PRESERVING GENERAL CAPABILITIES WITHOUT PAST DATA: PARAMETER-IMPORTANCE-DRIVEN CONTINUAL LEARNING FOR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Domain-specific post-training often causes catastrophic forgetting, making foundation models lose their general reasoning capability and limiting their adaptability to dynamic real-world environments. Preserving these capabilities while acquiring domain knowledge remains a key challenge for large language and multimodal models. Traditional continual learning methods, such as regularization, replay and architectural isolation, suffer from poor downstream performance, reliance on inaccessible historical data, or additional parameter overhead. While recent parameter-efficient tuning (PET) methods can alleviate forgetting, their effectiveness strongly depends on the choice of parameters and update strategies. In this paper, we introduce PIECE, a Parameter Importance Estimation-based Continual Enhancement method that preserves general ability while efficiently learning domain knowledge without accessing prior training data or increasing model parameters. PIECE selectively updates only 0.1% of core parameters most relevant to downstream tasks, guided by two importance estimators: PIECE-F based on Fisher Information, and PIECE-S based on a second-order normalization that combines gradient and curvature information. Experiments across three language models and two multimodal models show that PIECE maintains general capabilities and achieves state-of-the-art continual learning performance across diverse downstream tasks. Our results highlight a practical path to scalable, domain-adaptive foundation models without catastrophic forgetting.

1 INTRODUCTION

Foundation models, including large language models (LLMs) and multimodal LLMs (MLLMs), acquire broad knowledge and generalizable capabilities through large-scale pretraining (Li et al., 2025b). However, domain-specific fine-tuning often induces catastrophic forgetting (Liu et al., 2020), degrading general reasoning abilities and broad knowledge, and thereby limiting effective domain adaptation (Luo et al., 2025).

Continual learning (Shi et al., 2025) has been recognized as an effective mechanism for enabling models to incrementally acquire new domain knowledge. **Regularization methods** (Lopez-Paz & Ranzato, 2017; Kirkpatrick et al., 2017) constrain model updates near the original parameters, potentially limiting efficient downstream adaptation. **Replay methods** (Rebuffi et al., 2017; Rolnick et al., 2019) require historical training data, making them impractical for post-training foundation models (Touvron et al., 2023) and inefficient due to costly large-scale replay. **Architectural isolation methods** (Tong et al., 2025; Zhao et al., 2024b) mitigate forgetting by separating different tasks through mechanisms such as gating, but they introduce additional parameter overhead and impede cross-task generalization (Verma et al., 2021). To compare these methods, we take programming ability as a proxy for the general capabilities of foundation models, because it is particularly vulnerable to destruction (Chen et al., 2025). Figure 1 (b) evaluates the forgetting behaviors of these traditional approaches, showing that both Regularization and Replay methods[1] fail to meet the crucial requirement of preserving the broad and general capabilities.

[1]We focus on continual learning without parameter growth, excluding architectural isolation methods.

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.

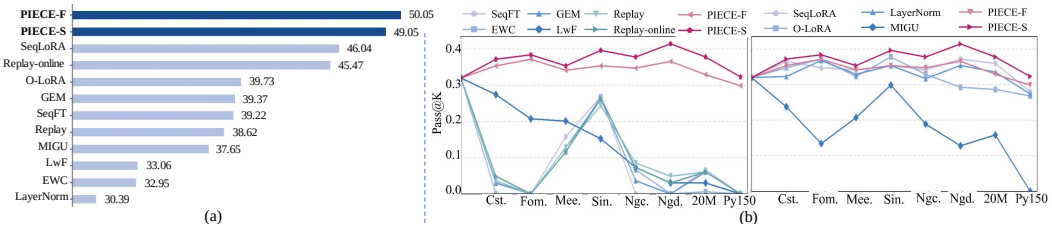


Figure 1: (a) Average downstream-task scores and (b) HumanEval Pass@K (K=1) of Llama3-8B on TRACE as tasks accumulate. PIECE consistently outperforms full fine-tuning (SeqFT), regularization (EWC, GEM, LwF), replay (Replay, Replay-online), and PET (SeqLoRA, O-LoRA, LayerNorm, MIGU) baselines in both downstream performance and capability retention.

Recently, parameter-efficient tuning (PET) methods (Du et al., 2024; Wang et al., 2023a; Hu et al., 2022; Zhao et al., 2024a) have demonstrated the potential for anti-forgetting learning (Sung et al., 2021; Xu et al., 2021) without historical data and labels, or modifying model architecture. As shown in Figure 1, **(1) PET outperforms traditional continual learning approaches in preserving the general capabilities of the model.** In contrast, regularization and replay methods struggle with catastrophic forgetting due to the lack of access to historical data. **(2) Different PET strategies exhibit varying levels of general capability preservation.** SeqLoRA (Hu et al., 2022), which updates only Attention components, outperforms LayerNorm tuning (Zhao et al., 2024a), suggesting that domain knowledge may be concentrated in Attention. However, it slightly weakens original abilities due to all layers structural disruption (see Section 4.3). Thus, the choice of fine-tuning parameters and update strategies plays critical roles in domain transfer and knowledge retention.

In this paper, we propose PIECE, a Parameter Importance Estimation-based Continual Enhancement method for learning domain-specific knowledge while preserving general capabilities. **Without historical data or additional model parameters**, PIECE estimates parameter importance before fine-tuning and updates only the top 0.1% of parameters. We introduce two importance estimation strategies: PIECE-F, based on Fisher information to measure sensitivity to the current-task loss, and PIECE-S, a theory-grounded second-order method combining gradient and curvature information. Notably, PIECE-S is our first theory-grounded variant and better preserves original model capabilities than PIECE-F. By freezing the remaining 99.9% of parameters, PIECE enables efficient domain adaptation with minimal degradation of general capabilities.

Extensive experiments [2] on three LLMs and two MLLMs (2B–14B parameters) show that PIECE preserves prior knowledge across diverse downstream tasks while achieving state-of-the-art continual learning performance. Further analyses show that PIECE concentrates updates in selected Attention and LayerNorm components while maintaining low parameter overlap across tasks, supporting its capability preservation and scalability. Our contributions can be summarized as follows:

- We find that parameter selection is an effective strategy to mitigate catastrophic forgetting, enabling continual learning without historical data or additional parameters.
- We propose two parameter importance estimations: PIECE-F and PIECE-S. Notably, PIECE-S is our first theory-grounded and better preserving original model capabilities.
- PIECE is a scalable, model-agnostic method for sustainable continual learning that preserves pre-trained abilities across diverse LLMs and MLLMs.

2 Related Work

Regularization-based methods preserve task-relevant parameters (Kirkpatrick et al., 2017; Aljundi et al., 2018) or constrain intermediate representations or outputs (Li & Hoiem, 2017) to mitigate catastrophic forgetting. **Replay-based methods** retain and replay real samples (Rolnick et al., 2019; Zhuo et al., 2023) or generated examples (Shin et al., 2017; Maekawa et al., 2023) from previous tasks. Although effective, both rely on pretraining data that ordinary foundation-model users often cannot access, limiting their practical use. **Architecture-based methods** expand or isolate model components to reduce cross-task interference. Inspired by Aljundi et al. (2017), methods (Le et al.,

[2] Code and data are available at `https://anonymous.4open.science/r/PIECE-5567`

2024; Chen et al., 2024) use gates to allocate task-specific modules. Tong et al. (2025) optimizes gate training to improve classification accuracy, while Zhao et al. (2024b) improves gate design to use task-specific modules more efficiently and promote knowledge transfer. However, such methods often incur growing parameter and storage overhead as tasks accumulate. **Parameter-Efficient Tuning** (PET) (Zhang et al., 2025a; Ding et al., 2023) has emerged as a practical paradigm for foundation-model post-training. Recent studies show that PET can effectively mitigate catastrophic forgetting (Biderman et al., 2024; Bhat et al., 2025), with LoRA (Hu et al., 2022) showing particularly strong retention (Zhao et al., 2024b). Wang et al. (2023a) proposed O-LoRA, which constrains LoRA updates to orthogonal subspaces for continual learning. Compared to these structured methods, unstructured parameter-efficient tuning focuses more on the model's transfer performance on downstream tasks. Sung et al. (2021) and Xu et al. (2021) showed that tuning task-relevant parameter subsets can outperform full fine-tuning. Du et al. (2024) introduced MIGU, which updates parameters associated with strong output activations. However, these parameters are not necessarily the most beneficial for learning the new task (Molchanov et al., 2017; Zhang et al., 2024). Building on this, we develop two principled importance estimators to identify task-relevant parameters for sparse continual adaptation.

3 METHOD

In this section, we present PIECE, including its formulation, two parameter-importance estimators (PIECE-F and PIECE-S), and mask-based implementation. As shown in Figure 2, PIECE first estimates parameter importance from current-task data to construct a fixed gradient mask. The mask freezes most parameters and restricts updates to a small task-relevant subset.

3.1 PRELIMINARIES

Continual learning aims to enable a model to learn new tasks sequentially while minimizing performance degradation on previously learned ones. Formally, the model learns a sequence of tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_T\}$. Each task T_t corresponds to a dataset $\mathcal{D}_t = \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^{n_t}$, containing n_t input–output pairs. Given a neural network $f(\cdot;\theta)$ parameterized by θ, the objective at each task step t is to adapt to task T_t by minimizing:

$$\theta_t^* = \arg\min_{\theta} \frac{1}{n_t} \sum_{i=1}^{n_t} L(f(x_i^{(t)};\theta), y_i^{(t)}). \tag{1}$$

where $L(\cdot,\cdot)$ denotes the cross-entropy loss.

