LEARNING WHAT TO PRESERVE: PARAMETER ATTRIBUTION FOR CONTINUAL LEARNING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) often suffer from catastrophic forgetting in continual learning: after learning a subsequent new task, they perform worse on earlier tasks. Existing replay and regularization based methods constrain model updates to mitigate forgetting, but may not reliably identify parameters that should be preserved or adapted. Common importance measures, such as Fisher information and gradient-based signals, capture local sensitivity rather than parameter contributions to model predictions. We propose a parameter-level attribution framework that traces output logits to individual parameters, enabling a mechanistic analysis of how the locations of important parameters overlap across tasks. We train LLMs independently on different tasks, analyze the resulting parameter importance patterns, and find that dissimilar tasks show lower overlap in LLMs layers. Motivated by this finding, we estimate parameter importance within each layer and selectively constrain updates to parameters important for previous tasks, while allowing less important parameters to adapt to new tasks. Experiments under both full fine-tuning and LoRA fine-tuning show that our method consistently outperforms all baselines across task settings on Llama and Qwen models, improving performance by at least 2 percentage points over replay-based methods and 3 percentage points over parameter importance based methods.

1 INTRODUCTION

Large Language Models (LLMs) (Achiam et al., 2023; Touvron et al., 2023a; Liu et al., 2024; Hui et al., 2024) acquire broad knowledge and capabilities through large scale pretraining. LLMs have achieved strong performance in tasks such as multi-step reasoning(Wei et al., 2022), instruction following (Chung et al., 2024), and code generation (Roziere et al., 2023). However, as data distributions and task requirements evolve, LLMs need continually acquire new knowledge and capabilities (Jin et al., 2022; Gupta et al., 2023). Since retraining from scratch or on all accumulated data is computationally expensive (Wang et al., 2023), LLMs are typically updated using newly available data. When such updates are performed incrementally over time, it naturally give to a continual learning setting. A key challenge in continual learning is catastrophic forgetting (McCloskey & Cohen, 1989), i.e., performance degradation on prior tasks after learning new ones.

Traditional continual learning methods include replay-based and regularization-based methods (Wang et al., 2024). Replay-based methods (Huang et al., 2024b; Scialom et al., 2022) mitigate forgetting by revisiting stored examples, but limited replay data may provide insufficient constraints on model updates, leading to the overwriting of critical parameters and underutilization of less important ones for adaptation. Some regularization-based methods estimate parameter importance and penalize changes to important parameters, using Fisher information (Kirkpatrick et al., 2017) or gradient-based signals (Sun et al., 2024). However, such metrics capture local sensitivity of the loss or parameter perturbations rather than a parameter's contribution to model predictions, and may therefore fail to identify parameters critical to preserving learned behaviors. This motivates us to measure parameter importance directly from its contribution to model predictions.

Continual learning requires not only identifying important parameters but also understanding how their importance changes across tasks. When two tasks rely on largely overlapping sets of important

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.

parameters, updating these shared parameters for a new task may affect previously learned behaviors. In contrast, when the important parameter sets overlap less, the model can learn new tasks with less interference. We develop an explanation method (Section 4.1) that decomposes the predicted next-token logit into parameter-level contributions across layers, such that the contributions sum to the predicted logit. Given T tasks, we independently fine-tune the same pretrained LLM on each task. For each resulting model, we compute parameter contribution matrices on examples with high-quality outputs and average them to obtain task-level importance scores for each parameter matrix $W^{(l)}$ at layer l. We then select the **indices** of the top-K most important parameters in each $W^{(l)}$ and measure the pairwise overlap of these index sets across tasks. Figure 1 reports overlaps across layers for the attention output projection of Llama-3.2-Instruct-3B on Concept-1K and TRACE when $K = 10\%$. Results for additional Llama parameters and Qwen2.5 parameters are reported in Appendix A.11 and A.12.

Before interpreting these overlap patterns, we first verify that the identified important parameters are indeed task relevant. Masking the top-10% important parameters leads to a larger accuracy drop than random masking, indicating that the scores faithfully capture task-relevant predictions (Figure 1 (right)). Then, we observe a task-dependent pattern in these overlaps: similar tasks exhibit greater overlap in the deeper layers of LLMs, whereas dissimilar tasks show lower overlap. This finding complements prior observations that higher layers in LLMs increasingly encode semantic and task-specific information (Durrani et al., 2022; Ju et al., 2024). On Concept-1K, whose tasks share a short-answer QA format and are closely related, overlap increases with depth and is highest in the final layers. TRACE contains diverse tasks, including stance classification (Tasks 0, 1), meeting summarization (Task 2), Python code completion (Task 3), science QA (Task 4), and numerical reasoning (Tasks 5, 6). Consistent with Concept-1K, Tasks 0, 1 and 5, 6 show higher overlap in deeper layers, whereas dissimilar tasks, such as Tasks 2, 3, maintain low overlap.

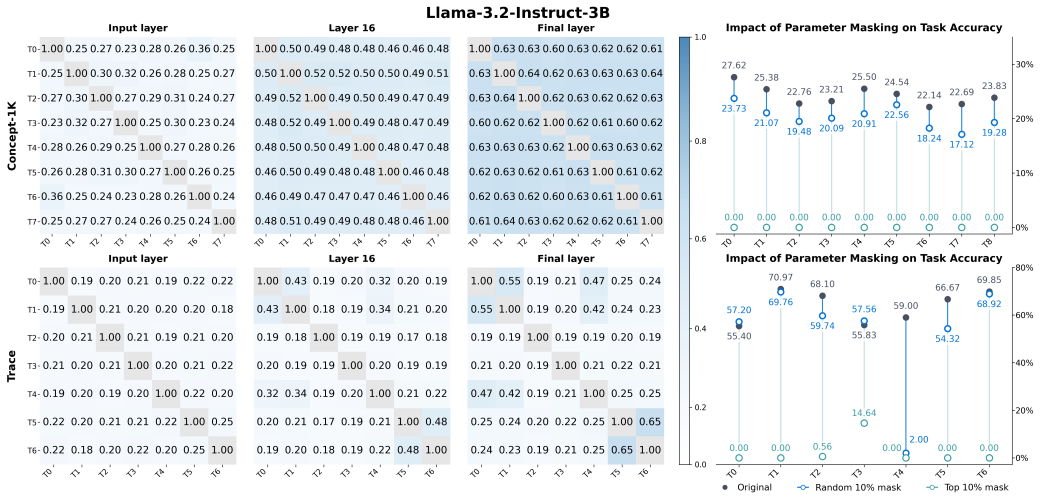

Figure 1: Overlap of important parameter locations in the attention output projection of Llama-3.2-Instruct-3B. We report pairwise overlaps across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. Overlap increases with depth on Concept-1K, while related TRACE tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap in deeper layers. Right: Black denotes independently fine-tuned models, blue random masking of 10% of parameters, and cyan masking of the top-10% important parameters. Masking important parameters causes a substantially larger performance drop than random masking.

The task-dependent structure of parameter importance observed in independently fine-tuned models suggests a natural strategy for continual learning: protect parameters important to previous tasks while leaving less important ones flexible for adaptation. We therefore propose an attribution-guided continual learning framework that identifies and protects task-relevant parameters (See Figure 3). For each task t, we first fine-tune the base LLM θ to obtain θ'_t and collect a subset $\mathcal{D}'_t$ on which the model produces high-quality outputs. We then compute parameter-level attributions to the next-token logits, $R^{(k)}(W^{(l,t)})$, for each $x_k \in \mathcal{D}'_t$ and average them to obtain task level importance scores

$R(W^{(l,t)})$ (Figure 3 (a)). During continual fine-tuning on task t, we construct a gradient mask from the maximum importance over previous tasks $M^{(l,t)} = 1 - \max_{0 \le i < t} R\left(W^{(l,i)}\right)$, which suppresses updates to previously important parameters while leaving less important ones flexible for learning new tasks (Figure 3 (b)). Experiments on LLMs under both full supervised fine-tuning and LoRA-based adaptation demonstrate that our framework consistently outperforms existing continual fine-tuning baselines on Llama and Qwen models. We improved performance by at least 2 percentage points over replay-based methods and 3 percentage points over parameter importance based methods.

