons.

We exploited an in-house implementation of TAIL [15] (note that the official code for TAIL [15] is not available). We verified that our implementation has the same number of parameters as reported in the TAIL paper, and therefore use it for all comparisons with confidence. As the official implementation is unavailable, we ensured architectural and training consistency with the original paper to maintain fair comparisons. We will make the corresponding code publicly available upon acceptance of our paper to foster future research in this direction.

TAIL [15] is a task-ID–based method that trains a separate adapter for each task to adapt the shared backbone to the corresponding data distribution. At test time, TAIL requires the task ID to select the correct adapter. To make TAIL compatible with the task-ID–agnostic LIL setting considered in the main paper, we first train it following the original procedure, without any modification. During evaluation, since multiple task-specific adapters are available at a given stage of the lifelong phase, we compute the success rate of each task as the average performance across all adapters available up to that point. This averaging simulates task-ID agnostic inference, where the correct adapter cannot be selected, and follows the same evaluation strategy adopted in LOTUS [26] when adapting task-incremental methods to LIL. All reported metrics use this averaged success rate, ensuring a fair comparison with LIL approaches that employ a single shared policy.

B. Additional Ablation Studies

B.1. Influence of the Angular Scaling Parameter α in IFA

We conducted an ablation study to evaluate the impact of different values of the angular scaling factor α in the adaptive IFA formulation (Tab. 7). The best-performing values of α are 0.3, 0.7, and 0.1 for LIBERO-OBJECT, LIBERO-GOAL, and LIBERO-50, respectively.

The adaptive margin δ is defined as proportional to

Table 7. Ablation on different angular scaling factor α in the adaptive IFA.

Dataset	α	FWT↑	NBT↓	AUC↑
LIBERO-OBJECT	0.10	81.8±3.2	12.4±3.6	77.3±1.4
	0.30	**84.6±1.9**	11.4±5.6	**79.4±1.5**
	0.50	82.1±0.7	18.3±10.3	73.2±4.1
	0.70	81.7±0.7	**8.4±3.3**	77.3±2.4
LIBERO-GOAL	0.10	**80.8±4.0**	12.0±11.3	75.4±8.8
	0.30	81.3±3.3	15.2±7.3	74.7±6.7
	0.50	79.2±1.9	12.8±2.8	73.1±2.3
	0.70	80.0±2.5	**6.9±0.9**	**77.2±1.8**
LIBERO-50	0.10	**60.8±2.8**	8.6±6.2	**56.1±1.8**
	0.30	53.6±4.0	**7.6±10.0**	52.0±1.4
	0.50	58.5±3.6	8.9±4.9	53.2±3.4
	0.70	60.5±2.2	10.6±1.1	54.2±2.3

Table 8. Comparison between LoRA adapters with different ranks (R) and full fine-tuning (FFT).

	LIBERO-OBJECT			LIBERO-GOAL		
	FWT↑	NBT↓	AUC↑	FWT↑	NBT↓	AUC↑
R=8	57.5±0.2	7.9±1.0	52.8±5.9	59.3±0.9	11.5±3.3	54.1±0.7
R=16	48.9±7.1	**-5.6±0.4**	50.4±6.9	56.3±5.3	**4.4±18.9**	53.6±2.4
R=32	53.8±0.2	-0.2±6.7	54.2±2.3	63.8±7.1	7.5±13.0	60.0±1.5
FFT	**84.6±1.9**	11.4±5.6	**79.4±1.5**	**80.0±2.5**	6.9±0.9	**77.2±1.8**

the angular distance between the reference embeddings of the current task and the most similar previous task: $\delta = \alpha \arccos\left(\frac{a^\top b}{|a|_2|b|_2}\right)$. Intuitively, this ensures that tasks with larger semantic differences are more strongly separated in the latent space, while similar tasks remain closer, preserving within-task coherence.

For LIBERO-50, the largest and most diverse benchmark, smaller α (0.1) gives the best overall performance, likely reflecting the denser arrangement of task reference embeddings—larger margins could over-penalize nearby tasks. However, performance at α = 0.7 remains comparable, indicating that the adaptive IFA is relatively robust to the choice of α even in large, densely packed task suites. This suggests that α can be tuned to balance inter-task separation and latent stability without critically affecting performance.

B.2. Adapters versus Full Fine-tuning.