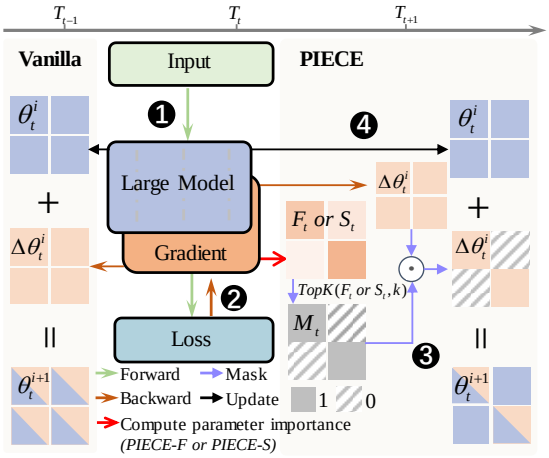


Figure 2: Illustration of PIECE. During parameter updates, PIECE performs standard ① forward and ② backward steps, but before ④ updating, it ③ applies a fixed gradient mask (computed from PIECE-F/S parameter importance) to protect most parameters and update only top-k task-relevant ones.

Typically, only the current dataset $\mathcal{D}_t$ is accessible during the training of task T_t. However, previous approaches often relax this constraint to varying degrees. For example, replay-based methods retain a small subset of past samples for replay or generate pseudo-samples; regularization-based methods store historical signals such as gradients or parameter importance to constrain updates; architecture-based methods adapt to new tasks by adding parameters or modifying the structure, sometimes with additional task-identification memory. In this work, we strictly follow a **no-history assumption**: while learning task T_t, the model is prohibited from accessing any past data or intermediate training information, and both the parameter budget and architecture remain fixed.

Unlike conventional settings that assume continual learning begins from a task-specific initial model, foundation models may already possess substantial task capabilities before adaptation. We therefore treat such pretrained capabilities as prior knowledge to be preserved, allowing continual adaptation to begin from an arbitrary pretrained model state rather than only from a predefined T_1.

3.2 FISHER INFORMATION-BASED IMPORTANCE

We employ Fisher Information (Fisher, 1925) to assess the importance of model parameters, which is widely used in modern machine learning for various purposes, such as protecting critical parameters of previous tasks (Kirkpatrick et al., 2017), guiding model compression and pruning (Singh & Alistarh, 2020), and selecting target parameters for efficient fine-tuning (Sung et al., 2021; Xu et al., 2021). From the perspective of parameter importance, a parameter that strongly influences the model's output will induce a significant change in the predictive distribution when updated. Let θ denote the model parameters and x the input, with the output distribution denoted by $p_\theta(y|x)$. When a small perturbation $||\delta|| \to 0$ is applied to the parameters (as in the fine-tuning stage), it can be shown (Martens, 2020; Pascanu & Bengio, 2013) that the resulting change in the predictive distribution can be approximated as:

$$\mathbb{E}_x[D_{\mathrm{KL}}(p_\theta(y|x)\|p_{\theta+\delta}(y|x))] \approx \frac{1}{2}\delta^\top F_\theta \delta, \tag{2}$$

where F_θ denotes the Fisher information matrix, which is defined as:

$$F_\theta = \mathbb{E}_x\left[\mathbb{E}_{y\sim p_\theta(y|x)}\left[\nabla_\theta \log p_\theta(y|x) \nabla_\theta \log p_\theta(y|x)^\top\right]\right]. \tag{3}$$

Since the true data distribution is typically unavailable and computing the full Fisher information matrix is expensive, we follow the approaches of Kirkpatrick et al. (2017), Sung et al. (2021), and Xu et al. (2021) to approximate the importance of each parameter using the diagonal elements of the empirical Fisher information matrix computed over the training set $\mathcal{D}_t$ of a given task. Specifically, for task T_t, the Fisher Information parameter importance of the i-th parameter is defined as:

$$F_{t,i} \approx \frac{1}{|D_t|}\sum_{(x,y)\in D_t}\left(\frac{\partial \log p_{\theta_{t-1}}(y|x)}{\partial \theta_{t-1,i}}\right)^2. \tag{4}$$

3.3 SECOND-ORDER NORMALIZATION IMPORTANCE

From a Bayesian perspective, we first revisit the rationale for using Fisher information to measure parameter importance for previous tasks, taking the optimization of task T_2 as an example:

$$p(\theta_2 \mid D_1, D_2) = \frac{p(D_2 \mid \theta_2)p(\theta_2 \mid D_1)}{p(D_2)}. \tag{5}$$

Here, the posterior distribution $p(\theta_2 \mid D_1)$ from the previous task T_1 serves as the prior when optimizing the new task, thereby preserving knowledge acquired from the previous task. Maximizing this posterior is equivalent to minimizing the new task loss $L_2(\theta_2)$ augmented with a regularization term induced by the prior from the old task:

$$\theta_2 = \arg\max_{\theta_2} \log p(\theta_2 \mid D_1, D_2) = \arg\min_{\theta_2} L_2(\theta_2) - \log p(\theta_2 \mid D_1). \tag{6}$$

We then perform a second-order Taylor expansion of $f(\theta_2) = \log p(\theta_2 \mid D_1)$ around the previous-task optimum θ_1:

$$f(\theta_2) \approx f(\theta_1) + J_{\theta_2,\theta_1}(\theta_2 - \theta_1) + \frac{1}{2}(\theta_2 - \theta_1)^\top H_{\theta_2,\theta_1}(\theta_2 - \theta_1), \tag{7}$$

$$J_{\theta_2,\theta_1} = \left.\frac{\partial f(\theta_2)}{\partial \theta_2}\right|_{\theta_2=\theta_1}, H_{\theta_2,\theta_1} = \left.\frac{\partial^2 f(\theta_2)}{\partial \theta_2^2}\right|_{\theta_2=\theta_1}.$$

Since θ_1 is a stationary point for the previous task, the Jacobian vanishes, i.e., $J_{\theta_2,\theta_1} = 0$. Moreover, under standard regularity conditions, the negative expected Hessian of the log-likelihood corresponds to the Fisher information matrix, which is commonly used to approximate the local posterior curvature. Thus, using Fisher information to quantify parameter importance is consistent with a Bayesian Laplace approximation.

Although this Fisher-based interpretation characterizes parameter importance around the optimum of the previous task, it relies on the assumption that the current parameters remain close to that optimum, where the gradients vanish. However, when learning a new task T_t, the initialization θ_{t-1}

is in general not a stationary point under the new data distribution, i.e., $\nabla_{\theta_{t-1}} \log p_{\theta_{t-1}}(\mathcal{D}_t) \neq 0$. Neglecting the first-order term thus ignores the primary source of parameter updates for the new task. To address this, we instead expand the new-task log-likelihood $f_t^*(\theta) = \log p(\mathcal{D}_t \mid \theta)$ around the current parameters θ_{t-1}:

$$f_t^*(\theta_t) \approx f_t^*(\theta_{t-1}) + J_{\theta_t,\theta_{t-1}}(\theta_t - \theta_{t-1}) + \frac{1}{2}(\theta_t - \theta_{t-1})^\top H_{\theta_t,\theta_{t-1}}(\theta_t - \theta_{t-1}), \tag{8}$$

and yield the approximate optimal update:

$$\theta_t^\star \approx \theta_{t-1} - H^{-1}_{\theta_t,\theta_{t-1}} J_{\theta_t,\theta_{t-1}}. \tag{9}$$

This expression indicates that the magnitude of parameter updates is jointly determined by the first-order gradient and the second-order curvature: parameters with large gradients and low curvature tend to change more during learning and thus contribute more to adaptation. Further, assuming that the parameters are near a local optimum, the Laplace approximation gives $p(\theta \mid \mathcal{D}_t) \approx \mathcal{N}\big(\theta_t^\star, (-H_{\theta_t^\star,\theta_{t-1}})^{-1}\big)$, while the current parameters θ_{t-1} can be viewed as a degenerate distribution $\delta(\theta_{t-1})$. We therefore use the standardized mean shift between these two distributions as the importance measure for each parameter. To avoid computing true data expectations and Hessians, we approximate the Jacobian by the empirical gradients and the negative Hessian by the empirical Fisher information computed at θ_{t-1} under task T_t. Specifically, for task T_t the Second-order Normalized parameter importance of the i-th parameter is defined as:

$$S_{t,i} = \frac{\left| \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \frac{\partial \log p_{\theta_{t-1}}(y|x)}{\partial \theta_{t-1,i}} \right|}{\sqrt{\frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \left(\frac{\partial \log p_{\theta_{t-1}}(y|x)}{\partial \theta_{t-1,i}} \right)^2 + \xi}}, \tag{10}$$

where $\xi > 0$ is a constant for numerical stability.