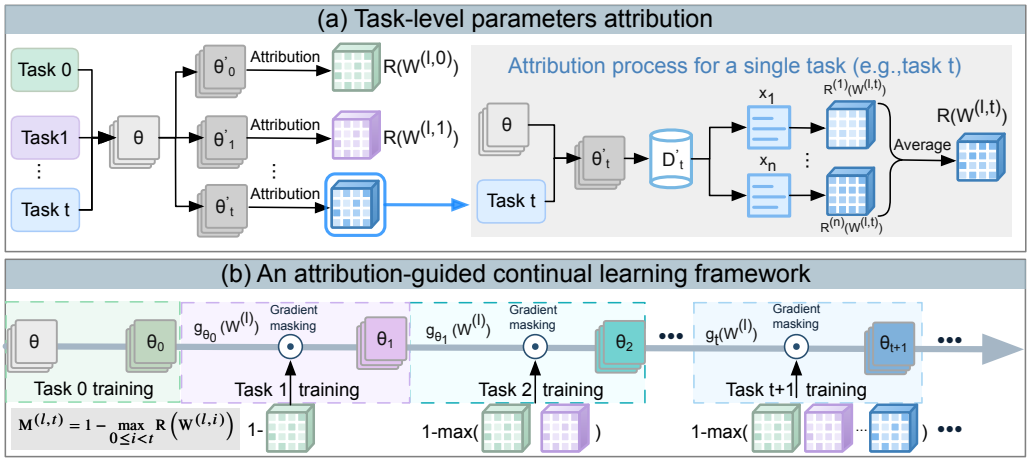

Figure 2: Overview of attribution-guided continual learning framework. **(a)**: For each task, we independently fine-tune the base LLM and average parameter-level attributions over sample data to estimate task-specific parameter importance. **(b)**: During continual learning, we use the maximum importance across previous tasks as the gradient mask, suppressing updates to previously important parameters while leaving the remaining parameters flexible for learning new tasks.

2 Related Work

Continual learning. Classical continual learning commonly mitigates catastrophic forgetting include replay and regularization (Wang et al., 2024). Replay-based methods revisit previous-task information during new-task training; for example, DER++ (Buzzega et al., 2020) combines a memory buffer with knowledge distillation. Some regularization-based methods constrain updates to important parameters. EWC (Kirkpatrick et al., 2017) estimates parameter importance using the Fisher information, while SNIP (Lee et al., 2019) measures parameter saliency through first-order connection sensitivity. Recent LLM continual learning methods extend these strategies. Zheng et al. (Zheng et al., 2025) preserve prior knowledge by freezing lower layers and fine-tuning upper layers, while SSRp (Huang et al., 2024a) generates and filters synthetic samples for rehearsal. KPIG He et al. (2025) selects replay samples based on information gain. However, existing methods rarely examine continual learning from a mechanistic perspective, leaving both the task specific importance of individual parameters and how to protect them during continual fine-tuning unexplored.

Explainability on LLMs. Prior work has investigated how model predictions can be attributed to input tokens and internal components. Layer-wise Relevance Propagation (LRP) (Bach et al., 2015) attributes predictions by backpropagating relevance scores to inputs, and has been adapted to Transformers (Chefer et al., 2021). AttnLRP (Achtibat et al., 2024) further extends LRP to LLMs with attention-aware relevance propagation. Beyond input attribution, mechanistic interpretability examines how internal components contribute to model behavior. Transformer circuit analysis (Elhage et al., 2021) studies interactions among model components, while edge-pruning methods (Conmy et al., 2023) identify influential connections in computational graphs. However, existing approaches mainly explain model behavior at the token, component, or connection level, leaving how to quantify task-specific importance at the parameter level and exploit it to guide continual learning unexplored.

3 BACKGROUND

Background on LLMs. LLMs can be formulated as deep compositions of functions that progressively transform token representations (Aubry et al., 2024). Given an input prompt tokenized as $x = [x_1, \ldots, x_m]$, each token x_i is mapped to a d-dimensional embedding with positional information, forming the initial hidden states $H^{(0)} = [h_1^{(0)}; \ldots; h_m^{(0)}] \in \mathbb{R}^{m \times d}$. The hidden states are then processed by L Transformer blocks, each comprising multi-head self-attention (MHA), a feed-forward network (FFN), layer normalization (LN), and residual connections. Let $H^{(l)}$ denote the hidden states after the l-th block. The $(l+1)$-th block computes

$$f^{(l)} = \mathrm{MHA}\big(\mathrm{LN}(H^{(l)})\big), \quad g^{(l)} = \mathrm{LN}\big(H^{(l)} + f^{(l)}\big), \quad H^{(l+1)} = H^{(l)} + f^{(l)} + \mathrm{FFN}\big(g^{(l)}\big). \tag{1}$$

After L layers, the $H^{(L)}$ are mapped to vocabulary logits as

$$Z = \mathrm{LN}(H^{(L)})W_{\mathrm{vocab}} \in \mathbb{R}^{m \times |\mathcal{V}|}, \tag{2}$$

where $\mathcal{V}$ denotes the vocabulary, $|\mathcal{V}|$ is its size, and $W_{\mathrm{vocab}} \in \mathbb{R}^{d \times |\mathcal{V}|}$ is the output projection matrix. The next-token prediction is determined by the logits at the final position: $\hat{j} = \arg\max_{j \in \{1,\ldots,|\mathcal{V}|\}} Z_{m,j}$, $\hat{x}_{m+1} = \mathcal{V}[\hat{j}]$.

Background on continual learning. We consider a sequence of tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$, where each task $\mathcal{T}_t$ is associated with a dataset $\mathcal{D}_t = \{(x^{(k)}, y^{(k)})\}_{k=1}^{N_t}$. $x^{(k)}$ and $y^{(k)}$ denote the input prompt and target response of the k-th example, respectively. Let θ_t denote the model parameters after training on task $\mathcal{T}_t$. At stage t, he model is initialized from θ_{t-1} and optimized on $\mathcal{D}_t$ using the standard autoregressive objective:

$$\mathcal{L}_t(\theta) = -\mathbb{E}_{(x^{(k)}, y^{(k)}) \sim \mathcal{D}_t} \left[\frac{1}{n} \sum_{j=1}^{n} \log p_\theta \left(y_j^{(k)} \mid x^{(k)}, y_{<j}^{(k)} \right) \right]. \tag{3}$$

4 METHOD

We propose an attribution-guided continual learning framework that uses parameter-level relevance to regulate parameter updates across tasks. Given a sequence of tasks, we independently fine-tune the same pretrained LLM on each task and compute element-wise parameter relevance with respect to its next-token logits (Section 4.1). We aggregate these relevance scores into task-specific importance scores. During continual learning, we combine the importance scores from previous tasks to construct an element-wise gradient mask, which suppresses updates to parameters important for prior tasks while allowing less important parameters to adapt to the current task (Section 4.2).

4.1 CALCULATING ELEMENT-WISE IMPORTANCE OF LLM PARAMETERS

To identify task-important parameters, we apply Layer-wise Relevance Propagation (LRP) Bach et al. (2015) to quantify each parameter's contribution to the predicted next-token logit. Figure 3 illustrates the attribution procedure. Figure 3 (a) shows the forward computation of a Transformer block. Starting from $R(H^{(L)})$, initialized from the predicted next-token logit, relevance is propagated backward across layers. As shown in Figure 3 (b), the output relevance $R(H^{(l+1)})$ at each layer is decomposed along the residual branches and through the FFN and MHA. Proposition 4.2 derives relevance for the FFN parameters and input, while Proposition 4.3 gives the corresponding decomposition for MHA. The resulting $R(H^{(l)})$ is recursively propagated to preceding layers, yielding element-wise relevance scores for all Transformer parameters. We next review the standard LRP formulation and its extension to Transformers.

4.1.1 LRP-BASED INPUT ATTRIBUTION

Original LRP. LRP attributes the prediction score to input neurons by propagating relevance backward through the network. For a neuron $a^{l+1} = \psi([a_1^{(l)}, \ldots, a_n^{(l)}])$, where ψ may be a linear function

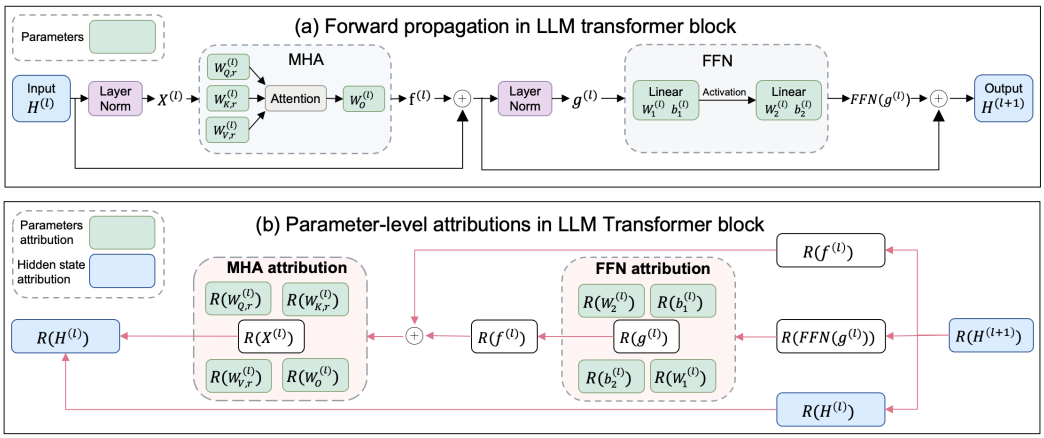

Figure 3: Forward propagation and LRP-based attribution in an LLM Transformer Block. Green boxes denote parameters or parameters attributions, while blue boxes denote inputs/outs and their corresponding attributions.

