A Detailed Explanation of the PEFT-CL Framework

A.1 Specific Methods within the PEFT-CL Framework

L2P L2P maintains a fixed pool of m prompt vectors and, for each example, uses the frozen backbone feature $\Phi(x)$ as the retrieval query. It computes the score vector

$$s = \frac{\Phi(x)\, K^\top}{\tau} \in \mathbb{R}^m,$$

applies a hard top-k operator to obtain a sparse, one-hot–like weight $g \in \{0,1\}^m$, and concatenates the selected prompts into the input. In our PEFT-CL framework, this corresponds to $x_{\text{rep}} = \Phi(x)$, $g(s) = \text{top-}k\{s\}$ at both train and test time, and a frozen key matrix K, so that

$$h \;\leftarrow\; h + \text{top-}k\{s\}\, \Delta H.$$

DualPrompt DualPrompt extends L2P by maintaining two disjoint pools—"general" and "domain-specific"—and by using a one-hot teacher signal δ_t during training to force selection of the correct domain prompt. At train time it sets

$$g\big(\Phi(x)K; t\big) = \delta_t,$$

and at test time uses

$$g\big(\Phi(x)K\big) = \arg\max_i \big[\Phi(x)K^\top\big]_i,$$

both yielding a one-hot weight vector. Under our formulation, $x_{\text{rep}} = \Phi(x)$, the branch outputs ΔH are mixed by these one-hot weights, and K remains frozen—thus decoupling routing from the cross-entropy loss—so that

$$h \;\leftarrow\; h + g\, \Delta H.$$

CoDA-Prompt CoDA-Prompt replaces hard, discrete routing with a fully differentiable soft router. Given the frozen feature $\Phi(x)$, it computes the same score $s = \Phi(x)K^\top/\tau$ but applies

$$g(s) = \text{softmax}(s) \in \Delta^{m-1}$$

to produce a dense mixture weight over all m prompt branches. Importantly, both K and the prompt parameters ΔH are updated via the downstream cross-entropy loss L_{CE}. In our unified framework this corresponds to $x_{\text{rep}} = \Phi(x)$, a trainable, differentiable routing function g, and joint optimization of K, yielding

$$h \;\leftarrow\; h + \text{softmax}(s)\, \Delta H.$$

HiDe-Prompt HiDe-Prompt refines the query representation itself by learning a lightweight adapter $\widetilde{\Phi}(x)$ on top of the frozen backbone. It then reuses the DualPrompt routing strategy—one-hot teacher δ_t during training and $\arg\max$ at inference—while keeping K frozen. Thus, in our PEFT-CL notation one sets $x_{\text{rep}} = \widetilde{\Phi}(x)$ and

$$g\big(\widetilde{\Phi}(x)K; t\big) = \delta_t, \quad g\big(\widetilde{\Phi}(x)K\big) = \arg\max\big(\widetilde{\Phi}(x)K\big),$$

so that

$$h \;\leftarrow\; h + g\, \Delta H,$$

with all other design choices—single retrieval, hard routing, frozen key—identical to DualPrompt.

A.2 Instantiation of ϕ for common PEFT modules.

For completeness, we spell out the concrete functional form of $\phi(x;\theta^{(i)})$ for several standard PEFT blocks, where $x \in \mathbb{R}^d$ denotes the layer hidden state and $\Delta h^{(i)} = \phi(x;\theta^{(i)})$.

Prefix Tuning. Given a query projection $W_q^{(i)}$, a key prefix matrix $P_k^{(i)}$ and a value prefix matrix $P_v^{(i)}$, the residual contributed by the i-th prefix block is

$$\phi_{\text{Prefix}}(x;\theta^{(i)}) = \text{softmax}\big(xW_q^{(i)}{P_k^{(i)}}^\top\big)\,P_v^{(i)}. \tag{18}$$

Adapter. For a bottleneck adapter with down- and up-projection matrices $W_{\text{down}}^{(i)}$ and $W_{\text{up}}^{(i)}$, we have

$$\phi_{\text{Adapter}}(x;\theta^{(i)}) = \text{ReLU}\big(xW_{\text{down}}^{(i)}\big)\,W_{\text{up}}^{(i)}. \tag{19}$$

LoRA. For a low-rank LoRA block parameterized by $W_{\text{down}}^{(i)}$ and $W_{\text{up}}^{(i)}$, the residual update is

$$\phi_{\text{LoRA}}(x;\theta^{(i)}) = x\,W_{\text{down}}^{(i)}W_{\text{up}}^{(i)}. \tag{20}$$

These instantiations are all special cases of our unified notation $\Delta h^{(i)} = \phi(x;\theta^{(i)})$ used in the main text.