3.4 PIECE

We introduce the core workflow of PIECE using standard gradient-based fine-tuning as an example. Let θ_t^s denote the model parameters at the s-th iteration on task T_t, with $\theta_t^0 = \theta_{t-1}$ denoting the initialization for task T_t. In standard fine-tuning, gradients of the loss $L(f(x^{(t)}; \theta_t^s), y^{(t)})$ are computed and applied to update all parameters via gradient descent:

$$\theta_t^{s+1} = \theta_t^s - \eta \frac{\partial L(f(x^{(t)}; \theta_t^s), y^{(t)})}{\partial \theta_t^s}. \tag{11}$$

In contrast, PIECE first computes the importance of each parameter from the current task data (as defined in Eq. 4 or Eq. 10), and selects the top-k most important parameters, forming a task-specific set of selected parameter indices $\mathcal{C}_t^k$. In practice, we set $k = \lfloor 0.001 \cdot |\theta| \rfloor$, i.e., only 0.1% of parameters are trainable for each task. We then define a binary mask with the same shape as θ:

$$M_{t,i} = \begin{cases} 1, & \theta_{t,i} \in \mathcal{C}_t^k \\ 0, & \theta_{t,i} \notin \mathcal{C}_t^k \end{cases}. \tag{12}$$

The resulting mask M_t is fixed throughout the optimization of task T_t and applied to the gradients, freezing unselected parameters while updating the selected ones:

$$\theta_t^{s+1} = \theta_t^s - \eta \left(\frac{\partial L(f(x^{(t)}; \theta_t^s), y^{(t)})}{\partial \theta_t^s} \odot M_t \right). \tag{13}$$

Unlike methods such as EWC that rely on parameter statistics retained from previous tasks, PIECE requires no historical data or training statistics. By restricting updates to a small parameter subset, PIECE limits perturbations to the pretrained model, thereby reducing interference with its original capabilities. Consequently, this design shifts the core challenge to precisely localizing task-relevant parameters for downstream tasks under a severely restricted update budget. This motivates us to develop more principled estimators for identifying task-relevant parameters under a highly constrained update budget. The subsequent sections will present extensive experiments to comprehensively validate the advantages of this mechanism design and the refined importance estimation strategy in continual learning.

4 EXPERIMENTS

4.1 EXPERIMENT SETUP

Datasets. For language tasks, we use the TRACE (Wang et al., 2023b) benchmark with 1,000 training samples per task. TRACE contains eight tasks spanning science (Sin.), finance (Fom.), meeting summarization (Mee.), multilingual tasks (German [20M.] and Chinese [Cst.]), code completion (Py150), and mathematical reasoning (Ngc, Ngd). We use the HumanEval (Chen et al., 2021) programming benchmark to assess the model's capability retention. For multi-modal tasks, we use VQA v2 (Goyal et al., 2017). Following VQACL (Zhang et al., 2023), tasks are split by question type, with 1,000 training samples randomly selected for each task while retaining the original test sets. We consider three question types: action (Act.), commonsense (Com.), and count (Cou.). Flickr30K (Young et al., 2014) is used to evaluate overall multimodal capability retention. HumanEval and Flickr30K are used only for evaluation and are not included in continual training. See Appendix A for dataset details.

Baselines. We compare PIECE with three categories of continual learning baselines. **Regularization** methods include EWC (Kirkpatrick et al., 2017), GEM (Lopez-Paz & Ranzato, 2017), and LwF (Li & Hoiem, 2017), which constrain updates via parameter importance, gradient projection, or distillation to preserve prior knowledge. **Replay** methods reuse past data to reduce forgetting, either after each new task (Replay) (Rebuffi et al., 2017) or continuously during training (Replay-online) (Rolnick et al., 2019). **Parameter-Efficient Tuning (PET)** baselines include SeqLoRA (Hu et al., 2022), O-LoRA (Wang et al., 2023a), LayerNorm (Zhao et al., 2024a), and MIGU (Du et al., 2024), which selectively updates original model parameters according to the L_1-norm of their output responses. For fair comparison, MIGU uses the same 0.1% trainable-parameter budget as PIECE. Full fine-tuning (SeqFT) is also included. See Appendix B for details.

Metrics. Let $R_{t,i}$ denote the model's performance on task i after learning task t. Accuracy is used for classification and single-token QA tasks, while the average of ROUGE-L and BLEU scores is used for other generation tasks. The sequential learning metrics include Overall Performance (OP) (Chaudhry et al., 2018), $OP_t = \frac{1}{t}\sum_{i=1}^{t} R_{t,i}$, which measures the average performance across all learned tasks, and Backward Transfer (BWT) (Lopez-Paz & Ranzato, 2017), $BWT_t = \frac{1}{t-1}\sum_{i=1}^{t-1}(R_{t,i} - R_{i,i})$, which measures the effect of subsequent learning on previous tasks. For capability preservation, code generation on HumanEval is evaluated using Pass@1, while image captioning on Flickr30K is evaluated using the mean of ROUGE-L and BLEU scores. See Appendix C for details.

Models and Implementation Details. We use Gemma2-2B (Team et al., 2024), Llama3-8B (Grattafiori et al., 2024), and Qwen3-14B (Team, 2025) for language tasks, and Qwen3-VL-4B (Bai et al., 2025) and LLaVA-1.5-7B (Liu et al., 2023; 2024) for multimodal tasks. Since continual learning performance can be sensitive to task order, we evaluate all methods under three language-task sequences, following common practice in continual learning studies (He et al., 2024). For multimodal tasks, we use a fixed order of action, commonsense, and count. All models use the Adam optimizer with batch size 64 and random seed 42. For Gemma2-2B, the learning rate is 5×10^{-4} for SeqLoRA/O-LoRA and 5×10^{-5} for all other methods; for the remaining models, the corresponding learning rates are 1×10^{-4} and 1×10^{-5}. No warmup is applied. Qwen3-14B is trained for 1 epoch per task, while other models are trained for 5 epochs per task. Experiments are conducted on 4×80GB A100, 4×80GB A800, and 8×40GB A100 GPUs. See Appendix B for deployment details.

4.2 MAIN RESULTS

Table 1 and Table 2 report the main results on language and multimodal tasks, respectively, from which we draw the following conclusions:

PIECE demonstrates outstanding performance across both language and multimodal tasks. Across models and task orders, PIECE consistently achieves competitive or best OP, with at least one variant attaining the highest OP in most settings. PIECE also remains generally competitive in BWT, suggesting a reasonable balance between new-task adaptation and retention of previously learned tasks.

PIECE effectively preserves the original capabilities of foundation models. On HumanEval

Table 1: Performance of methods on language tasks. OP and BWT are computed on sequential tasks; inherent capability variation is measured via HumanEval. *Qwen3-14B trained one epoch per task, showing less forgetting.

Methods	Order 1			Order 2			Order 3		
	HumanEval	OP	BWT	HumanEval	OP	BWT	HumanEval	OP	BWT
Gemma2-2B									
SeqFT	0.0	15.13	-19.08	0.0	13.98	-15.47	0.0	9.19	-21.74
EWC	0.0	17.13	-13.99	0.0	16.93	-10.65	0.0	9.88	-25.38
GEM	0.0	19.15	-16.27	0.0	15.53	-13.32	0.0	8.47	-23.10
LwF	0.0	16.86	-10.10	0.0	15.40	-4.80	0.0	5.93	-14.48
replay	0.0	26.14	-1.43	0.0	27.60	**0.84**	0.0	26.94	**-0.61**
replay-online	0.0	29.10	-3.82	0.0	29.50	-2.18	0.0	28.38	-4.15
SeqLoRA	10.37	39.77	-10.80	7.93	31.88	-7.69	9.15	29.95	-9.17
O-LoRA	13.41	43.15	-3.45	12.20	36.56	-6.13	0.61	30.25	-11.39
LayerNorm	21.95	27.23	-1.67	19.51	23.38	-0.64	20.73	23.04	-0.76
MIGU	1.22	22.71	-10.36	0.0	26.18	-5.25	0.0	21.65	-14.11
PIECE-F (ours)	32.32	**44.33**	-3.21	19.51	28.01	-5.15	**31.71**	41.39	-7.14
PIECE-S (ours)	**33.54**	42.03	**-1.18**	**31.10**	**44.60**	-3.97	26.22	**42.80**	-4.26
Llama3-8B									
SeqFT	0.0	39.22	-10.69	0.61	29.78	-10.25	0.0	20.10	-28.79
EWC	0.0	32.95	-10.05	0.0	34.33	-11.27	2.44	36.92	-6.74
GEM	0.0	39.37	-10.24	0.0	32.03	-12.44	0.0	39.07	-7.15
LwF	0.0	33.06	-9.94	1.83	27.20	-8.30	1.22	26.69	-13.53
replay	0.0	38.62	-2.94	0.61	41.90	**-0.24**	0.61	39.56	-3.87
replay-online	0.0	45.47	-1.99	0.0	41.33	-1.48	0.0	40.90	-1.64
SeqLoRA	28.05	46.04	-4.39	27.44	44.36	-7.62	30.49	32.14	-2.80
O-LoRA	26.83	39.73	-7.19	26.22	41.39	-4.07	29.27	43.43	-2.43
LayerNorm	27.44	30.39	-1.86	34.76	31.01	-0.98	31.71	31.36	**-0.32**
MIGU	0.0	37.65	-10.02	4.88	43.76	-6.21	1.22	44.45	-6.26
PIECE-F (ours)	29.88	**50.05**	-2.66	31.71	38.54	-3.35	30.49	36.05	-3.11
PIECE-S (ours)	**32.32**	49.05	**-1.78**	**35.37**	**45.55**	-1.37	**34.76**	**44.79**	-2.42
Qwen3-14B *									
SeqFT	57.93	47.50	-6.42	**62.80**	43.31	-9.55	58.54	43.97	-10.67
EWC	58.54	47.22	-6.67	60.98	40.55	-12.45	54.88	47.12	-6.51
GEM	**61.59**	42.73	-8.84	49.39	43.35	-5.49	57.32	42.46	-5.79
LwF	58.54	39.57	-6.62	53.66	37.21	-4.19	54.88	34.46	0.59
replay	51.83	47.82	-2.40	**62.80**	49.96	-1.08	53.05	49.37	**1.29**
replay-online	57.32	49.09	-2.60	62.20	49.48	-0.25	**63.41**	49.30	-2.50
SeqLoRA	57.32	44.94	**4.74**	47.56	36.71	**3.71**	40.85	33.40	-3.77
O-LoRA	44.51	43.23	-2.20	50.00	39.73	1.78	46.95	31.48	-2.98
LayerNorm	47.56	22.16	-0.17	50.00	22.35	0.02	46.95	22.16	-0.19
MIGU	42.68	**50.95**	-0.90	30.49	43.62	-1.42	45.73	46.53	-7.60
PIECE-F (ours)	50.61	49.80	-0.33	51.83	49.66	-5.37	50.00	51.77	-1.04
PIECE-S (ours)	59.15	49.35	-0.16	61.59	**50.11**	0.69	56.10	**53.15**	-1.61

for language models and Flickr30K for multimodal models, PIECE consistently achieves strong capability retention and generally outperforms competing approaches. For Qwen3-14B, which is trained for only one epoch per task, PIECE-S remains close to the best HumanEval performance under Orders 1 and 2.