or a composition of a linear function with a nonlinear activation, its relevance R^{l+1} is attributed to neurons in the preceding layer as:

$$R_i^{(l)} = \frac{a_i^{(l)} w_i^l}{\sum_{i'} a_{i'}^{(l)} w_{i'}^l} R^{(l+1)}, \quad \sum_i R_i^{(l)} = R^{(l+1)}. \tag{4}$$

where w_i^l denotes the weight connecting neuron $a_i^{(l)}$ to neuron $a^{(l+1)}$. For a multi-layer network, Eq. (4) is recursively applied from the output logit $R^{(L)}$ to the input layer, producing input relevance scores $R_i^{(0)}$. The $R_i^{(0)}$ is computed as follows:

$$R_i^{(0)} = \sum_{j,\ldots,k} \frac{a_i^{(0)} w_i^0}{\sum_{i'} a_{i'}^{(0)} w_{i'}^0} \cdots \frac{a_k^{(L-1)} w_k^{L-1}}{\sum_{k'} a_{k'}^{(L-1)} w_{k'}^{L-1}} \cdot R^{(L)}, \quad \sum_i R_i^{(0)} = R^{(L)} \tag{5}$$

LRP in Transformer. Standard LRP is not directly applicable to multiplicative interactions of the form $a^{(l+1)} = \sum_{i=1}^n a_i^{(l)} \cdot c_i^{(l)}$, where both $a_i^{(l)}$ and $c_i^{(l)}$ are input neurons whose relevance needs to be attributed. Such interactions are prevalent in Transformer self-attention due to matrix multiplication. To address this issue, AttnLRP Achtibat et al. (2024) treats matrix multiplication as a bilinear operation: the relevance of $a_i^{(l)}$ is computed with $c_i^{(l)}$ fixed, and vice versa. Formally, given the output relevance score $R(a^{(l+1)})$, the relevance assigned to $a_i^{(l)}$ and $c_i^{(l)}$, together with the conservation property, is given by:

$$R(a_i^{(l)}) = R(c_i^{(l)}) = \frac{1}{2} \frac{a_i^{(l)} c_i^{(l)}}{\sum_{j=1}^n a_j^{(l)} c_j^{(l)}} R(a^{(l+1)}), \quad \sum_{i=1}^n R(a_i^{(l)}) + \sum_{i=1}^n R(c_i^{(l)}) = R(a^{(l+1)}) \tag{6}$$

4.1.2 LRP-BASED PARAMETERS ATTRIBUTION IN LLMS

Existing LRP methods, including their Transformer extensions, primarily attribute predictions to inputs, while leaving **model parameters unattributed**. We adopt the bilinear attribution principle of AttnLRP to the FFN and MHA modules and derive **parameter-level relevance scores for each LLM layer** to the next token logit $Z_{m,\hat{j}}$, where $\hat{j} = \arg\max_{j \in \{1,\ldots,|\mathcal{V}|\}} Z_{m,j}$. These relevance scores are then used to guide fine-tuning process.

Lemma 4.1. *For a linear transformation $C = PW + B$, where $P \in \mathbb{R}^{m \times d_{\text{in}}}$, $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, and $B, C \in \mathbb{R}^{m \times d_{\text{out}}}$. Let $\odot$ denote element-wise multiplication, and the division is applied element-wise. Given the output relevance $R(C) \in \mathbb{R}^{m \times d_{\text{out}}}$, the relevance assigned to P, W, and B under*

the bilinear LRP rule is obtained from the following formula and satisfies the conservation property:

$$R(P) = \frac{1}{2}P \odot \left[\left(\frac{R(C)}{C}\right)W^\top\right], R(W) = \frac{1}{2}W \odot \left[P^\top\left(\frac{R(C)}{C}\right)\right], R(B) = B \odot \frac{R(C)}{C},$$
$$\sum R(P) + \sum R(W) + \sum R(B) = \sum R(C), \tag{7}$$

The proof of Lemma 4.1 is provided in Appendix A.1. Lemma 4.1 quantifies the contributions of input activations, model parameters, and biases in matrix multiplication. We apply this result to derive parameter-level relevance scores for the FFN and MHA modules in LLMs.

LRP Attribution in FFN. For the l-th Transformer layer, let $\Delta^{(l)}_{\mathrm{FFN}} = \mathrm{FFN}\big(g^{(l)}\big)$, the FFN is given by as

$$U^{(l)} = g^{(l)}W_1 + b_1, \quad S^{(l)} = \sigma(U^{(l)}), \quad \Delta^{(l)}_{\mathrm{FFN}} = S^{(l)}W_2 + b_2,$$

where $W_1 \in \mathbb{R}^{d\times d_{\mathrm{ff}}}$, $b_1 \in \mathbb{R}^{d_{\mathrm{ff}}}$, $W_2 \in \mathbb{R}^{d_{\mathrm{ff}}\times d}$, and $b_2 \in \mathbb{R}^d$. Accordingly, $U^{(l)}, S^{(l)} \in \mathbb{R}^{m\times d_{\mathrm{ff}}}$, and the FFN output satisfies $\Delta^{(l)}_{\mathrm{FFN}} \in \mathbb{R}^{m\times d}$. Applying Lemma 4.1 sequentially to the two linear transformations yields the relevance propagation rules for the FFN input and parameters.

Proposition 4.2 (FFN relevance propagation)**.** *Given the output relevance $R(\Delta^{(l)}_{\mathrm{FFN}})$, and treating the activation $\sigma(\cdot)$ as identity relevance propagation , i.e., $R(U^{(l)}) = R(S^{(l)})$. Let $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively, and $\mathbf{1}_m \in \mathbb{R}^{m\times 1}$ be the all-ones vector. Let $\Gamma^{(l)}_1 = R(S^{(l)}) \oslash U^{(l)}, \Gamma^{(l)}_2 = R(\Delta^{(l)}_{\mathrm{FFN}}) \oslash \Delta^{(l)}_{\mathrm{FFN}}$ The relevance scores of the FFN input and parameters are given by*

$$\begin{aligned} &R(S^{(l)}) = \tfrac{1}{2}S^{(l)} \odot [\Gamma^{(l)}_2 W^{(l)\top}_2],\ R(W^{(l)}_2) = \tfrac{1}{2}W^{(l)}_2 \odot [S^{(l)\top}\Gamma^{(l)}_2],\ R(b^{(l)}_2) = b^{(l)}_2 \odot \mathbf{1}^\top_m\Gamma^{(l)}_2, \\ &R(g^{(l)}) = \tfrac{1}{2}g^{(l)} \odot [\Gamma^{(l)}_1 W^{(l)\top}_1],\ R(W^{(l)}_1) = \tfrac{1}{2}W^{(l)}_1 \odot [g^{(l)\top}\Gamma^{(l)}_1],\ R(b^{(l)}_1) = b^{(l)}_1 \odot \mathbf{1}^\top_m\Gamma^{(l)}_1. \end{aligned} \tag{8}$$

The proof of Proposition 4.2 is provided in Appendix A.2. Proposition 4.2 provides relevance scores for both inputs and parameters in LLM FFN modules. These scores further satisfy the following conservation property:

$$\sum R(g^{(l)}) + \sum R(W^{(l)}_1) + \sum R(b^{(l)}_1) + \sum R(W^{(l)}_2) + \sum R(b^{(l)}_2) = \sum R(\Delta^{(l)}_{\mathrm{FFN}}).$$

LRP Attribution in MHA. To propagate relevance further to the preceding MHA module, we treat the LayerNorm as identity relevance propagation and pass the relevance through unchanged,i.e., $R(f^{(l)}) = R(g^{(l)})$. We then propagate $R(f^{(l)})$ through the MHA module. Let $X^{(l)} = \mathrm{LN}(H^{(l)})$, where $H^{(l)}, X^{(l)} \in \mathbb{R}^{m\times d}$. Let d_{head} denote the dimension of each attention head and $n_h = d/d_{\mathrm{head}}$. For $r = 1, \ldots, n_h$, the r-th head is computed as

$$\begin{aligned} &Q^{(l)}_r = X^{(l)}W^{(l)}_{Q,r}, \qquad K^{(l)}_r = X^{(l)}W^{(l)}_{K,r}, \qquad V^{(l)}_r = X^{(l)}W^{(l)}_{V,r}, \\ &A^{(l)}_r = \mathrm{softmax}\left(\frac{Q^{(l)}_r K^{(l)\top}_r}{\sqrt{d_{\mathrm{head}}}}\right), \qquad O^{(l)}_r = A^{(l)}_r V^{(l)}_r, \\ &O^{(l)} = \mathrm{Concat}\Big(O^{(l)}_1, \ldots, O^{(l)}_{n_h}\Big), \qquad f^{(l)} = O^{(l)}W^{(l)}_O. \end{aligned}$$

Where $W^{(l)}_{Q,r}, W^{(l)}_{K,r}, W^{(l)}_{V,r} \in \mathbb{R}^{d\times d_{\mathrm{head}}}$, $Q^{(l)}_r, K^{(l)}_r, V^{(l)}_r, O^{(l)}_r \in \mathbb{R}^{m\times d_{\mathrm{head}}}$, $A^{(l)}_r \in \mathbb{R}^{m\times m}$, $W^{(l)}_O \in \mathbb{R}^{d\times d}$ and $f^{(l)} \in \mathbb{R}^{m\times d}$. Applying Lemma 4.1 to the MHA module gives the following relevance propagation rules for its inputs and parameters.