B Pseudocode

Algorithm 1 Latent Deliberation: Iterative Retrieval and Integration

Require: Backbone depth L, max iterations $T_{\max}$, tolerance ε, temperature T, blend factor α, (optional) Top-k

for $l = 1$ **to** L **do**
  Input: previous hidden state $h^{(l-1)} \in \mathbb{R}^d$
  Initialize query: $q^{(1)} \leftarrow h^{(l-1)}$
  Initialize empty list $\mathcal{V} \leftarrow []$
  for $t = 1$ **to** $T_{\max}$ **do**
    Compute retrieval weights: $S^{(t)} \leftarrow \text{softmax}\big(q^{(t)}(K^{(l)})^\top/T\big)$
    if Top-k enabled **then**
      Keep only top-k entries of $S^{(t)}$, zero out others
    end if
    Retrieve memory: $v^{(t)} \leftarrow S^{(t)}\,V^{(l)}$
    Append $v^{(t)}$ to $\mathcal{V}$
    Update query: $q^{(t+1)} \leftarrow \alpha\,q^{(t)} + (1-\alpha)\,P^{(l)}\big(v^{(t)}\big)$
    if $t > 1$ **and** $\|v^{(t)} - v^{(t-1)}\|^2 < \varepsilon$ **then**
      break
    end if
  end for
  One-shot fusion: $V_{\text{cat}} \leftarrow \text{concat}(\mathcal{V}) \in \mathbb{R}^{T_{\max}\,d_v}$
  Compute layer output: $h^{(l)} \leftarrow \text{ViT}^{(l)}\big([\,h^{(l-1)} \parallel V_{\text{cat}}\,]\big)$
end for

C Proof of Proposition 1

Proof of Proposition 1. We regard one gradient-descent "inner" step as a map

$$F(q;\theta) \;=\; q - \eta\,\nabla\phi(q), \qquad q^{(t+1)} = F\big(q^{(t)};\theta\big),$$

where θ denotes *outer* parameters that may affect the step only through some auxiliary operation (e.g. the retrieval projection in our Latent Deliberation loop). Let $q^\star$ be a fixed point of the inner dynamics so that $\nabla\phi(q^\star) = 0$. Linearising F at $(q^\star, \theta)$ gives the *state Jacobian* $J = \frac{\partial F}{\partial q}\Big|_{q^\star} = I - \eta\,\nabla^2\phi(q^\star) = I - \eta H$ and the *parameter Jacobian* $P = \frac{\partial F}{\partial \theta}\big|_{q^\star}$.

Step 1: Recursion on sensitivities. Differentiating the inner recurrence w.r.t. θ yields

$$\frac{\partial q^{(t+1)}}{\partial \theta} \;=\; J\,\frac{\partial q^{(t)}}{\partial \theta} \;+\; P, \qquad t = 0, \dots, T_{\max} - 1,$$

with the base term $\frac{\partial q^{(0)}}{\partial \theta} = 0$ because the initial hidden state is taken as constant for the outer optimisation. Solving the first-order, non-homogeneous linear recursion gives

$$\frac{\partial q^{(T_{\max})}}{\partial \theta} \;=\; \sum_{k=0}^{T_{\max}-1} J^k\,P.$$

Step 2: Chain rule for the outer loss. Applying the chain rule,

$$\nabla_\theta \mathcal{L}_{T_{\max}} \;=\; \Big(\tfrac{\partial q^{(T_{\max})}}{\partial \theta}\Big)^\top g_{\text{out}} \;=\; \sum_{k=0}^{T_{\max}-1} (J^\top)^k P^\top g_{\text{out}} \;=\; \sum_{k=0}^{T_{\max}-1} (J^\top)^k\, b_\theta,$$

where $b_\theta := P^\top g_{\text{out}}$. This is exactly the Krylov series operator $\mathcal{K}_{T_{\max}}(H) = \sum_{k=0}^{T_{\max}-1}(I - \eta H^\top)^k$ acting on b_θ.

Step 3: Convergence to the Neumann–series inverse. Because H is symmetric positive definite and $0 < \eta < 2/\lambda_{\max}(H)$, the spectral radius $\rho(J) = \rho(I - \eta H) < 1$. Hence the Neumann series converges:

$$\lim_{T_{\max}\to\infty} \mathcal{K}_{T_{\max}}(H) \;=\; (I - J^\top)^{-1} \;=\; \big(\eta\,H^\top\big)^{-1}.$$

Taking the limit in the gradient expression gives $\nabla_\theta \mathcal{L}_\infty = H^{-1}\,b_\theta$, which corresponds to exact second-order (Newton–style) preconditioning.