PIECE-S offers a stronger overall retention-adaptation profile. Across the nine language and two multimodal settings, PIECE-S achieves higher BWT than PIECE-F in ten, with Qwen3-14B under Order 3 as the sole exception. It also achieves the best BWT among all methods for both multimodal models. For OP, PIECE-S leads PIECE-F in eight settings, whereas PIECE-F leads for all three language models under Order 1. These results favor PIECE-S across most evaluated settings while identifying a consistent, order-specific advantage for PIECE-F.

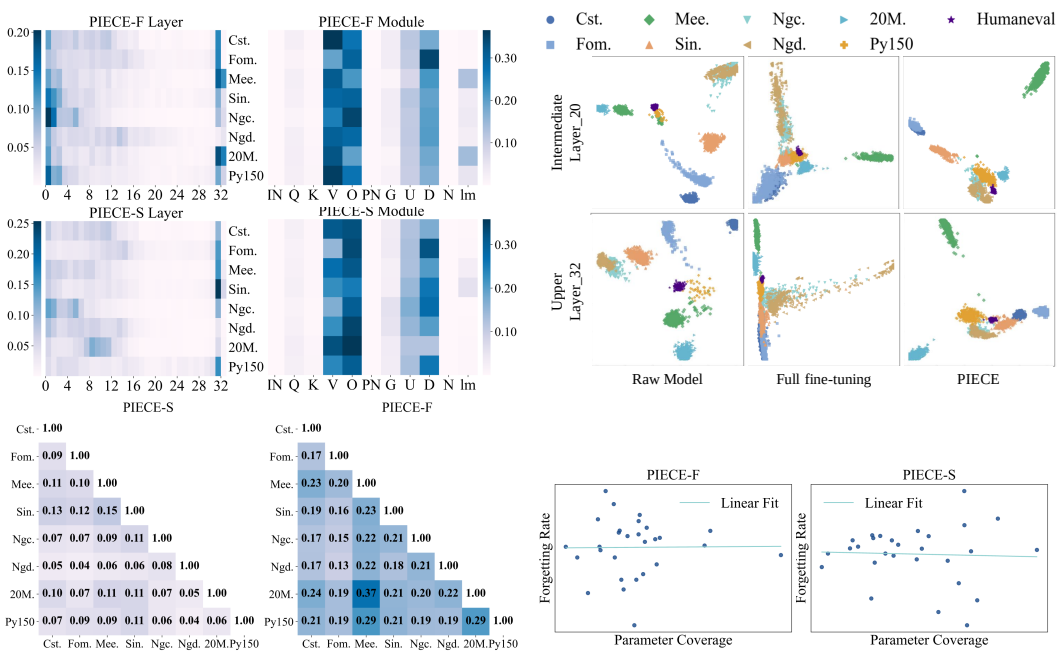


Figure 3: Top-left: Distribution of critical parameters identified by **PIECE-F** and **PIECE-S** across different tasks. Top-right: Visualization of intermediate and upper layer task representations comparing the base model, full fine-tuning, and **PIECE**. Bottom-left: Parameter overlap across tasks. Bottom-right: Correlation of parameter overlap and forgetting rate.

4.3 Analysis

Beyond overall performance, we analyze **PIECE** through three key questions on **Llama3-8B** under **Order 1**.

1. Where does PIECE learn, and why is it effective?

To explore where **PIECE** learns and why it is stable, we visualize the important parameters selected by **PIECE-F** and **PIECE-S** across tasks. Figure 3 Top-left shows both methods mainly target attention submodules `V`/`O` and feed-forward layers (`U`/`D`). This aligns with Yao et al. (2024), which suggests that the `V`/`O` modules play a more crucial role than `Q`/`K` in downstream adaptation, and Dai et al. (2022), which identifies `U` and `D` as key locations for knowledge storage. At the layer level, both methods primarily attend to the higher layers, consistent with Zhao et al. (2024c) linking them to downstream task relevance. The distinction is that **PIECE-F** also emphasizes lower layers, potentially aiding better adaptation to downstream tasks, although adjusting these shared modules may introduce some forgetting. Notably, both mechanisms largely avoid the intermediate layers, which may indicate that preserving these layers is beneficial for maintaining pretrained representations. To further examine the role of intermediate layers in the model's multitask capabilities, we visualized task representations at intermediate and upper layers using dimensionality reduction. As shown in Figure 3 Top-right, compared with

Table 2: Results on multimodal tasks.

Methods	Flickr30k(↑)	VQA			OP(↑)	BWT(↑)
		Act.	Com.	Cou.		
Qwen3-VL-4B	*26.19*					
SeqFT	7.42	47.29	48.66	44.03	46.66	-13.86
EWC	4.34	40.06	40.42	38.40	39.62	-18.14
GEM	8.28	47.64	42.54	41.34	43.84	-12.89
LwF	26.67	**66.97**	58.19	40.89	55.35	-5.74
replay	20.79	61.20	60.88	**52.27**	**58.11**	-2.72
replay-online	8.86	65.65	**69.68**	50.67	62.00	-0.83
SeqLoRA	32.18	38.66	45.56	23.38	35.87	-1.62
O-LoRA	5.23	40.06	48.90	37.01	41.99	-1.25
LayerNorm	23.77	60.29	59.09	43.06	54.15	-1.37
MIGU	27.44	47.64	57.46	37.29	47.46	-4.87
PIECE-F (ours)	35.59	62.66	58.76	48.19	56.54	-1.53
PIECE-S (ours)	**38.74**	62.15	61.53	47.50	57.06	**-0.50**
LLaVA-1.5-7B	*30.82*					
SeqFT	10.69	41.72	39.36	43.03	41.37	-26.84
EWC	21.92	46.38	52.40	24.21	41.00	-10.04
GEM	24.17	49.65	53.63	30.16	44.48	-7.18
LwF	36.02	67.66	58.11	2.07	42.62	-0.51
replay	13.61	65.79	70.90	45.62	60.77	-0.78
replay-online	10.46	67.25	69.11	44.93	60.43	-1.66
SeqLoRA	31.04	**75.31**	75.63	53.10	68.01	-0.51
O-LoRA	6.80	70.38	72.94	52.30	65.21	-0.26
LayerNorm	30.71	69.68	60.39	2.91	44.33	-0.20
MIGU	21.81	67.66	52.40	42.89	54.32	-8.62
PIECE-F (ours)	37.51	74.48	77.18	53.06	68.24	-0.29
PIECE-S (ours)	**41.94**	**75.31**	**77.34**	**53.27**	**68.64**	**-0.16**

the base model, full fine-tuning causes larger changes in intermediate-layer representations and is associated with poorer task separability, whereas **PIECE** preserves clearer task distinctions.

2. Does PIECE forget more as the number of tasks increases?

The core of **PIECE** lies in fine-tuning only a small subset of critical parameters, raising a natural question about its scalability: as the number of tasks increases, could important parameters increasingly overlap across tasks and thereby exacerbate forgetting? We investigate this question from both parameter-level and extended-sequence perspectives. As shown in Figure 3 Bottom-left, the parameter overlap across tasks remains relatively low for both **PIECE-F** and **PIECE-S**, with **PIECE-S**, which places greater emphasis on current-task relevance, exhibiting the lowest overlap. Moreover, the correlation between parameter overlap and task forgetting is negligible (Figure 3 Bottom-right; Pearson $r = -0.04$ for **PIECE-S** and $r = 0.01$ for **PIECE-F**), suggesting that increased parameter reuse across tasks does not systematically translate into greater forgetting.

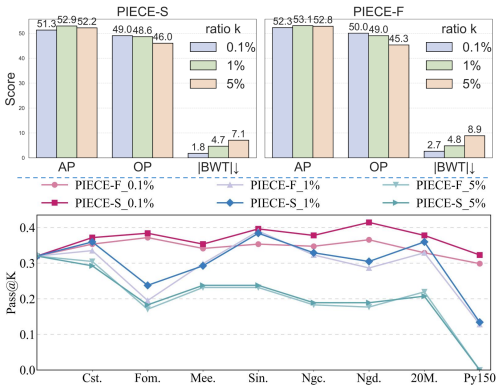


Figure 4: Effect of the number of updated parameters on performance and forgetting. Average Performance (AP) is the mean performance across sequential tasks after each task, reflecting transfer ability: $AP_t = \frac{1}{t}\sum_{i=1}^{t} R_{i,i}$.

Table 3: Results on Super-NI.