Proposition 4.3 (MHA relevance propagation)**.** *Given the output relevance $R(f^{(l)})$ of the MHA module, let $[\cdot]_r$ denote the column block associated with the r-th attention head. We treat the softmax operation as identity relevance propagation, $\Gamma^{(l)}_f = R(f^{(l)}) \oslash f^{(l)}$, $R(O^{(l)}) = \frac{1}{2}O^{(l)} \odot \Big(\Gamma^{(l)}_f (W^{(l)}_O)^\top\Big)$, $\Gamma^{(l)}_{O,r} = \big[R(O^{(l)})\big]_r \oslash O^{(l)}_r$. For each attention head, $\Phi^{(l)}_r = \Big(A^{(l)}_r \odot \Big[\Gamma^{(l)}_{O,r}(V^{(l)}_r)^\top\Big]\Big) \oslash E^{(l)}_r$. We further define $G^{(l)}_{Q,r} = \alpha\Phi^{(l)}_r K^{(l)}_r$, $G^{(l)}_{K,r} = \alpha\Phi^{(l)\top}_r Q^{(l)}_r$, and*

$G^{(l)}_{V,r} = \frac{1}{4}A^{(l)\top}_r \Gamma^{(l)}_{O,r}$, *where* $\alpha = \frac{1}{8\sqrt{d_{\text{head}}}}$. *Let* $\odot$ *and* $\oslash$ *denote element-wise multiplication and division, respectively. Then the parameters and input relevance scores of the MHA module are:*

$$\begin{aligned} R(W^{(l)}_O) &= \tfrac{1}{2}W^{(l)}_O \odot \big[O^{(l)\top}\Gamma^{(l)}_f\big], \\ R(W^{(l)}_{P,r}) &= W^{(l)}_{P,r} \odot \big[X^{(l)\top}G^{(l)}_{P,r}\big], \quad P \in \{Q,K,V\}, \\ R(X^{(l)}) &= X^{(l)} \odot \sum_{r=1}^{n_h} \sum_{P\in\{Q,K,V\}} G^{(l)}_{P,r}W^{(l)\top}_{P,r}. \end{aligned} \tag{9}$$

The proof of Proposition 4.3 is provided in Appendix A.3. Moreover, these relevance scores satisfy:

$$\sum R(X^{(l)}) + \sum R(W^{(l)}_O) + \sum_{r=1}^{n_h}\Big[\sum R(W^{(l)}_{Q,r}) + \sum R(W^{(l)}_{K,r}) + \sum R(W^{(l)}_{V,r})\Big] = \sum R(f^{(l)}).$$

LoRA-based attribution. For LoRA-adapted MHA, each projection matrix is written as $\widetilde{W}^{(l)}_O = W^{(l)}_O + A^{(l)}_O B^{(l)}_O$ and $\widetilde{W}^{(l)}_{P,r} = W^{(l)}_{P,r} + A^{(l)}_{P,r}B^{(l)}_{P,r}$, where $P \in \{Q,K,V\}$, and A, B denote the trainable LoRA factors. We compute relevance scores for the frozen backbone weights, the LoRA parameters, and the input. The full derivation is provided in Appendix A.4.

LRP Attribution in Transformer Block. We compose the FFN and MHA propagation rules to obtain block-wise parameter attribution. $H^{(l+1)} = H^{(l)} + f^{(l)} + FFN(g^{(l)})$. Given $R(H^{(l+1)})$, we first decompose the output relevance across the three additive branches, yielding relevance for the input $H^{(l)}$, the MHA output $f^{(l)}$ and the FFN output. Proposition 4.2 propagates the FFN relevance to its parameters and input $g^{(l)}$. We then pass $R(g^{(l)})$ backward through the LayerNorm without redistribution and accumulate the resulting relevance with the relevance already assigned to $f^{(l)}$ by the residual decomposition. The accumulated relevance of $f^{(l)}$ is subsequently propagated through the MHA module using Proposition 4.3, yielding relevance scores for the MHA parameters and its input. Finally, the input relevance from the MHA branch is combined with the residual relevance of $H^{(l)}$, producing the total relevance propagated to the preceding Transformer block. We denote the parameter set by $\mathcal{W}^{(l)} = \Big\{W^{(l)}_O, \{W^{(l)}_{P,r} : P \in \{Q,K,V\}, r = 1,\cdots,n_h\}, W^{(l)}_1, W^{(l)}_2, b^{(l)}_1, b^{(l)}_2\Big\}$. The block-wise propagation can be summarized as $R(H^{(l+1)}) \;\to\; \big\{R(H^{(l)}), R(\mathcal{W}^{(l)})\;\big\}$. The relevance propagation rules for LoRA-adapted Transformer are provided in Appendix A.5.

4.2 Attribution guided fine-tuning for continual learning

Importance prior estimation for each task. For each task, we estimate parameter importance using a model trained independently on that task and use it as a prior for continual learning. Given task $\mathcal{T}_t$ with dataset $\mathcal{D}_t = \{(x^{(k)}, y^{(k)})\}_{k=1}^{N_t}$, we train a task-specific model with parameters θ'_t. Let $\mathcal{W}^{(l)} = \Big\{W^{(l)}_O, \{W^{(l)}_{P,r} : P \in \{Q,K,V\}, r = 1,\cdots,n_h\}, W^{(l)}_1, W^{(l)}_2, b^{(l)}_1, b^{(l)}_2\Big\}$ denote the parameters of the l-th Transformer block, including attention projections, feed-forward weights, and biases. To obtain reliable attribution signals, we only consider samples correctly generated by the task-specific model $\mathcal{C}_t$ and normalize each sample-level relevance map as

$$\widehat{R}^{(k)}(W^{(l,t)}) = \operatorname{sign}\Big(R^{(k)}(W^{(l,t)})\Big) \odot \frac{\log\big(1 + \big|R^{(k)}(W^{(l,t)})\big|\big)}{\max \log\big(1 + \big|R^{(k)}(W^{(l,t)})\big|\big) + \epsilon}. \tag{10}$$

We aggregate attributions into a task-level importance prior by retaining the mean relevance values :

$$R(W^{(l,t)})[u] = \frac{1}{|\mathcal{C}_t|}\widehat{R}^{(k)}(W^{(l,t)})[u]. \tag{11}$$

The element-wise importance $R(W^{(l,t)})$ quantifies which parameters are important in task $\mathcal{T}_t$.

Attribution-guided continual learning During continual learning, for the task t, the $R(W^{(l,1)}), \ldots, R(W^{(l,t-1)})$ are used to protect parameters that are critical to previously learned

tasks. When learning the current task $\mathcal{T}_t$, we summarize historical importance for each parameter by taking the element-wise maximum over all prior tasks:

$$\bar{R}(W^{(l,t)}) = \left[\max_{\tau<t} R(W^{(l,\tau)})\right]_+, \quad \bar{R}(W^{(l,t)}) \in [0,1], \tag{12}$$

where $[x]_+ = \max(x, 0)$ is applied element-wise. We incorporate this prior by modulating the gradient during fine-tuning:

$$M^{(l,t)} = \left(\mathbf{1} - \bar{R}(W^{(l,t)})\right), \quad \widetilde{\nabla}_{W^{(l,t)}}\mathcal{L}_t = M^{(l,t)} \odot \nabla_{W^{(l,t)}}\mathcal{L}_t. \tag{13}$$

Here, $M^{(l,t)}$ serves as an element-wise gradient gate. Parameters that are important to any previous task have large historical relevance, making the gate close to zero and thus receiving smaller updates. Parameters with low historical relevance keep gates close to one, so their gradients are largely unchanged when learning the current task. The Algorithm 1 in appendix shows the overall process.

5 Experiments

Datasets. We evaluate our method on two benchmark datasets. The first is Concept-1K (Zheng et al., 2024), which consists of 10 diverse tasks designed to assess the model's ability to learn and generalize across different concepts. The second is TRACE, from which we select seven representative tasks: FOMC, MeetingBank, PY150, ScienceQA, NumGLUE-cm, NumGLUE-ds, and 20Minute. These tasks cover a broad range of scenarios, including financial text understanding, meeting summarization, code completion, scientific question answering, numerical reasoning, and long-context comprehension. The details of the datasets is shown in Appendix A.7.