Step 4: Finite-step interpretation. For any finite $T_{\max}$, $\mathcal{K}_{T_{\max}}(H)$ is a degree-$(T_{\max}-1)$ polynomial that approximates H^{-1} in the Krylov subspace $\operatorname{span}\{b_\theta, H^\top b_\theta, \dots, (H^\top)^{T_{\max}-1} b_\theta\}$. Because the error decays geometrically in $\rho(J)$, practitioners often find $T_{\max} = 2$–4 iterations already provide most of the curvature correction at only linear cost.

This completes the proof. □

D Evaluation Metrics and Hyperparameters

Our code is available at https://github.com/TODO-YOUR-REAL-REPO/HippoTune and also included in the supplementary material archive.

Accuracy (Acc) For any evaluation point i and task j, the *accuracy* is simply

$$\mathrm{Acc}(i, j) = a_{i,j},$$

i.e. the test accuracy on task j immediately after learning task i.

Average Accuracy Across Tasks (AAA) At the end of task i, the *average accuracy* across all seen tasks is

$$\mathrm{AAA}(i) \;=\; \frac{1}{i}\sum_{j=1}^{i} a_{i,j}.$$

In particular, $\mathrm{AAA}(T)$ summarizes overall performance after the final task.

Forgetting Measure (FM) For each earlier task $j < i$, its *forgetting* after learning up to task i is

$$f_j^i = \max_{1 \le l < i} a_{l,j} \; - \; a_{i,j},$$

i.e. the largest drop from its best-seen accuracy to its current accuracy. The *average forgetting* at the end of all tasks is then

$$\mathrm{FM} \; = \; \frac{1}{T-1} \sum_{j=1}^{T-1} f_j^T.$$

A smaller FM indicates less catastrophic forgetting.

We now introduce the hyperparameters one by one, including key hyperparameters of the ViT backbone as well as those specific to our HippoTune training procedure.

learning_rate Initial learning rate used by the optimizer.

batch_size Number of samples processed in each training batch.

epoch Number of full passes over the training dataset.

pretrained Whether to load pretrained backbone weights (1: yes; 0: no).

clip_grad Maximum norm for gradient clipping to prevent exploding gradients.

size Number of prompt vectors in the prompt pool.

length Length of each prompt vector (in tokens).

initializer Initialization scheme for prompt values (e.g. `uniform` for uniform distribution).

prompt_key_init Initialization scheme for prompt keys (e.g. `uniform`).

batchwise_prompt Whether to share the same prompt across the entire batch (1: yes; 0: per-example).

global_pool Pooling method over ViT outputs: `token` (use class token) or `avg` (global average pooling).

head_type Input to the classification head: `token`, `gap` (global average pooling), `prompt`, or `token+prompt`.

freeze List of backbone submodules to freeze during training, e.g. `[blocks, patch_embed, cls_token, norm, pos_embed]`.

$\boldsymbol{\lambda_{\text{orth}}}$ Weight of the orthogonality regularization term on prompt keys.

layer_idx Indices of Transformer layers where prompting/retrieval is applied, e.g. `[0,1,2,3,4,5,6]`.

$\boldsymbol{T}$ Softmax temperature for computing attention weights over keys.

delib_steps ($T_{\max}$) Maximum number of latent deliberation (iterative retrieval) steps.

$\boldsymbol{\alpha}$ Query blending ratio in each iteration: $q_{t+1} = \alpha\, q_t + (1-\alpha)\, P(v_t)$.

eps_stop Early-stop threshold on $\|v_t - v_{t-1}\|$ for terminating latent deliberation.

topk Number of top keys to retain for sparse retrieval; `None` means full softmax.

fuse Method to combine outputs across steps: `mean` (average) or `last`.

$\boldsymbol{\lambda_{\text{ent}}}$ Weight for entropy regularization on the retrieval distribution.