Methods	HumanEval(↑)	OP(↑)	BWT(↑)
SeqLoRA	26.22	44.42	-15.69
O-LoRA	0.0	50.74	-15.64
PIECE-F (ours)	<u>26.83</u>	<u>53.55</u>	<u>-14.64</u>
PIECE-S (ours)	**28.66**	**55.06**	**-11.76**

To further verify whether this property translates into empirical scalability over longer task sequences, we conduct an extended experiment with 25 tasks from Super-NI (Wang et al., 2022). Detailed task lists and ordering are provided in Appendix E. As shown in Table 3, although the longer sequence introduces stronger interference with the model's original capabilities, **PIECE** consistently achieves the best performance across all three core metrics: original capability retention, OP, and BWT. Together, the low cross-task parameter overlap, its negligible association with forgetting, and the strong performance over 25 sequential tasks demonstrate that **PIECE** remains effective as the task sequence grows, supporting its scalability to longer continual-learning scenarios.

3. How does the sparsity ratio affect performance?

To investigate the effect of the number of updated parameters on performance, we varied the selection ratio k, with the results shown in Figure 4. The experiments indicate that increasing k initially improves the model's transfer performance on downstream tasks, but this improvement is not sustained; at $k = 5\%$, performance is actually lower than at $k = 1\%$. Meanwhile, larger k values substantially exacerbate forgetting, resulting in a decrease in OP and causing more severe degradation of the original capabilities.

5 Conclusion

We present PIECE, a Parameter Importance Estimation-based Continual Enhancement method for foundation models. PIECE mitigates catastrophic forgetting by selectively updating only the most critical 0.1% of parameters, guided by theoretically grounded importance estimation. Without replaying past-task data or modifying the model architecture, PIECE achieves efficient domain adaptation while preserving general reasoning and programming abilities. Extensive experiments on diverse LLMs and MLLMs show that PIECE surpasses existing approaches in both retention and transfer. By coupling parameter-efficient tuning with importance estimation, PIECE offers a scalable, model-agnostic pathway toward sustainable, domain-adaptive foundation models. In the future, we will further distinguish fine-tuning key parameters from stability-critical ones to better balance downstream transfer and forgetting mitigation.

AI USE STATEMENT

The generative AI model GPT-5.6 was employed exclusively for the linguistic refinement of author-prepared text, including grammar correction and improvements in clarity and readability. It was not involved in developing or assessing the ideas, methodology, equations, code, data, results, figures, references, or conclusions of this study. The authors reviewed all suggested revisions and remain fully responsible for the final content.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3366–3375, 2017.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Claas Beger and Saikat Dutta. Coconut: Structural code understanding does not fall out of a tree. In *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, pp. 128–136. IEEE, 2025.

Prashant Shivaram Bhat, Shakib Yazdani, Elahe Arani, and Bahram Zonooz. Parameter efficient continual learning with dynamic low-rank adaptation. *arXiv preprint arXiv:2505.11998*, 2025.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and forgets less. *Transactions on Machine Learning Research*, 2024.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018.

Jie Chen, Zhipeng Chen, Jiapeng Wang, Kun Zhou, Yutao Zhu, Jinhao Jiang, Yingqian Min, Wayne Xin Zhao, Zhicheng Dou, Jiaxin Mao, et al. Towards effective and efficient continual pre-training of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5779–5795, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Qian Chen, Lei Zhu, Hangzhou He, Xinliang Zhang, Shuang Zeng, Qiushi Ren, and Yanye Lu. Low-rank mixture-of-experts for continual medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 382–392. Springer, 2024.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235, 2023.

Wenyu Du, Shuang Cheng, Tongxu Luo, Zihan Qiu, Zeyu Huang, Ka Chun Cheung, Reynold Cheng, and Jie Fu. Unlocking continual learning abilities in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6503–6522, 2024.

Yujie Feng, Xujia Wang, Zexin Lu, Shenghong Fu, Guangyuan Shi, Yongxin Xu, Yasha Wang, Philip S Yu, Xu Chu, and Xiao-Ming Wu. Recurrent knowledge identification and fusion for language model continual learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27396–27413, 2025.

Ronald Aylmer Fisher. Theory of statistical estimation. In *Mathematical proceedings of the Cambridge philosophical society*, volume 22, pp. 700–725. Cambridge University Press, 1925.

Annette Rios Gonzales, Nicolas Spring, Tannon Kew, Marek Kostrzewa, Andreas Säuberli, Mathias Müller, and Sarah Ebling. A new dataset and efficient baselines for document-level text simplification in german. In *Proceedings of the Third Workshop on New Frontiers in Summarization*, pp. 152–161, 2021.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. Seekr: Selective attention-guided knowledge retention for continual learning of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3254–3266, 2024.

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. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yebowen Hu, Timothy Ganter, Hanieh Deilamsalehy, Franck Dernoncourt, Hassan Foroosh, and Fei Liu. Meetingbank: A benchmark dataset for meeting summarization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16409–16423, 2023.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Minh Le, An Nguyen, Huy Nguyen, Trang Nguyen, Trang Pham, Linh Van Ngo, and Nhat Ho. Mixture of experts meets prompt-based continual learning. *Advances in Neural Information Processing Systems*, 37:119025–119062, 2024.

Jia Li, Ge Li, Yongmin Li, and Zhi Jin. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–23, 2025a.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Zongxia Li, Xiyang Wu, Hongyang Du, Fuxiao Liu, Huy Nghiem, and Guangyao Shi. A survey of state of the art large vision language models: Benchmark evaluations and challenges. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1587–1606, 2025b.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 26296–26306, 2024.

Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. Learning to adapt to evolving domains. *Advances in neural information processing systems*, 33:22338–22348, 2020.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*, 2021.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231.

Aru Maekawa, Hidetaka Kamigaito, Kotaro Funakoshi, and Manabu Okumura. Generative replay inspired by hippocampal memory indexing for continual language learning. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 930–942, 2023.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3505–3523. Association for Computational Linguistics, 2022.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJGCiw5gl.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Agam Shah, Suvan Paturi, and Sudheer Chava. Trillion dollar words: A new financial dataset, task & market analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6664–6679, 2023.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Kai Tong, Kang Pan, Xiao Zhang, Erli Meng, Run He, Yawen Cui, Nuoyan Guo, and Huiping Zhuang. Any-ssr: How recursive least squares works in continual learning of large language model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3047–3057, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vinay Kumar Verma, Kevin J Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient feature transformations for discriminative and generative continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13865–13875, 2021.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023a.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, et al. Trace: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023b.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. 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, 2022.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9514–9528, 2021.

Xinhao Yao, Hongjin Qian, Xiaolin Hu, Gengze Xu, Wei Liu, Jian Luan, Bin Wang, and Yong Liu. Theoretical insights into fine-tuning attention mechanism: Generalization and optimization. *arXiv preprint arXiv:2410.02247*, 2024.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

Dan Zhang, Tao Feng, Lilong Xue, Yuandong Wang, Yuxiao Dong, and Jie Tang. Parameter-efficient fine-tuning for foundation models. *arXiv preprint arXiv:2501.13787*, 2025a.

Xi Zhang, Feifei Zhang, and Changsheng Xu. Vqacl: A novel visual question answering continual learning setting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19102–19112, 2023.

Xinyu Zhang, Yuanquan Hu, Fangchao Liu, and Zhicheng Dou. P3: Prompts promote prompting. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11948–11965, 2025b.

Zhi Zhang, Qizhe Zhang, Zijun Gao, Renrui Zhang, Ekaterina Shutova, Shiji Zhou, and Shanghang Zhang. Gradient-based parameter selection for efficient fine-tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28566–28577. IEEE, 2024.

Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. Tuning layernorm in attention: Towards efficient multi-modal llm finetuning. In *ICLR*, 2024a.

Chenye Zhao, Yingjie Li, and Cornelia Caragea. C-stance: A large dataset for chinese zero-shot stance detection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13369–13385, 2023.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11641–11661, 2024b.

Zheng Zhao, Yftah Ziser, and Shay B Cohen. Layer by layer: Uncovering where multi-task learning happens in instruction-tuned large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15195–15214, 2024c.

Tao Zhuo, Zhiyong Cheng, Zan Gao, Hehe Fan, and Mohan Kankanhalli. Continual learning with strong experience replay. *arXiv preprint arXiv:2305.13622*, 2023.

APPENDIX

A DATASETS

A.1 LANGUAGE TASK DETAILS

The language tasks in this paper are derived from the TRACE dataset, which was constructed following three principles:

1. the datasets should be sufficiently novel so that large language models (LLMs) exhibit weak initial performance;
2. the tasks should be challenging enough to comprehensively test reasoning and generalization capabilities;
3. the tasks should cover a wide range of domains and types.

Due to its well-designed and diverse task settings, TRACE has been adopted in several recent continual learning studies (Tong et al., 2025; He et al., 2024; Feng et al., 2025; Du et al., 2024). The language tasks in TRACE can be categorized into four groups: **Domain-Specific Tasks**, **Multi-lingual Tasks**, **Code QA Tasks**, and **Mathematical Reasoning Tasks**. The statistics for each category are shown in Table 4, and configurations are introduced below.

Domain-Specific Tasks

Domain-specific tasks evaluate a model's knowledge acquisition and application within specialized domains. When relevant domain information is underrepresented in training corpora, model performance can degrade significantly. TRACE includes the following three datasets under this category:

- **ScienceQA (Sin.)** (Lu et al., 2022): Collected from elementary and high school science curricula, covering topics across natural science, social science, and language science. This dataset requires multi-hop reasoning and scientific knowledge understanding. Only text-only samples are retained in this work.
- **FOMC (Fom.)** (Shah et al., 2023): A financial-domain classification task aimed at identifying whether Federal Reserve policy statements are "hawkish" or "dovish." The original corpus consists of three parts—meeting minutes, press conference transcripts, and speeches. A combined version of all three is used in this work.
- **MeetingBank (Mee.)** (Hu et al., 2023): A newly introduced dataset for city council meeting summarization, requiring global comprehension and compression of lengthy meeting transcripts. As a relatively unexplored domain, it effectively evaluates models' long-context understanding and abstraction abilities.