During the lifelong learning stage, our default configuration fine-tunes all parameters of the temporal decoder. For comparison, we also evaluate a parameter-efficient variant based on Low-Rank Adaptation (LoRA) [5], where the temporal decoder is frozen and LoRA modules are inserted into the attention projection layers (query/key/value and output projections) and the MLP feed-forward layers. We experiment with different LoRA ranks, and the results are summarized in Tab. 8.

These results reveal a substantial performance gap: full fine-tuning significantly outperforms the LoRA-based adaptation across all datasets and metrics. This indicates that, in our setting, the temporal decoder requires sufficient capacity to capture complex temporal dynamics and integrate information coming from the MLR and IFA losses, which parameter-efficient approaches alone cannot provide. Furthermore, LoRA introduces additional hyperparameters that require grid search, making it less practical for large-scale lifelong learning scenarios.

Table 9. Effect of using FiLM layers during lifelong learning.

	LIBERO-OBJECT			LIBERO-GOAL		
	FWT↑	NBT↓	AUC↑	FWT↑	NBT↓	AUC↑
w/o FiLM	43.8±3.5	**8.8±7.7**	41.6±0.2	57.6±17.7	**0.83±7.5**	56.3±2.1
w FiLM	**84.6±1.9**	11.4±5.6	**79.4±1.5**	**80.0±2.5**	6.9±0.9	**77.2±1.8**

B.3. Influence of Using FiLM Layers.

In our model, the outputs of the vision and state encoders are modulated by FiLM layers [18] to obtain task-conditioned representations. To empirically justify their contribution, we ablate the model by removing FiLM layers during the lifelong learning stage (see Tab. 9).

As seen, incorporating FiLM layers significantly improves FWT and AUC compared to the variant without FiLM. Although the no-FiLM variant shows slightly lower NBT, this is due to its overall weaker learning performance rather than genuine resistance to forgetting. These results indicate that explicit feature-wise modulation is crucial for effective adaptation: by dynamically scaling and shifting intermediate activations, FiLM allows the network to adjust its internal representations efficiently to the conditioning signal, improving forward transfer and overall learning effectiveness.

C. Extended Analysis

C.1. Selection of a stable task reference

As described in the main paper, our proposed Incremental Feature Adjustment (IFA) is a regularization technique designed to counter representation drift across tasks during Lifelong Learning (LIL). IFA achieves this by ensuring that the global latent representation ($g_t(T_k)$), which is the output of the temporal decoder for the current task T_k, remains closer to its own task reference ($h^{(r)}(T_k)$) than to the references of previously learned tasks ($h^{(r)}(T_j)$). This introduces repulsive forces between the current task's global latent representation and previous task references, promoting inter-task disentanglement and preserving within-task coherence.

The efficacy of IFA relies critically on selecting a stable

Table 10. Task-reference comparison on **LIBERO-OBJECT** and **LIBERO-GOAL**. See text for the explanation of "Mean Global" and "Global". "Language", "AgentView", "Eye-in-hand", and "State" use the corresponding modality-specific latent feature as the task reference. The smallest cosine distance is highlighted in **bold**, and the second smallest is underlined.

	LIBERO-OBJECT				LIBERO-GOAL			
Task Reference	**Task 6**	**Task 7**	**Task 8**	**Task 9**	**Task 6**	**Task 7**	**Task 8**	**Task 9**
Mean global	**0.565**	**0.555**	**0.549**	**0.454**	**0.485**	**0.354**	**0.501**	**0.437**
Global	0.811	0.802	0.797	0.703	0.735	0.583	0.752	0.684
Language	0.668	0.701	0.656	0.677	0.591	0.610	0.629	0.680
AgentView	0.947	0.951	0.973	0.994	0.976	0.957	0.980	1.051
Eye-in-hand	1.008	0.974	1.005	1.005	1.018	1.037	0.987	1.042
State	1.068	1.012	1.024	1.006	1.033	1.072	1.005	1.036

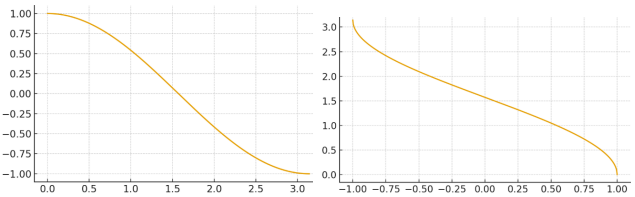
Figure 5. UMAP visualization of the global latent representations $g(T_k)$ obtained when not enforcing the IFA loss, each color represents the global latent representations of one task.

and representative task reference ($h^{(r)}$) for each task. The motivation for this selection can be seen from the visualization in Fig. 5: we observe that the global latent representations of a given task naturally cluster together in the embedding space. This clustering indicates that each task can be summarized by a single, representative point that reflects the overall location of its features.