Dataset	pretrained	clip_grad	size	length	initializer	prompt_key_init	batchwise_prompt	lr	batch_size	epochs
Seq-CIFAR100	1	1.0	30	10	uniform	uniform	1	0.001	128	5
Seq-ImageNet-R	1	1.0	30	10	uniform	uniform	1	0.001	128	5
Seq-CUB200	1	1.0	30	10	uniform	uniform	1	0.001	128	5

(a) Basic Training Hyperparameters

Dataset	global_pool	head_type	freeze
Seq-CIFAR100	token	token	[blocks, patch_embed, cls_token, norm, pos_embed]
Seq-ImageNet-R	token	token	[blocks, patch_embed, cls_token, norm, pos_embed]
Seq-CUB200	token	token	[blocks, patch_embed, cls_token, norm, pos_embed]

(b) ViT-related Hyperparameters

Dataset	λ_{orth}	layer_idx	T	delib_steps(T_{max})	α	eps_stop	topk	fuse	λ_{ent}
Seq-CIFAR100	1.0	[0,1,2,3,4,5,6]	0.01	4	0.2	1e-5	5	mean	1
Seq-ImageNet-R	1.0	[0,1,2,3,4,5,6]	0.01	4	0.2	1e-5	5	mean	1
Seq-CUB200	1.0	[0,1,2,3,4,5,6,7,8]	0.01	4	0.2	1e-5	5	mean	1

(c) Hyperparameter Settings for Multi-Key Retrieval and Latent Deliberation

E Additional Experiments

E.1 Online Continual Learning

Our method remains highly effective in the strictly online setting with a single data pass: on Seq-CIFAR100, **HippoTune-PreT** achieves **84.52%** $\pm$ **0.23** Acc, **89.09%** $\pm$ **0.21** AAA, and **7.48** $\pm$ **0.17** FM—only $\sim$2.8% below its offline (5-epoch) result and clearly outperforming buffer-free baselines such as L2P (75.38% $\pm$ 1.05 Acc) and DualPrompt (80.89% $\pm$ 0.58 Acc) (Table 5). On Seq-CUB200, it attains **65.99%** $\pm$ **0.24** Acc, **74.52%** $\pm$ **0.64** AAA, and **3.55** $\pm$ **0.35** FM, surpassing CODA-Prompt's 62.63% $\pm$ 0.34 Acc. The efficient variant **HippoTune-PreT-E** also yields strong performance (84.07% $\pm$ 0.28 Acc, 88.62% $\pm$ 0.36 AAA, 8.07 $\pm$ 0.15 FM on Seq-CIFAR100; 64.69% $\pm$ 0.53 Acc, 72.94% $\pm$ 0.48 AAA, 4.19 $\pm$ 0.32 FM on Seq-CUB200). Moreover, a sharing-favored prompt allocation ($T = 1$) outperforms an isolation-favored one ($T = 0.01$) by $\sim$0.8% on Seq-CIFAR100 and $\sim$1.3% on Seq-CUB200, indicating that emphasizing shared knowledge significantly aids convergence in the challenging one-pass online regime.

E.2 Effectiveness on Diverse Pre-trained Backbones

Experiments using DINO and SAM backbones further demonstrate the strong generalization of HippoTune. As shown in Table 6, our method consistently achieves superior final and average accuracy across both architectures, significantly outperforming baselines like L2P and DualPrompt. Notably, it surpasses CODA-Prompt by over 4% on the SAM backbone. These results indicate that the latent iterative deliberation mechanism is architecture-agnostic and adapts well to feature distributions from diverse pre-training objectives, effectively leveraging heterogeneous representations while minimizing forgetting.

E.3 Stability and Backward Transfer Analysis

As shown in Table 7, in experiments on ImageNet-R split into 10 tasks, HippoTune outperforms standard prompt-based baselines in both accuracy and retention. While LoRA and adapter-based methods such as SD-Lora and EASE achieve high plasticity due to their architectural capacity, they suffer from significant catastrophic forgetting. In contrast, HippoTune maintains a Forgetting Measure of 4.03%, which is significantly lower than the 6% to 7% range observed in these adapter variants. This demonstrates that our method offers superior stability and effectively mitigates the interference common in high-capacity adapter approaches.

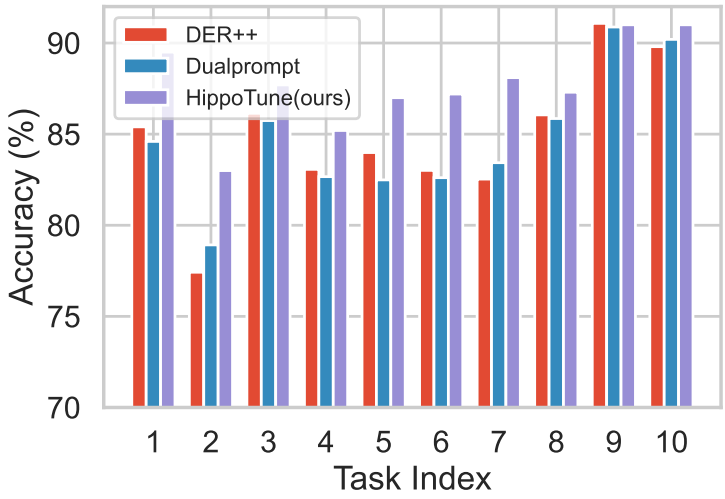

Figure 4: **Per-task accuracy comparison across 10 sequential tasks.** We report the task-wise Accuracy (%) of DER++, Dualprompt, and our HippoTune method on Seq-CIFAR100. HippoTune consistently outperforms the baselines, especially on early and late tasks, demonstrating its effective latent deliberation mechanism.