Multi-lingual Tasks

Cross-lingual capability serves as an important measure of a large language model's generalization ability. However, due to vocabulary coverage and corpus imbalance, most models perform worse on non-English languages. TRACE includes two datasets for evaluating such multi-lingual abilities:

Dataset	Language Task									Multi-modal Task			
	Cst.	Fom.	Mee.	Sin.	Ngc.	Ngd.	20M.	Py150.	HumanEval	Act.	Com.	Cou.	Flickr30k
Train	1000								-	1000			-
Test	2000	496	692	2000	81	325	200	2000	164	1448	1227	2905	1000

Table 4: Dataset Statistics for Language and Multi-modal Tasks.

- **C-STANCE (Cst.)** (Zhao et al., 2023): A Chinese stance detection dataset collected from Sina Weibo. It includes two subtasks: target-based and domain-based stance detection. We adopt the target-based subtask, where targets in the test set do not appear in the training set, to evaluate zero-shot and cross-domain generalization.
- **20Minuten (20M.)** (Gonzales et al., 2021): A German text simplification dataset derived from the Swiss news magazine 20 Minuten. Each instance consists of an original article and its simplified summary, used to evaluate model performance in German text generation.

Code QA Tasks
Code-related tasks are used to assess a model's understanding and generation capabilities under structured, long-context conditions. TRACE employs the Py150 (Lu et al., 2021) corpus to construct the code QA task. The corpus contains approximately 150,000 Python programs collected from GitHub repositories. The task requires predicting the subsequent content given code context, where label formats differ from those in standard code generation tasks, posing additional challenges to capability retention. Considering that **HumanEval** is used in this paper to measure the retention of code generation ability, this task plays a crucial role in assessing model performance. Thus, we place Py150 last in TRACE testing order to limit its impact on intrinsic capabilities.
Mathematical Reasoning Tasks
Mathematical reasoning tasks are designed to test models' arithmetic and logical reasoning capabilities. TRACE selects the first two subtasks (Ngc., Ngd.) from the **NumGLUE** benchmark (Mishra et al., 2022). Both subtasks require numerical computation and logical inference in natural language settings. These tasks impose higher demands on symbolic reasoning and are particularly useful for evaluating capability retention under abstract reasoning scenarios.

Given that programming ability is widely recognized as a representative prior knowledge that is particularly susceptible to forgetting during post-training (Chen et al., 2025), we adopt it as a measure of the model's retention of original capabilities and quantify it using the **HumanEval** benchmark. HumanEval is a standardized evaluation dataset for Python function generation tasks, designed to assess the code generation capabilities of large language models (Li et al., 2025a; Beger & Dutta, 2025; Zhang et al., 2025b). The dataset contains 164 programming problems, each consisting of a function description (function docstring) and a function signature. The model is required to generate a complete, executable function implementation based on the natural language description. Unlike typical natural language generation tasks, HumanEval evaluation relies on unit tests rather than manual annotation. The dataset is not used for training, and is solely employed to assess the model's retention of original capabilities after multi-task continual learning. Since programming tasks demand a high degree of logical consistency, semantic accuracy, and structured reasoning, and these abilities are particularly vulnerable to degradation during multi-task training, performance changes on HumanEval provide a sensitive and reliable indicator of knowledge retention and forgetting in continual learning scenarios.

A.2 Multi-modal Task Details

The multi-modal tasks in this paper are derived from the VQA v2 dataset (Goyal et al., 2017). To facilitate continual learning experiments, we follow the task grouping scheme of VQACL (Zhang et al., 2023), splitting the tasks by question type and randomly sampling 1,000 training examples per task while retaining the original test sets. Based on performance improvements and capability retention, we focus on the following three question types in this study:

- **Action (Act.)**: Action recognition questions, which require the model to understand the actions of people or objects in an image;
- **Commonsense (Com.)**: Commonsense reasoning questions, which require the model to make judgments based on both visual information and general knowledge;

- **Count (Cou.)**: Counting questions, which require the model to accurately identify the number of target objects in an image.

In addition, we use the Flickr30K dataset (Young et al., 2014) to evaluate the model's overall capability retention in multi-modal tasks. This dataset contains rich image–text pairs, with captions typically longer than those in VQA tasks, making it suitable for assessing the model's comprehensive performance in multi-modal understanding and generation.

B Baselines and Implementation Details

This section provides detailed descriptions of all continual learning baselines compared in this work, including their core ideas and optimization objectives. The baselines fall into three categories: **Regularization-based methods**, **Replay-based methods**, and **Parameter-Efficient Tuning (PET)**.

B.1 Regularization-based Methods

EWC (Elastic Weight Consolidation).
EWC (Kirkpatrick et al., 2017) estimates the importance of model parameters on previous tasks and applies a quadratic penalty to discourage significant updates on important parameters when learning new tasks. The overall objective for task t is :

$$\mathcal{L}_{\text{EWC}} = \mathcal{L}_t(\theta) + \lambda \sum_i F_i(\theta_i - \theta_i^*)^2, \tag{14}$$

where θ^* denotes the parameter values after training the previous task, F_i is the Fisher information representing the importance of parameter i and λ controls the strength of the regularizer (set to 0.5 in our experiments).

GEM (Gradient Episodic Memory).
GEM (Lopez-Paz & Ranzato, 2017) stores a small number of samples from previous tasks in an episodic memory and constrains gradient updates so that the loss on old tasks does not increase. During training on task t, GEM requires:

$$\langle g, g_k \rangle \geq 0, \quad \forall k < t, \tag{15}$$

where g is the gradient for the current task and g_k is the gradient computed on memory samples of task k. If this constraint is violated, i.e., $\langle g, g_k \rangle < 0$, GEM solves the following quadratic program to obtain a projected gradient:

$$\begin{aligned} \min_{\tilde{g}} \quad & \|\tilde{g} - g\|^2 \\ \text{s.t.} \quad & \langle \tilde{g}, g_k \rangle \geq 0, \forall k < t. \end{aligned} \tag{16}$$

The projected gradient $\tilde{g}$ is then used for parameter updates, ensuring no negative backward transfer.

LwF (Learning without Forgetting).
LwF (Li & Hoiem, 2017) employs the predictions of the previous model as soft targets and introduces a distillation loss to constrain changes in the output distribution when learning new tasks:

$$\mathcal{L}_{\text{LwF}} = \mathcal{L}_t^{\text{CE}} + \alpha \cdot \mathcal{L}_{\text{KD}}, \tag{17}$$

where the distillation loss is:

$$\mathcal{L}_{\text{KD}} = -\sum_c p_c^{\text{old}}(T) \log p_c^{\text{new}}(T), \tag{18}$$

with softened probabilities computed using temperature T:

$$p_c(T) = \frac{\exp(z_c/T)}{\sum_j \exp(z_j/T)}. \tag{19}$$

Following common practice, we set $\alpha = 0.5$ and $T = 2$.

B.2 REPLAY-BASED METHODS

Replay-based methods mitigate forgetting by reusing data from previous tasks when training new ones. We implement two representative replay strategies: **Replay (offline replay).** After completing training on the current task, we perform an additional replay phase where we uniformly sample 1% of data from each previous task stored in a unified memory buffer. These real samples are used for extra gradient steps to reinforce previously learned knowledge.
Replay-online (online replay). During each training iteration of a new task, we dynamically sample approximately 1% of data from the same memory buffer and mix it with the current task's mini-batch. This enables continuous rehearsal throughout training, improving stability in knowledge retention.

Both strategies rely solely on real data and share the same unified memory buffer. No synthetic data or auxiliary generative models are used. We also enforce uniform sampling across tasks to ensure fairness in the replay process.

B.3 PARAMETER-EFFICIENT TUNING (PET) BASELINES

SeqLoRA (Sequential Low-Rank Adaptation):
LoRA (Hu et al., 2022) introduces a trainable low-rank update to the weight matrix of attention layers. For a linear transformation $h = Wx$, LoRA reparameterizes the weight as:

$$W' = W + \Delta W, \quad \Delta W = BA, \tag{20}$$

where $A \in \mathbb{R}^{r\times d}$, $B \in \mathbb{R}^{d\times r}$, and $r = 8$ is the rank used in our experiments. A scaling factor α is applied to stabilize optimization (set to 32):

$$\Delta W = \frac{\alpha}{r} BA, \tag{21}$$

We follow common practice and apply LoRA to the attention projection matrices q_proj and v_proj, with dropout set to 0.1. SeqLoRA shares the same LoRA parameters across all tasks, enabling a simple sequential PET baseline.
O-LoRA (Orthogonal LoRA).
O-LoRA (Wang et al., 2023a) prevents interference across tasks by enforcing orthogonality between the LoRA subspaces of different tasks. For LoRA parameters A_i and A_j of task i and j, the orthogonality regularizer is:

$$\mathcal{L}_{\text{ortho}} = \gamma \left\| A_i^\top A_j \right\|_F^2 . \tag{22}$$

and the final test-time weight combines LoRA updates from all tasks:

$$W_{\text{test}} = W_0 + \sum_t B_t A_t. \tag{23}$$

LoRA hyperparameters follow the same settings as SeqLoRA, with orthogonality coefficient $\gamma = 0.5$.
LayerNorm.
LayerNorm (Zhao et al., 2024a) updates only the LayerNorm parameters while keeping all other model weights frozen.
MIGU (MagnItude-based Gradient Updating for continual learning).
MIGU (Du et al., 2024) is an unstructured PET method that selectively updates parameters based on their output magnitude. For a linear layer with weight $W \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ and input x, the response of the i-th output channel is:

$$h_i = W_i \cdot x \tag{24}$$

and its importance score is the L1 norm:

$$n_i = \|h_i\|_1. \tag{25}$$

During training, only the top 0.1% parameters ranked by these scores are updated, while gradients on the remaining parameters are masked out, making the update pattern consistent with our proposed method.