To justify our choice of the language latent feature as the task reference ($h^{(r)}$), we compared it against other viable candidates. We computed the average cosine distance between these candidates and the global latent representations of their corresponding tasks (on LIBERO-OBJECT and LIBERO-GOAL, Tasks 6–9), evaluating both their representativeness and their stability during incremental learning. The task reference candidates evaluated were the mean latent feature derived from different modalities (first column of Tab. 10): Mean global, the mean of all global latent representations for a task; Global, a measure derived by averaging the distances obtained when treating each individual global latent representation as a separate, unstable candidate reference; Modality-Specific Latent Features, the mean latent feature of a specific modality, such as language, agent view, eye-in-hand, or state (see Section 4). For example, when using the agent-view modality, we take the mean agent-view latent features of the task as the task reference and compute its average cosine distance to all global latent representations of that task (see Tab. 10). This data confirmed that using the mean of all global latent representations (the Global Latent Feature candidate) yields the smallest cosine distance, proving it is the most statistically representative point. Despite being the most representative, the mean global reference is fundamentally unstable. During the ongoing process of Lifelong Incremental Learning (LIL), the dynamic changes in model parameters cause the latent representations, and consequently their mean, to change over time. This drifting reference hinders the ability of IFA to consistently regulate representation drift. Crucially, the language latent feature produced the second-smallest distances. Furthermore, the language-based task reference (which is identical to its mean since the language modality provides a single, fixed embedding per task description) remains stable throughout training. It even outperformed the unstable "Global" baseline. Given that a stable anchor is essential for IFA to effectively prevent representation drift across tasks, we adopt the language latent feature as our definitive task reference ($h^{(r)}$), prioritizing its stability for reliable regularization over the slightly higher representativeness of the unstable mean global feature.

C.2. Extended details for MLR and IFA.

MLR. In the lifelong learning stage, we utilize the proposed Multimodal Latent Replay (MLR). Instead of storing raw sensory data, a compact buffer $\mathcal{B}$ maintains the multimodal latent representations ($\mathbf{H}$), which are concatenated from the frozen encoders, alongside the associated action (a) for training. In subsequent tasks, the stored $\mathbf{H}$ from previous tasks is fed into the temporal decoder for replay.



(a) $\cos(\theta)$ on $[0, \pi]$. (b) $\arccos(x)$ on $[-1, 1]$.

Figure 6. Illustration of the cosine function and its inverse.

IFA. To prevent interference (catastrophic forgetting) between the current task (T_k) and the previously learned tasks (T_j), we introduce the IFA. IFA's core mechanism encourages the global latent representation of the current task ($g_t(T_k)$) to remain closer to its own stable reference ($h^{(r)}(T_k)$) than to the references of previously learned tasks ($h^{(r)}(T_j)$).The IFA loss is explicitly defined to penalize violations of this distance constraint:

$$\mathcal{L}_{\text{IFA}} = \frac{1}{|\mathcal{P}|} \sum_{\substack{(j,k)\in\mathcal{P} \\ j<k}} \max\Big(0,\, d\big(g_t(T_k), h^{(r)}(T_k)\big) - d\big(g_t(T_k), h^{(r)}(T_j)\big) + \delta\Big), \tag{9}$$

The margin δ adapts to the relative positions of the task references themselves:

$$\delta = \alpha\, d\Big(h^{(r)}(T_k),\, h^{(r)}(T_j)\Big) \quad , \tag{10}$$

Angle-based distance. The IFA loss is explicitly driven by the difference in distance between the global latent representation $g_t(T_k)$ and the task references: the positive reference $h^{(r)}(T_k)$ (for the current task T_k) and the negative references $h^{(r)}(T_j)$ (for the previous tasks T_j). Therefore, we compared how the standard cosine similarity and its angular counterpart (distance $d(\cdot,\cdot) = \arccos(\cdot)$) behave when used inside this crucial distance difference.