Baselines. We compare attribution-guided fine-tuning (AGFT) against seven representative baselines spanning four major continual-learning strategies: sequential fine-tuning (**SEQ**), replay-based methods (**Replay** and **DER++** (Buzzega et al., 2020)), parameter-importance-based methods (**EWC** (Kirkpatrick et al., 2017), **SNIP** (Lee et al., 2019), and **Wanda** (Sun et al., 2024)), and parameter freezing (**Freeze** (Zheng et al., 2025)). **SEQ** sequentially fine-tunes on each task without any explicit mechanism for knowledge preservation. **Replay** jointly trains on current-task data and buffered samples from previous tasks, while DER++ additionally enforces prediction consistency on replayed samples to better retain prior knowledge. **EWC** estimates parameter importance using the diagonal Fisher information and penalizes updates to parameters that are important for previously learned tasks. **SNIP** measures parameter importance via connection sensitivity, which we adapt from its original pruning setting to parameter selection for continual LLM fine-tuning. **Wanda** scores parameters using weight magnitude together with the corresponding input-activation norms. **Freeze** updates only the upper Transformer layers throughout continual training.

Continual learning setting and models. We first independently train a model for each task and estimate parameter importance using our attribution method. During sequential continual learning, the estimated importance scores from previous tasks are used to guide model updates. We evaluate AGFT on four LLM backbones with different model scales, including Llama-3.2-3B-Instruct, Qwen2.5-3B (Yang et al., 2025), Llama-2-7B (Touvron et al., 2023b), and Qwen2.5-7B (Yang et al., 2025). For the 3B models, i.e., Llama-3.2-3B-Instruct and Qwen2.5-3B, we perform full-parameter fine-tuning, whereas for the 7B models, i.e., Llama-2-7B and Qwen2.5-7B, we adopt LoRA (Hu et al., 2022) for parameter-efficient fine-tuning. All four models are evaluated on both Concept-1K (Zheng et al., 2024) and TRACE (Wang et al., 2023). Detailed experimental configurations are provided in Appendix A.9 for completeness and reproducibility.

Evaluation metrics. We evaluate model performance using different metrics for dfferent task type, including Accuracy, ROUGE-L, Edit Similarity, and SARI. Accuracy is computed by exact match for all models. ROUGE-L, Edit Similarity, and SARI are used for summarization, code completion, and text simplification tasks, respectively, with details provided in Appendix A.10.

Performance. AGFT consistently achieves the best performance across both Llama and Qwen backbones on Concept-1K and TRACE, under both full fine-tuning and LoRA adaptation. Compared with **replay-based** baselines, AGFT improves over Replay and DER++ by **3.284** and **2.077** percentage points on average, respectively. Replay-based methods mitigate forgetting by revisiting previously observed examples, but their effectiveness depends on the limited information retained in

the replay buffer and does not explicitly identify which parameters encode task-relevant knowledge. AGFT also consistently outperforms **importance-based** baselines, with average gains of **4.116** and **3.180** points over SNIP and Wanda, respectively. On Concept-1K, AGFT further surpasses EWC by **6.673** points on average. These methods estimate parameter importance using indirect criteria, such as gradient sensitivity, weight magnitude, or Fisher information, which may not faithfully capture the contribution of individual parameters to task-specific model outputs. We do not report EWC results on TRACE, as estimating the Fisher information matrix is memory-intensive at LLM scale and leads to GPU out-of-memory errors on long-sequence examples. Compared with Freeze and standard sequential fine-tuning (SEQ), AGFT achieves average improvements of **4.750** and **4.718** points, respectively. Freeze reduces forgetting by preventing updates to a predefined subset of parameters, but this static constraint can limit the model's capacity to acquire new knowledge. Overall, these results suggest that AGFT achieves a more effective balance between knowledge preservation and adaptation across model families, datasets, and fine-tuning settings.

Table 1: Continual learning performance on Concept-1K. Best results are shown in **bold**, and second-best results are underlined.

Method	Llama3.2-3B (Full)	Llama2-7B (LoRA)	Qwen2.5-3B (Full)	Qwen2.5-7B (LoRA)
SEQ	19.897	21.118	21.284	20.581
Freeze	18.900	17.537	21.187	21.496
EWC	19.863	19.024	18.308	18.925
SNIP	23.387	19.202	25.324	23.098
Wanda	21.507	21.686	21.636	23.185
Replay	20.210	21.373	21.404	23.509
DER++	23.907	22.615	22.421	23.623
AGFT	**26.336**	**23.707**	**28.292**	**24.476**

Table 2: Continual learning performance on TRACE. Best results are shown in **bold**, and second-best results are underlined.

Method	Llama3.2-3B (Full)	Llama2-7B (LoRA)	Qwen2.5-3B (Full)	Qwen2.5-7B (LoRA)
SEQ	46.584	45.661	51.124	51.141
Freeze	51.932	44.532	51.992	49.554
SNIP	52.100	44.262	52.402	42.431
Wanda	51.861	44.699	53.452	51.666
Replay	52.088	45.838	52.028	52.411
DER++	54.114	45.573	54.174	52.091
AGFT	**54.880**	**47.775**	**54.922**	**54.745**

6 Conclusions and Limitations

We study continual learning in LLMs from the perspective of their underlying mechanisms. We introduce an attribution method that decomposes next-token logits into parameter-level contributions. Our analysis reveals a clear task-dependent structure: related tasks exhibit greater overlap in important parameters, particularly in deeper layers, whereas dissimilar tasks show substantially less overlap across the model. Building on this observation, we propose an attribution-guided continual learning framework that selectively protects parameters important to previous tasks while leaving less important ones available for effective adaptation to new tasks. Specifically, we construct gradient masks from accumulated parameter importance to explicitly balance knowledge retention and new-task learning during sequential training. Experiments with both full supervised fine-tuning and LoRA adaptation on Llama and Qwen models show consistent improvements over existing baselines across different model scales and task sequences. We improved performance by at least 2 percentage points over replay-based methods and 3 percentage points over parameter importance based methods. The main limitation of our method is that estimating task-specific parameter importance introduces additional computational overhead compared with standard fine-tuning approaches.

AI USE STATEMENT

We used generative AI tools solely to assist with language editing and improving the clarity and readability of the manuscript.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. Attnlrp: attention-aware layer-wise relevance propagation for transformers. *arXiv preprint arXiv:2402.05602*, 2024.

Murdock Aubry, Haoming Meng, Anton Sugolov, and Vardan Papyan. Transformer block coupling and its correlation with generalization in llms. *arXiv preprint arXiv:2407.07810*, 2024.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 782–791, 2021.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Firoj Alam. On the transformation of latent space in fine-tuned NLP models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1495–1516, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.97. URL https://aclanthology.org/2022.emnlp-main.97/.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to (re) warm your model? *arXiv preprint arXiv:2308.04014*, 2023.

Yongquan He, Wenyuan Zhang, Xuancheng Huang, Peng Zhang, Lingxun Meng, Xiang Zhou, Ke Zeng, and Xunliang Cai. Don't half-listen: Capturing key-part information in continual instruction tuning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23649–23668, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1153. URL https://aclanthology.org/2025.acl-long.1153/.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1416–1428, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.77. URL https://aclanthology.org/2024.acl-long.77/.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1416–1428, 2024b.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. Lifelong pretraining: Continually adapting language models to emerging corpora. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4764–4780, 2022.

Tianjie Ju, Weiwei Sun, Wei Du, Xinwei Yuan, Zhaochun Ren, and Gongshen Liu. How large language models encode context knowledge? a layer-wise probing study. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 8235–8246, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.722/.

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.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, 2022.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

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*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Junhao Zheng, Shengjie Qiu, and Qianli Ma. Concept-1k: A novel benchmark for instance incremental learning. *arXiv preprint arXiv:2402.08526*, 2024.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ScI7IlKGdI.

A APPENDIX

A.1 PROOF OF LEMMA 4.1

Proof. Let $P = [p_1; \ldots; p_m] \in \mathbb{R}^{m \times d_{\text{in}}}$, where $p_i \in \mathbb{R}^{1 \times d_{\text{in}}}$. Given $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ and $\mathcal{B} = [\beta_1; \ldots; \beta_m] \in \mathbb{R}^{m \times d_{\text{out}}}$, where $\beta_i \in \mathbb{R}^{1 \times d_{\text{out}}}$, equivalently, $C = [c_1; \ldots; c_m] \in \mathbb{R}^{m \times d_{\text{out}}}$, where $c_i \in \mathbb{R}^{1 \times d_{\text{out}}}$.