B.4 TASK ORDERS

To evaluate the sensitivity of continual learning performance to task order, we consider three different sequences for the language tasks. Each sequence contains all eight TRACE tasks, with Py150 fixed as the final task to reduce its direct influence on HumanEval-based programming capability evaluation. The task orders are:

Order 1: C-STANCE → FOMC → MeetingBank → ScienceQA → NumGLUE-cm → NumGLUE-ds → 20Minuten → Py150.

Order 2: NumGLUE-cm → NumGLUE-ds → FOMC → 20Minuten → C-STANCE → MeetingBank → ScienceQA → Py150.

Order 3: MeetingBank → FOMC → 20Minuten → C-STANCE → ScienceQA → NumGLUE-cm → NumGLUE-ds → Py150.

C METRICS

Let the performance score of the model on task i after learning task t be denoted as $R_{t,i}$. For classification tasks or question-answering tasks with single-token labels, the performance score is measured by **accuracy**; for other tasks, the performance score is the average of the **Rouge-L** and **BLEU** scores. We use the following metrics to assess the learning effects of different methods on sequential tasks:

C.1 OVERALL PERFORMANCE (OP)

Overall performance measures the average performance of the model across all learned tasks after training task t:

$$OP_t = \frac{1}{t}\sum_{i=1}^{t} R_{t,i} \tag{26}$$

C.2 BACKWARD TRANSFER (BWT)

Backward transfer quantifies how learning new tasks affects the performance on previously learned tasks:

$$BWT_t = \frac{1}{t-1}\sum_{i=1}^{t-1}(R_{t,i} - R_{i,i}) \tag{27}$$

where $R_{i,i}$ is the performance on task i immediately after its training. Positive BWT indicates that new task learning improves old tasks, while negative BWT indicates forgetting. **Note**: OP and BWT are calculated only on sequential tasks to enable fair comparison with other baselines.

C.3 ORIGINAL CAPABILITY RETENTION

Language Models (Programming Ability)
For language models, we use the HumanEval dataset to evaluate code generation ability. Pass@1 is the fraction of generated solutions that pass all unit tests.
Multimodal Models (Image Captioning Ability)
For multimodal models, we use the Flickr30K dataset, with performance measured as the average of Rouge-L and BLEU scores. BLEU-N is defined as:

$$\begin{aligned} \mathrm{BLEU}_N &= \mathrm{BP}\cdot\exp\left(\frac{1}{N}\sum_{n=1}^{N}\log p_n\right), \\ \mathrm{BP} &= \begin{cases} 1, & c > r \\ e^{1-r/c}, & c \le r \end{cases} \end{aligned} \tag{28}$$

where c is the length of the generated sequence, r is the reference sequence length, and p_n is the n-gram precision. ROUGE-L is defined as:

$$\begin{aligned} \text{ROUGE-L} &= \frac{(1+\beta^2)\cdot P \cdot R}{R + \beta^2 \cdot P}, \\ P &= \frac{\text{LCS}(X,Y)}{|X|}, \\ R &= \frac{\text{LCS}(X,Y)}{|Y|}, \\ \beta &= 1 \end{aligned} \tag{29}$$

where X is the generated sequence, Y is the reference sequence, and $\text{LCS}(X,Y)$ denotes the length of the longest common subsequence.

D Extended Experimental Analysis

This section provides additional experimental details and analyses, including the distribution of key parameters in the PIECE-F and PIECE-S models, as well as the overlap of importance parameters across tasks and their corresponding correlation coefficients, and it presents the PIECE training logs of selected models, to help readers gain a more comprehensive understanding of how the proposed method operates.

D.1 Distribution of Critical Parameters

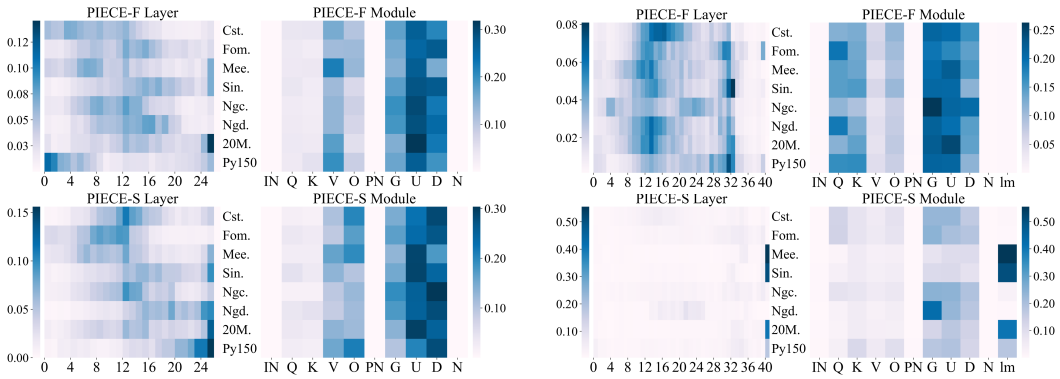
Figure 5a-5e illustrate the distribution of critical parameters across different models. We adopt a unified set of abbreviations for the major network modules within the model: the attention-related modules (q_proj, k_proj, v_proj, o_proj) are denoted as Q, K, V, and O, respectively; the feed-forward modules gate_proj, up_proj, and down_proj are denoted as G, U, and D; the normalization layers input_layernorm, post_attention_layernorm, and norm are denoted as IN, PN, and N; and the final output layer lm_head is denoted as lm. These abbreviations are used consistently throughout the paper. To better observe the important parameters in large-scale Qwen3-14B model, we expanded the proportion of selected important parameters from 0.1% to 1%, as shown in Figure 5b and 5c.

Experimental results indicate that **architectural characteristics** and **task attributes** all potentially influence the distribution of important parameters within a network. First, **from the perspective of model architecture**, the distribution of important parameters differs significantly across architectures, while models within the same architecture exhibit a high degree of similarity. For example, despite notable differences in task types, the distributions of important parameters across layers and submodules in Qwen3-VL-4B and Qwen3-14B are largely consistent in Figure 5c and 5d. Similarly, LLaVA-1.5-7B and Llama3-8B, which share the LLaMA architecture, show comparable patterns in Figure 3 and 5e. Moreover, Gemma2-2B exhibits update patterns similar to those observed in LLaMA architecture in Figure 5a.

Second, **task attributes likely constitute another key factor influencing the distribution of important parameters.** Take the Qwen series as an example, although the 4B VLMs and 14B LMs share the same architecture, their preferences within feed-forward modules differ in Figure 5c and 5d. The 4B VLM tends to select higher-importance parameters in the up-proj (U) module, whereas the 14B LM shows greater involvement in the gate-proj (G) module.

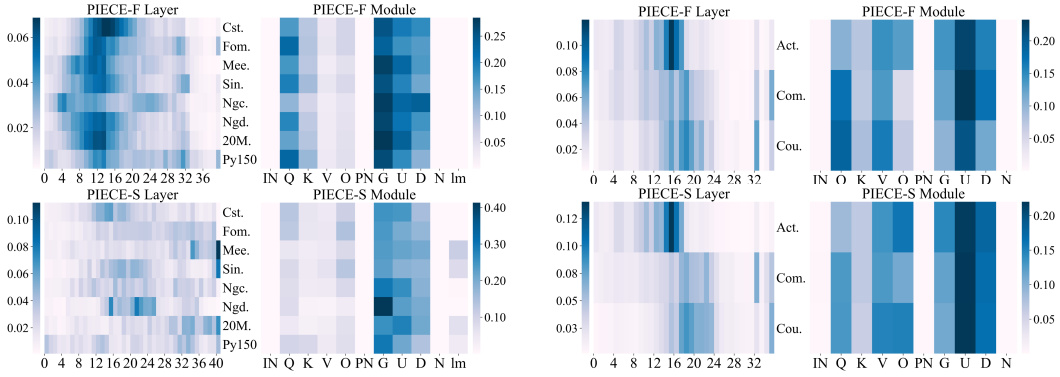
Although different models may vary in their objectives and intensity of parameter "protection" due to architectural and task differences, **all models consistently avoid updating certain parameters across layers and modules.** We posit that this parameter protection mechanism is a key reason why unstructured parameter-efficient fine-tuning achieves strong performance in continual learning.

Taken together, these observations suggest that model architecture and task attributes jointly influence the distribution of important parameters. While these hypotheses require further systematic investigation, they at least indicate that **evaluating important parameters through manual design or fixed heuristics is extremely challenging**, and they further highlight the practical significance of PIECE.



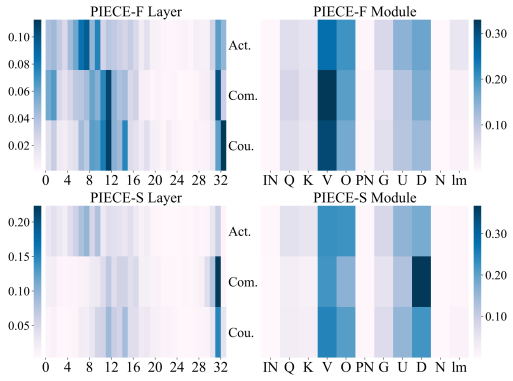
(a) Gemma2-2B.

(b) Qwen3-14B ($k = 0.1\%$).



(c) Qwen3-14B ($k = 1\%$).

(d) Qwen3-VL-4B.