E.4 Accuracy Comparison Across All Tasks After Training

We provide a detailed visualization of the accuracy evolution across all tasks to evaluate the stability of the model throughout the training process. As shown in Figure 4, HippoTune consistently maintains a significant performance advantage over the baseline methods across the entire sequence.

E.5 Results in the Task-Incremental Setting

Table 8 presents the comparative results under the Task-Incremental Learning (TIL) setting. HippoTune consistently outperforms the PEFT-CL baselines across both Seq-CIFAR100 and ImageNet-R benchmarks. While existing methods like CODA-Prompt already mitigate interference by conditioning on task identities, our approach further pushes the performance boundary, achieving the highest average accuracy and the lowest forgetting measures. This superiority indicates that HippoTune effectively leverages task-specific contexts to refine feature representations, ensuring robust learning of new tasks without compromising the stability of previously acquired knowledge.

Table 5: Comparison of Continual Learning Methods on Seq-CIFAR100 and Seq-CUB200 in Terms of Accuracy, Average Accuracy Across All Tasks (AAA), and Forgetting Measure (FM).

Method	Seq-CIFAR100			Seq-CUB200		
	Acc	AAA	FM	Acc	AAA	FM
PEFT-CL (w/o buffer)						
L2P	75.38±1.05	84.38±0.58	10.17±0.62	60.78±0.42	69.21±0.46	5.37±0.23
DualPrompt	80.89±0.58	86.74±0.37	10.32±0.55	62.79±0.27	71.25±0.53	4.87±0.25
CODA-Prompt	82.68±0.39	88.01±0.46	9.96±0.53	62.63±0.34	71.63±0.41	4.79±0.52
Ours (w/o buffer)						
HippoTune	**84.52±0.23**	**89.09±0.21**	**7.48±0.17**	**65.99±0.24**	**74.52±0.64**	**3.55±0.35**

Table 6: Performance comparison of PEFT-CL methods and HippoTune on ImageNet-R (N=10) with DINO and SAM backbones (metrics: Acc, AAA, FM).

Method	ImageNet-R (N=10, Dino)			ImageNet-R (N=10, SAM)		
	Acc	AAA	FM	Acc	AAA	FM
PEFT-CL						
L2P	51.12	62.68	**1.34**	56.93	59.38	6.46
DualPrompt	60.18	66.91	4.40	65.13	70.05	5.29
CODA-Prompt	63.74	69.07	5.43	66.61	71.63	5.72
Ours						
HippoTune	**65.62**	**70.29**	3.78	**70.92**	**75.54**	**5.16**

Table 7: Forgetting rates (FM) and backward transfer (bwt).

Metric	l2p	dualprompt	codaprompt	sdlora	ease	ranpac	hippotune
FM	4.11	5.18	5.39	7.34	6.37	4.62	4.03
bwt	-4.03	-5.18	-5.26	-7.24	-6.21	-4.56	-3.94

Table 8: Task-Incremental Learning (TIL) performance comparison on Seq-CIFAR100 and ImageNet-R. We report Average Accuracy (Acc), Average After Accuracy (AAA), and Forgetting Measure (FM). Best results are highlighted in **bold**.

Method	Seq-CIFAR100			ImageNet-R		
	Acc	AAA	FM	Acc	AAA	FM
PEFT-CL						
L2P	96.84	97.04	0.57	89.45	89.24	0.53
DualPrompt	97.25	97.52	0.62	87.72	88.21	0.71
CODA-Prompt	97.83	98.04	0.50	89.57	89.66	0.52
Ours						
HippoTune	**98.54**	**98.68**	**0.44**	**90.43**	**90.41**	**0.48**

F THE USE OF LARGE LANGUAGE MODELS (LLMS)

In this paper, large language models (LLMs) were used only as general-purpose assistive tools for language polishing, improving writing structure, and retrieving and organizing references. LLMs did not contribute to research ideation, method design, experiment execution, or result analysis, and thus do not constitute a substantive scholarly contribution.