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1d_{\text{in}}} \\ p_{21} & \cdots & p_{2d_{\text{in}}} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{md_{\text{in}}} \end{bmatrix}, \quad W = \begin{bmatrix} w_{11} & \cdots & w_{1d_{\text{out}}} \\ w_{21} & \cdots & w_{2d_{\text{out}}} \\ \vdots & \ddots & \vdots \\ w_{d_{\text{in}}1} & \cdots & w_{d_{\text{in}}d_{\text{out}}} \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1d_{\text{out}}} \\ \beta_{21} & \cdots & \beta_{2d_{\text{out}}} \\ \vdots & \ddots & \vdots \\ \beta_{m1} & \cdots & \beta_{md_{\text{out}}} \end{bmatrix}.$$

$$C = PW + \mathcal{B} = \begin{bmatrix} \sum_{k=1}^{d_{\text{in}}} p_{1k}w_{k1} + \beta_{11} & \cdots & \sum_{k=1}^{d_{\text{in}}} p_{1k}w_{kd_{\text{out}}} + \beta_{1d_{\text{out}}} \\ \sum_{k=1}^{d_{\text{in}}} p_{2k}w_{k1} + \beta_{21} & \cdots & \sum_{k=1}^{d_{\text{in}}} p_{2k}w_{kd_{\text{out}}} + \beta_{2d_{\text{out}}} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{d_{\text{in}}} p_{mk}w_{k1} + \beta_{m1} & \cdots & \sum_{k=1}^{d_{\text{in}}} p_{mk}w_{kd_{\text{out}}} + \beta_{md_{\text{out}}} \end{bmatrix}.$$

Following the bilinear attribution principle, we distribute the relevance of each output element c_{ij} to three parts: the input activation: the input p_{ik}, the weight parameter w_{kj} and the bias β_{ij}:

$$R(p_{ik}) = \frac{1}{2}\sum_{j=1}^{d_{\text{out}}} \frac{p_{ik}w_{kj}}{c_{ij}} R(c_{ij}), \quad R(w_{kj}) \quad = \frac{1}{2}\sum_{i=1}^{m} \frac{p_{ik}w_{kj}}{c_{ij}} R(c_{ij}), \quad R(\beta_{ij}) = \frac{\beta_{ij}}{c_{ij}} R(c_{ij}). \tag{14}$$

Equivalently, these element-wise relevance scores can be written in matrix form:

$$R(P) = P \odot \left[\left(\frac{R(C)}{C} \right) W^\top \right], \quad R(W) = W \odot \left[P^\top \left(\frac{R(C)}{C} \right) \right], \quad R(\mathcal{B}) = \mathcal{B} \odot \frac{R(C)}{C}. \tag{15}$$

To satisfy relevance conservation, we assign half of the bilinear relevance to $R(P)$ and $R(W)$. The final formula is as follows:

$$R(P) = \frac{1}{2}P \odot \left[\left(\frac{R(C)}{C} \right) W^\top \right], R(W) = \frac{1}{2}W \odot \left[P^\top \left(\frac{R(C)}{C} \right) \right], R(\mathcal{B}) = \mathcal{B} \odot \frac{R(C)}{C}. \tag{16}$$

$$\sum R(P) + \sum R(W) + \sum R(\mathcal{B}) = \sum R(C).$$

where $\odot$ denotes element-wise multiplication and division is applied element-wise. □

A.2 PROOF OF PROPOSITION 4.2

Proof. Let $g^{(l)} \in \mathbb{R}^{m \times d}$ be the input to the FFN module. The FFN is written as

$$U^{(l)} = g^{(l)}W_1^{(l)} + \mathcal{B}_1^{(l)}, \quad S^{(l)} = \sigma(U^{(l)}), \quad \Delta_{\text{FFN}}^{(l)} = S^{(l)}W_2^{(l)} + \mathcal{B}_2^{(l)}.$$

Here, $W_1^{(l)} \in \mathbb{R}^{d \times d_{\text{ff}}}$, $W_2^{(l)} \in \mathbb{R}^{d_{\text{ff}} \times d}$, $b_1^{(l)} \in \mathbb{R}^{1 \times d_{\text{ff}}}$, and $b_2^{(l)} \in \mathbb{R}^{1 \times d}$. The bias matrices are

$$\mathcal{B}_1^{(l)} = [b_1^{(l)}; \ldots; b_1^{(l)}] \in \mathbb{R}^{m \times d_{\text{ff}}}, \quad \mathcal{B}_2^{(l)} = [b_2^{(l)}; \ldots; b_2^{(l)}] \in \mathbb{R}^{m \times d}.$$

Given the output relevance $R(\Delta_{\mathrm{FFN}}^{(l)})$, we first apply Lemma 4.1 to the second linear function $\Delta_{\mathrm{FFN}}^{(l)} = S^{(l)}W_2^{(l)} + \mathcal{B}_2^{(l)}$:

$$R(S^{(l)}) = \frac{1}{2}S^{(l)} \odot \left[\left(\frac{R(\Delta_{\mathrm{FFN}}^{(l)})}{\Delta_{\mathrm{FFN}}^{(l)}}\right)\left(W_2^{(l)}\right)^\top\right],$$
$$R(W_2^{(l)}) = \frac{1}{2}W_2^{(l)} \odot \left[\left(S^{(l)}\right)^\top\left(\frac{R(\Delta_{\mathrm{FFN}}^{(l)})}{\Delta_{\mathrm{FFN}}^{(l)}}\right)\right],$$
$$R(b_2^{(l)}) = b_2^{(l)} \odot \mathbf{1}_m^\top\left(\frac{R(\Delta_{\mathrm{FFN}}^{(l)})}{\Delta_{\mathrm{FFN}}^{(l)}}\right).$$

where $\mathbf{1}_m \in \mathbb{R}^{m\times 1}$ is an all-one vector.

For the activation $S^{(l)} = \sigma(U^{(l)})$, we denote the propagated relevance as $R(U^{(l)})$, and we take $R(U^{(l)}) = R(S^{(l)})$. Next, applying Lemma 4.1 to $U^{(l)} = g^{(l)}W_1^{(l)} + \mathcal{B}_1^{(l)}$, we obtain

$$R(g^{(l)}) = \frac{1}{2}g^{(l)} \odot \left[\left(\frac{R(U^{(l)})}{U^{(l)}}\right)\left(W_1^{(l)}\right)^\top\right],$$
$$R(W_1^{(l)}) = \frac{1}{2}W_1^{(l)} \odot \left[\left(g^{(l)}\right)^\top\left(\frac{R(U^{(l)})}{U^{(l)}}\right)\right],$$
$$R(b_1^{(l)}) = b_1^{(l)} \odot \mathbf{1}_m^\top\left(\frac{R(U^{(l)})}{U^{(l)}}\right).$$

Therefore, the FFN module yields the following relevance scores:

$$\begin{aligned}
&R(W_2^{(l)}) = \frac{1}{2}W_2^{(l)} \odot \left[\left(S^{(l)}\right)^\top\left(\frac{R(\Delta_{\mathrm{FFN}}^{(l)})}{\Delta_{\mathrm{FFN}}^{(l)}}\right)\right], \quad R(b_2^{(l)}) = b_2^{(l)} \odot \mathbf{1}_m^\top\left(\frac{R(\Delta_{\mathrm{FFN}}^{(l)})}{\Delta_{\mathrm{FFN}}^{(l)}}\right),\\
&R(S^{(l)}) = \frac{1}{2}S^{(l)} \odot \left[\left(\frac{R(\Delta_{\mathrm{FFN}}^{(l)})}{\Delta_{\mathrm{FFN}}^{(l)}}\right)\left(W_2^{(l)}\right)^\top\right], R(g^{(l)}) = \frac{1}{2}g^{(l)} \odot \left[\left(\frac{R(S^{(l)})}{U^{(l)}}\right)\left(W_1^{(l)}\right)^\top\right],\\
&R(W_1^{(l)}) = \frac{1}{2}W_1^{(l)} \odot \left[\left(g^{(l)}\right)^\top\left(\frac{R(S^{(l)})}{U^{(l)}}\right)\right], \quad R(b_1^{(l)}) = b_1^{(l)} \odot \mathbf{1}_m^\top\left(\frac{R(S^{(l)})}{U^{(l)}}\right).
\end{aligned} \tag{17}$$

By Lemma 4.1 and the relevance conservation of the activation step, the FFN relevance satisfies

$$\sum R(g^{(l)}) + \sum R(W_1^{(l)}) + \sum R(b_1^{(l)}) + \sum R(W_2^{(l)}) + \sum R(b_2^{(l)}) = \sum R(\Delta_{\mathrm{FFN}}^{(l)}).$$

□

A.3 Proof of Proposition 4.3

Proof. Given the output relevance $R(f^{(l)})$, we first propagate relevance through the output projection $f^{(l)} = O^{(l)}W_O^{(l)}$. Let $D_f^{(l)} = R(f^{(l)})/f^{(l)}$, where the division is applied element-wise. By Lemma 4.1, we obtain

$$R(O^{(l)}) = \frac{1}{2}O^{(l)} \odot \left[D_f^{(l)}\left(W_O^{(l)}\right)^\top\right], \quad R(W_O^{(l)}) = \frac{1}{2}W_O^{(l)} \odot \left[\left(O^{(l)}\right)^\top D_f^{(l)}\right].$$

Since $O^{(l)}$ is obtained by column-wise concatenating the head outputs, we use $[\cdot]_r$ to denote the r-th column-wise block corresponding to the r-th head. The relevance assigned to the r-th head output is

$$R(O_r^{(l)}) = \left[\frac{1}{2}O^{(l)} \odot \left(\frac{R(f^{(l)})}{f^{(l)}}\left(W_O^{(l)}\right)^\top\right)\right]_r.$$