(e) LLaVA-1.5-7B.

Figure 5: Distribution of critical parameters across different models and settings.

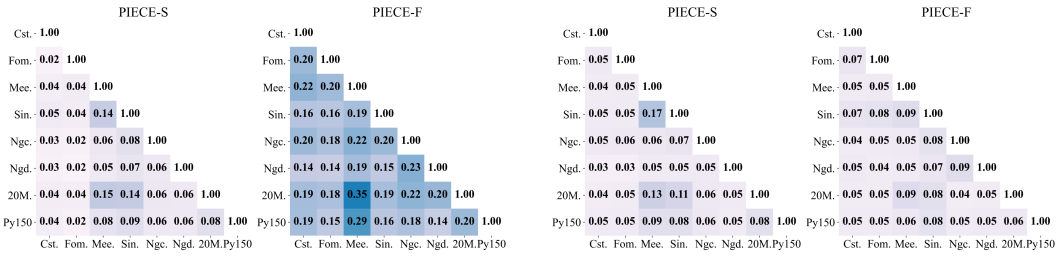
D.2 Distribution Overlap and Correlation Across Tasks

Figures 6a-6d illustrate the coverage of important parameters across tasks for various models. The results indicate that, in general, as model size increases, the proportion of parameters shared across tasks decreases, which aligns with intuition. Nevertheless, even for the smaller 2B model (Figure 6a), task-wise coverage remains relatively low. Across all model sizes, PIECE-S consistently exhibits lower coverage than PIECE-F, indicating greater task specificity in parameter allocation.

Moreover, although the absolute coverage values differ across models, the relative overlap patterns between tasks remain largely consistent. For instance, if two tasks show higher parameter overlap than other task pairs in one model, this relative trend tends to persist in other models as well, suggest-

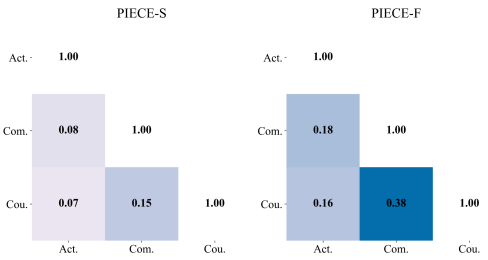
	Gemma2-2B		Llama3-8B		Qwen3-14B	
	PIECE-F	PIECE-S	PIECE-F	PIECE-S	PIECE-F	PIECE-S
Pearson r	0.03	-0.06	0.01	-0.04	-0.39	-0.30
Pearson p	0.88	0.75	0.96	0.83	0.45	0.39
Spearman ρ	0.01	-0.17	0.06	-0.01	-0.42	-0.29
Spearman p	0.97	0.39	0.77	0.98	0.39	0.40

Table 5: Importance Parameter Correlations Across Tasks. Pearson r and Spearman ρ indicate correlation strength (closer to ±1 = stronger correlation, near 0 = weak/no correlation); p-values indicate significance (closer to 0 = significant, near 1 = not significant).

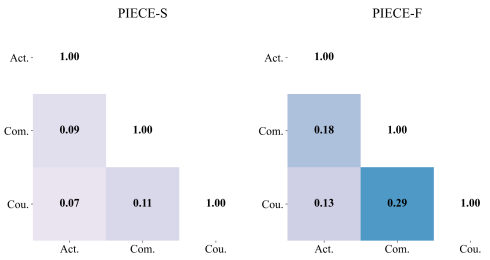


(a) Gemma2-2B.

(b) Qwen3-14B.



(c) Qwen3-VL-4B.



(d) LLaVA-1.5-7B.

Figure 6: Parameter overlap across tasks under different models.

ing that certain task relationships are reflected robustly in parameter sharing patterns, independent of model size.

Table 5 further quantifies the relationship between coverage and task forgetting, showing that task forgetting is weakly or very weakly correlated with the extent of parameter overlap. This indicates that while overlap exists between important parameters across tasks, it does not directly predict forgetting, highlighting the complex relationship between parameter sharing and continual learning performance.

D.3 Training Logs and Parameter Dynamics

We present representative PIECE training logs in Figures 7-9, showing the task-specific validation-loss trajectories throughout continual learning. To reduce computational cost, we constructed a validation subset for each dataset by randomly sampling 200 examples disjoint from the training and test sets. Loss was evaluated on these fixed subsets throughout training. The dataset abbreviations on the horizontal axis indicate the task being trained.

The results suggest that, compared with SeqFT, PIECE effectively mitigates interference across tasks during training, allowing the model to progressively improve its overall performance in a continual learning setting. Notably, the mathematical reasoning tasks Ngc. and Ngd. appear to be more strongly influenced by other tasks, which may indicate that these tasks impose higher demands on the model's integrated reasoning capabilities.

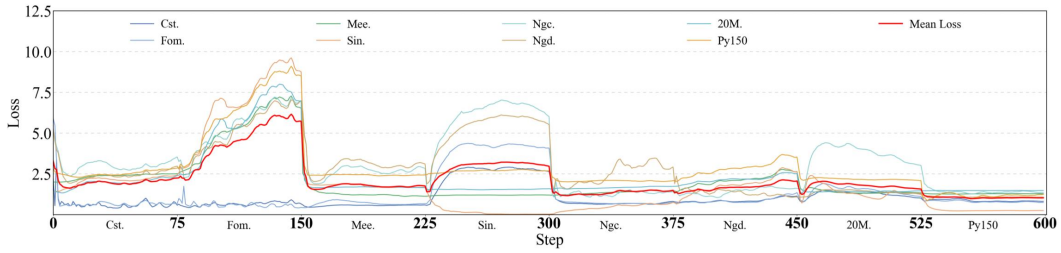


Figure 7: Task-wise Validation Loss Trajectories for Llama3-8B (SeqFT).

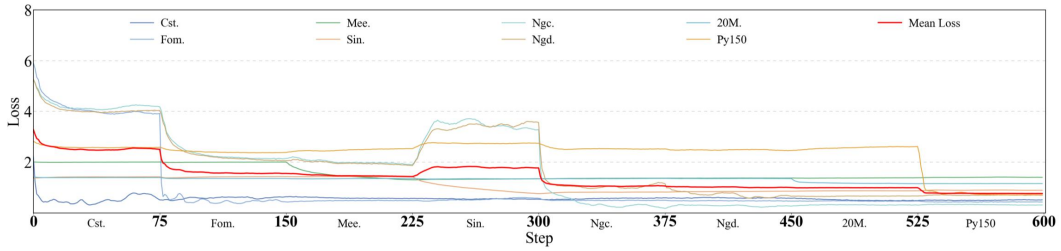


Figure 8: Task-wise Validation Loss Trajectories for Llama3-8B (PIECE-F).

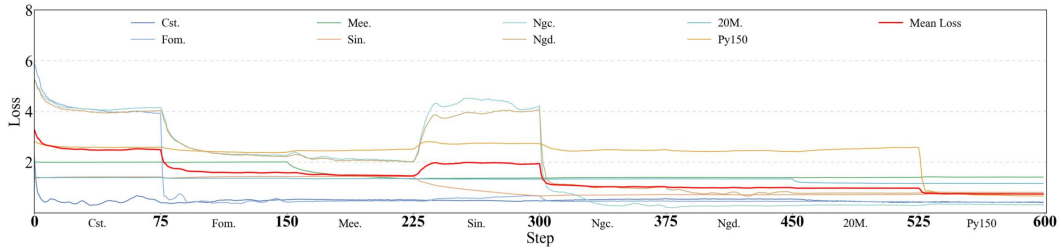


Figure 9: Task-wise Validation Loss Trajectories for Llama3-8B (PIECE-S).

E SuperNI Task Details

The 25 Super-NI tasks utilized in this study were selected to cover a diverse range of task types. Detailed information and their sequence order are provided in Table 6.

F Fair Comparison under Identical Parameter Budgets

To ensure a controlled comparison, we evaluated SeqLoRA, O-LoRA, and PIECE on Llama 3-8B using the TRACE dataset. We adjusted the SeqLoRA and O-LoRA configurations to match PIECE's 0.1% trainable-parameter budget. This constraint ensured that performance differences were not simply attributable to variations in the number of trainable parameters. As shown in Table 7, PIECE consistently outperformed both baselines under the matched budget, suggesting that it uses the limited adaptation capacity more effectively.

Task Order	Task Name	Category
1	task116	Commonsense Classification
2	task146	Text Matching
3	task150	Text Matching
4	task158	Program Execution
5	task209	Stance Detection
6	task297	Text Completion
7	task317	Stereotype Detection
8	task319	Stereotype Detection
9	task342	Gender Classification
10	task343	Gender Classification
11	task383	Misc.
12	task388	Information Extraction
13	task389	Question Generation
14	task398	Sentiment Analysis
15	task607	Toxic Language Detection
16	task934	Text Simplification
17	task1284	Text Quality Evaluation
18	task1339	Fill in The Blank
19	task1399	Question Answering
20	task1400	Wrong Candidate Generation
21	task1401	Sentence Composition
22	task1418	Word Relation Classification
23	task1495	Text Categorization
24	task1583	Word Relation Classification
25	task1714	Dialogue Generation

Table 6: Sequence Order and Task Types of the Super-NI Dataset.

Methods	HumanEval(↑)	$OP(\uparrow)$	$BWT(\uparrow)$
SeqLoRA	26.22	39.14	-3.85
O-LoRA	28.05	43.47	-2.87
PIECE-F (ours)	29.88	**50.05**	-2.66
PIECE-S (ours)	**32.32**	49.05	**-1.78**

Table 7: Fair comparison under 0.1% parameter budget.