Fig. 6 illustrates this comparison. The left panel plots $\cos(\theta)$ as a function of the angle $\theta \in [0, \pi]$ between two unit vectors, while the right panel shows $\arccos(x)$ as a function of $x \in [-1, 1]$. Cosine (left) quickly saturates in the high-similarity regime: when two representations are already very close, even noticeable angular changes produce almost no change in $\cos\theta$, making the resulting distance difference extremely small and hard to distinguish. In contrast, the angular distance $\arccos(x)$ (right) expands this high-similarity region; small changes in x near 1 translate into clearly separable angular differences, providing more stable resolution across similarity levels and allowing IFA to better capture small deviations between task-level representations. Outside this range, $\arccos(x)$ behaves similarly to cosine similarity—differences between representations remain distinguishable.

Table 11. Runtime and FLOPs comparison. See text for the explanation of Forward Time, S-FLOPs, and T-FLOPs. All experiments are conducted on **LIBERO-GOAL**. P means the probability of storing features in the buffer, which is ablated in Tab. 5. RR stands for raw replay, which stores past samples and replays them during training. Note that BASE stands for running our method without MLR or IFA. It is included solely to quantify the computational cost of the base architecture.

Method	P	Forward Time (ms)	S-FLOPs	T-FLOPs
BASE	-	75.5 ± 2.43	3.63×10^{11}	4.36×10^{17}
BASE + RR	0.5	75.8 ± 2.80	3.63×10^{11}	12.96×10^{17}
BASE + MLR	0.5	75.5 ± 2.43	3.63×10^{11}	7.62×10^{17}
BASE + MLR + IFA	0.5	75.8 ± 2.80	3.63×10^{11}	7.62×10^{17}
BASE + MLR + IFA	0.2	75.8 ± 2.43	3.63×10^{11}	5.66×10^{17}
BASE + MLR + IFA	0.1	75.8 ± 2.43	3.63×10^{11}	5.01×10^{17}

Adaptive δ. The choice of δ as a fixed hyperparameter would incur in a risk of choosing an overly large margin. Specifically, if δ is set larger than the distance between the two task references, i.e., $\delta > d\big(h^{(r)}(T_k), h^{(r)}(T_j)\big)$, the IFA loss will always enforce a penalty. This is guaranteed by the triangle inequality property of spherical geometry, which states that the difference between the distances to the two references is bounded by the distance between the references themselves:

$$d\big(g_t(T_k), h^{(r)}(T_k)\big) - d\big(g_t(T_k), h^{(r)}(T_j)\big) \le d\big(h^{(r)}(T_k), h^{(r)}(T_j)\big),$$

If δ exceeds this maximum possible difference, the loss term inside the $\max(\cdot)$ will always be positive. This would enforce an unnecessary repulsive force, even when the current global latent representation $g_t(T_k)$ is already sufficiently closer to its own reference $h^{(r)}(T_k)$, potentially destabilizing learning.

To avoid this unnecessary penalty and allow the margin to adapt to the distance between references, we define δ dynamically:

$$\delta = \alpha\, d\big(h^{(r)}(T_k), h^{(r)}(T_j)\big), \quad \alpha \in (0, 1).$$

This guarantees that the margin is always bounded by the inter-reference distance, and scales proportionally to how far apart the tasks are in the angular space.

D. Computational efficiency

We report the computational efficiency using several primary metrics related to policy execution in Tab. 11. Forward Time refers to the time required to produce one action (i.e., a single forward pass). We computed forward time

using a NVIDIA A100 GPU, by first running a warming up stage of 10 inferences (to avoid transient effects from initialization and caching), and then computing the average and standard deviation on 100 subsequent inferences.

The S-FLOPs correspond to the total number of Floating-Point Operations (FLOPs) required for a single forward pass of the policy to produce one action. This metric reflects model inference cost and is independent of data or training procedure. For completeness, we also report T-FLOPs, the total training FLOPs, computed as per-forward FLOPs multiplied by the total number of forward pass, backward pass, and gradient update during training (i.e., FLOPs × epochs × training sampless × 3).

As shown in the result table, both MLR and IFA have a minimal overhead on per-inference efficiency, indicating that their additional computations introduce negligible runtime cost. In terms of total training FLOPs, adding MLR (P = 0.5) increases the computational cost by roughly $0.75\times$ on **LIBERO-GOAL** due to replay, and the less the buffer is, the less the T-FLOPs. But this increase remains reasonable and acceptable given the resulting performance gains compared to the performance in Tab. 5.