Next, for the attention-value multiplication $O_r^{(l)} = A_r^{(l)} V_r^{(l)}$, we again apply Lemma 4.1. Let $D_{O,r}^{(l)} = R(O_r^{(l)})/O_r^{(l)}$. Then

$$R(A_r^{(l)}) = \frac{1}{2} A_r^{(l)} \odot \left[D_{O,r}^{(l)} \left(V_r^{(l)} \right)^\top \right], \quad R(V_r^{(l)}) = \frac{1}{2} V_r^{(l)} \odot \left[\left(A_r^{(l)} \right)^\top D_{O,r}^{(l)} \right].$$

For the softmax operation $A_r^{(l)} = \mathrm{softmax}(E_r^{(l)})$, we use the identity relevance propagation rule: $R(E_r^{(l)}) = R(A_r^{(l)})$. Accordingly,

$$D_{E,r}^{(l)} = \frac{R(E_r^{(l)})}{E_r^{(l)}} = \frac{R(A_r^{(l)})}{E_r^{(l)}}.$$

For the attention score matrix, the relevance propagated to $Q_r^{(l)}$ and $K_r^{(l)}$ is

$$R(Q_r^{(l)}) = \frac{1}{2\sqrt{d_{\text{head}}}} Q_r^{(l)} \odot \left[D_{E,r}^{(l)} K_r^{(l)} \right], \quad R(K_r^{(l)}) = \frac{1}{2\sqrt{d_{\text{head}}}} K_r^{(l)} \odot \left[\left(D_{E,r}^{(l)} \right)^\top Q_r^{(l)} \right].$$

Let $D_{Q,r}^{(l)} = R(Q_r^{(l)})/Q_r^{(l)}$, $D_{K,r}^{(l)} = R(K_r^{(l)})/K_r^{(l)}$, and $D_{V,r}^{(l)} = R(V_r^{(l)})/V_r^{(l)}$. Applying Lemma 4.1 to the input functions gives:

$$\begin{aligned}
R(W_{Q,r}^{(l)}) &= \frac{1}{2} W_{Q,r}^{(l)} \odot \left[\left(X^{(l)} \right)^\top D_{Q,r}^{(l)} \right], \\
R(W_{K,r}^{(l)}) &= \frac{1}{2} W_{K,r}^{(l)} \odot \left[\left(X^{(l)} \right)^\top D_{K,r}^{(l)} \right], \\
R(W_{V,r}^{(l)}) &= \frac{1}{2} W_{V,r}^{(l)} \odot \left[\left(X^{(l)} \right)^\top D_{V,r}^{(l)} \right].
\end{aligned}$$

The input relevance of the MHA module is obtained by accumulating the relevance propagated through the query, key, and value branches across all heads:

$$R(X^{(l)}) = \frac{1}{2} X^{(l)} \odot \sum_{r=1}^{d/d_{\text{head}}} \left[D_{Q,r}^{(l)} \left(W_{Q,r}^{(l)} \right)^\top + D_{K,r}^{(l)} \left(W_{K,r}^{(l)} \right)^\top + D_{V,r}^{(l)} \left(W_{V,r}^{(l)} \right)^\top \right].$$

For compactness, we further define

$$\Phi_r^{(l)} = \frac{A_r^{(l)} \odot \left[\left(\frac{R(O_r^{(l)})}{O_r^{(l)}} \right) \left(V_r^{(l)} \right)^\top \right]}{E_r^{(l)}}, \tag{18}$$

where all divisions are element-wise. Substituting the intermediate relevance terms into the above propagation rules yields the following parameter relevance scores:

$$\begin{aligned}
R(W_O^{(l)}) &= \frac{1}{2} W_O^{(l)} \odot \left[\left(O^{(l)} \right)^\top \left(\frac{R(f^{(l)})}{f^{(l)}} \right) \right], \\
R(W_{Q,r}^{(l)}) &= \frac{1}{8\sqrt{d_{\text{head}}}} W_{Q,r}^{(l)} \odot \left[\left(X^{(l)} \right)^\top \left(\Phi_r^{(l)} K_r^{(l)} \right) \right], \\
R(W_{K,r}^{(l)}) &= \frac{1}{8\sqrt{d_{\text{head}}}} W_{K,r}^{(l)} \odot \left[\left(X^{(l)} \right)^\top \left(\left(\Phi_r^{(l)} \right)^\top Q_r^{(l)} \right) \right], \\
R(W_{V,r}^{(l)}) &= \frac{1}{4} W_{V,r}^{(l)} \odot \left[\left(X^{(l)} \right)^\top \left(\left(A_r^{(l)} \right)^\top \frac{R(O_r^{(l)})}{O_r^{(l)}} \right) \right].
\end{aligned} \tag{19}$$

Similarly, after substituting the intermediate relevance terms, the input relevance can be written as

$$\begin{aligned}
R(X^{(l)}) = X^{(l)} \odot \sum_{r=1}^{d/d_{\text{head}}} \Bigg[& \frac{1}{8\sqrt{d_{\text{head}}}} \left(\Phi_r^{(l)} K_r^{(l)} \right) \left(W_{Q,r}^{(l)} \right)^\top + \frac{1}{8\sqrt{d_{\text{head}}}} \left(\left(\Phi_r^{(l)} \right)^\top Q_r^{(l)} \right) \left(W_{K,r}^{(l)} \right)^\top \\
& + \frac{1}{4} \left(\left(A_r^{(l)} \right)^\top \frac{R(O_r^{(l)})}{O_r^{(l)}} \right) \left(W_{V,r}^{(l)} \right)^\top \Bigg].
\end{aligned} \tag{20}$$

□

A.4 LoRA-based MHA attribution

For LoRA-based MHA, each projection matrix is written as the sum of a frozen backbone weight and a trainable low-rank update:

$$\widetilde{W}_O^{(l)} = W_O^{(l)} + A_O^{(l)} B_O^{(l)}, \quad \widetilde{W}_{\star,r}^{(l)} = W_{\star,r}^{(l)} + A_{\star,r}^{(l)} B_{\star,r}^{(l)}, \quad \star \in \{Q, K, V\}.$$

Here, $W_O^{(l)}$ and $W_{\star,r}^{(l)}$ are frozen backbone weights, while $A_O^{(l)}, B_O^{(l)}, A_{\star,r}^{(l)}, B_{\star,r}^{(l)}$ are trainable LoRA parameters. Given $X^{(l)} = \mathrm{LN}(H^{(l)})$, the forward pass of the r-th attention head is

$$Q_r^{(l)} = X^{(l)} \widetilde{W}_{Q,r}^{(l)}, \quad K_r^{(l)} = X^{(l)} \widetilde{W}_{K,r}^{(l)}, \quad V_r^{(l)} = X^{(l)} \widetilde{W}_{V,r}^{(l)},$$
$$E_r^{(l)} = \frac{Q_r^{(l)} (K_r^{(l)})^\top}{\sqrt{d_{\mathrm{head}}}}, \quad A_r^{(l)} = \mathrm{softmax}(E_r^{(l)}), \quad O_r^{(l)} = A_r^{(l)} V_r^{(l)}.$$

The multi-head output and the final MHA output are computed as

$$O^{(l)} = \mathrm{Concat}\left(O_1^{(l)}, \ldots, O_{d/d_{\mathrm{head}}}^{(l)}\right), \quad f^{(l)} = O^{(l)} \widetilde{W}_O^{(l)}.$$

Given the output relevance $R(f^{(l)})$, let $\Lambda_f^{(l)} = \frac{R(f^{(l)})}{f^{(l)}}$ and all divisions are applied element-wise. For the output projection, the relevance scores of the frozen weight and LoRA parameters are

$$R(W_O^{(l)}) = \frac{1}{2} W_O^{(l)} \odot \left[\left(O^{(l)}\right)^\top \Lambda_f^{(l)}\right], \quad R(B_O^{(l)}) = \frac{1}{2} B_O^{(l)} \odot \left[\left(O^{(l)} A_O^{(l)}\right)^\top \Lambda_f^{(l)}\right],$$
$$R(A_O^{(l)}) = \frac{1}{4} A_O^{(l)} \odot \left[\left(O^{(l)}\right)^\top \Lambda_f^{(l)} \left(B_O^{(l)}\right)^\top\right].$$

The relevance propagated to the concatenated head output is

$$R(O^{(l)}) = \frac{1}{2} O^{(l)} \odot \left[\Lambda_f^{(l)} \left(W_O^{(l)}\right)^\top\right] + \frac{1}{4} O^{(l)} \odot \left[\left(\Lambda_f^{(l)} \left(B_O^{(l)}\right)^\top\right) \left(A_O^{(l)}\right)^\top\right].$$

Let $[\cdot]_r$ denote the r-th column-wise block corresponding to the r-th attention head, then $R(O_r^{(l)}) = \left[R(O^{(l)})\right]_r$. We further define the following intermediate relevance ratios:

$$\Lambda_{O,r}^{(l)} = \frac{R(O_r^{(l)})}{O_r^{(l)}}, \quad \Phi_r^{(l)} = \frac{A_r^{(l)} \odot \left[\Lambda_{O,r}^{(l)} \left(V_r^{(l)}\right)^\top\right]}{E_r^{(l)}}, \quad \Lambda_{Q,r}^{(l)} = \frac{1}{4\sqrt{d_{\mathrm{head}}}} \Phi_r^{(l)} K_r^{(l)},$$
$$\Lambda_{K,r}^{(l)} = \frac{1}{4\sqrt{d_{\mathrm{head}}}} \left(\Phi_r^{(l)}\right)^\top Q_r^{(l)}, \quad \Lambda_{V,r}^{(l)} = \frac{1}{2} \left(A_r^{(l)}\right)^\top \Lambda_{O,r}^{(l)}.$$

For each input projection, the relevance scores of the frozen weight and LoRA parameters are

$$R(W_{\star,r}^{(l)}) = \frac{1}{2} W_{\star,r}^{(l)} \odot \left[\left(X^{(l)}\right)^\top \Lambda_{\star,r}^{(l)}\right], \quad R(B_{\star,r}^{(l)}) = \frac{1}{2} B_{\star,r}^{(l)} \odot \left[\left(X^{(l)} A_{\star,r}^{(l)}\right)^\top \Lambda_{\star,r}^{(l)}\right],$$
$$R(A_{\star,r}^{(l)}) = \frac{1}{4} A_{\star,r}^{(l)} \odot \left[\left(X^{(l)}\right)^\top \Lambda_{\star,r}^{(l)} \left(B_{\star,r}^{(l)}\right)^\top\right], \quad \star \in \{Q, K, V\}.$$

The relevance propagated to the MHA input is obtained by aggregating the frozen-weight and LoRA-branch contributions across all heads:

$$R(X^{(l)}) = X^{(l)} \odot \sum_{r=1}^{d/d_{\mathrm{head}}} \sum_{\star \in \{Q,K,V\}} \left[\frac{1}{2} \Lambda_{\star,r}^{(l)} \left(W_{\star,r}^{(l)}\right)^\top + \frac{1}{4} \left(\Lambda_{\star,r}^{(l)} \left(B_{\star,r}^{(l)}\right)^\top\right) \left(A_{\star,r}^{(l)}\right)^\top\right].$$

Thus, the relevance scores of frozen backbone parameters in the LoRA-based MHA module are

$$R(\mathcal{W}^{(l)}_{\text{MHA}}) = \Big\{R(W^{(l)}_O), \{R(W^{(l)}_{Q,r}), R(W^{(l)}_{K,r}), R(W^{(l)}_{V,r})\}_r\Big\}.$$

The relevance scores of trainable LoRA parameters are

$$R(\mathcal{W}^{(l)}_{\text{LoRA}}) = \Big\{R(A^{(l)}_O), R(B^{(l)}_O), \{R(A^{(l)}_{Q,r}), R(B^{(l)}_{Q,r}), R(A^{(l)}_{K,r}), R(B^{(l)}_{K,r}), R(A^{(l)}_{V,r}), R(B^{(l)}_{V,r})\}_r\Big\}.$$

Together, $R(\mathcal{W}^{(l)}_{\text{MHA}})$, $R(\mathcal{W}^{(l)}_{\text{LoRA}})$, and $R(X^{(l)})$ quantify the importance of frozen backbone parameters, trainable LoRA parameters, and the MHA input, respectively.

A.5 LRP Attribution in LoRA-based Transformer Block

We compose the FFN and MHA propagation rules to obtain block-wise parameter attribution. $H^{(l+1)} = H^{(l)} + f^{(l)} + FFN(g^{(l)})$. Given $R(H^{(l+1)})$, we first decompose the output relevance across the three additive branches, yielding relevance for the input $H^{(l)}$, the MHA output $f^{(l)}$ and the FFN output. Proposition 4.2 propagates the FFN relevance to its parameters and input $g^{(l)}$. We then pass $R(g^{(l)})$ backward through the LayerNorm without redistribution and accumulate the resulting relevance with the relevance already assigned to $f^{(l)}$ by the residual decomposition. The accumulated relevance of $f^{(l)}$ is subsequently propagated through the MHA module using Proposition A.4, yielding relevance scores for the MHA parameters and its input. Finally, the input relevance from the MHA branch is combined with the residual relevance of $H^{(l)}$, producing the total relevance propagated to the preceding Transformer block. We denote the parameter set by $\mathcal{W}^{(l)} = \Big\{W^{(l)}_O, \{W^{(l)}_{P,r} : P \in \{Q,K,V\}, r = 1, \cdots, n_h\}, W^{(l)}_1, W^{(l)}_2, b^{(l)}_1, b^{(l)}_2\Big\}$. The Lora parameter set is $R(\mathcal{W}^{(l)}_{\text{LoRA}}) = \Big\{R(A^{(l)}_O), R(B^{(l)}_O), \{A^{(l)}_{P,r}, B^{(l)}_{P,r} : P \in \{Q,K,V\}, r = 1, \cdots, n_h\}\Big\}$. The block-wise propagation can be summarized as $R(H^{(l+1)}) \rightarrow \big\{R(H^{(l)}), R(\mathcal{W}^{(l)}), R(\mathcal{W}^{(l)}_{\text{LoRA}})\big\}$. Since only LoRA parameters are updated during continual fine-tuning, we use $R(\mathcal{W}^{(l)}_{\text{LoRA}})$ as the task-specific importance prior for guiding parameter updates.

A.6 The overall framework

The over framework of our method is shown in Algorithm 1.

A.7 Datasets

TRACE: we further evaluate our method on seven tasks from TRACE Wang et al. (2023). The selected tasks cover financial text understanding, long-context summarization, code completion, scientific question answering, mathematical reasoning, and multilingual text generation.

- **FOMC**: A monetary-policy stance classification task. The LLMs read a message and predicts whether the stance is dovish, hawkish or neutral.
- **MeetingBank**: A long-context meeting summarization task. The input is a meeting transpcript, and the LLMs generates a concise summary of the discussion and decisions.
- **PY150**: A python code completion task. Given a partial python program as input, the LLMs is required to generate the subsequent code.
- **ScienceQA**: A science question-answering task with explanation oriented answers. It requires the model to answer science-related questions, and provides supporting reasoning.
- **NumGLUE-cm**: A mathematical reasoning task. It focuses on arithmetic reasoning in commonsense contexts, requiring the model to solve math problems expressed in natural language.
- **NumGLUE-ds** is another mathematical reasoning task from NumGLUE. Compared with NumGLUE-cm, it requires arithmetic reasoning grounded in more domain-specific or grade-school scientific knowledge. The LLMs infer the correct numeric result from the problem statement.
- **20Minuten**: A German text simplification task based on news articles. The LLMs is required to generate a simplified version of the input paragraph in German.

Algorithm 1 Attribution-guided continual fine-tuning

Input: Pre-trained LLM parameters θ_0, sequential tasks $\{(\mathcal{T}_t, \mathcal{D}_t)\}_{t=1}^{T}$, parameter sets $\{\mathcal{W}^{(l)}\}_{l=1}^{L}$

Stage I: Task-level attribution

for $t = 1, \ldots, T$ **do**
  Fine-tune a task-specific model θ_t' on $\mathcal{D}_t$ and collect high quality generated samples $\mathcal{C}_t \subseteq \mathcal{D}_t$
  for all $(x^{(k)}, y^{(k)}) \in \mathcal{C}_t$ **do**
    Obtain the target logit $Z_{m,\hat{j}}$ from θ_t' and initialize $R^{(k)}(H^L) \leftarrow Z_{m,\hat{j}}$
    for all $l = L-1, \ldots, 1$ and $W^{(l,t)} \in \mathcal{W}^{(l)}$ **do**
      Propagate relevance to $W^{(l,t)}$ following Sec. 4.1.2 for SFT and App. A.5 for LoRA
      Normalize parameter relevance $\widehat{R}^{(k)}(W^{(l,t)})$ using Eq. (10)
    end for
  end for
  for all $l = 1, \ldots, L$, $W^{(l,t)} \in \mathcal{W}^{(l)}$, and element u of $W^{(l,t)}$ **do**
    Compute task-level importance $R(W^{(l,t)})[u]$ using Eq. (11)
  end for
  Store the task-level prior $\mathcal{R}_t = \{R(W^{(l,t)})\}_{l=1}^{L}$
end for

Stage II: Attribution-guided continual fine-tuning

Initialize $\mathcal{R}_0 \leftarrow \mathbf{0}$ for all parameter sets, and set $\theta_1 \leftarrow \theta_0$
for $t = 1, \ldots, T$ **do**
  Compute the task loss $\mathcal{L}_t$ with continual model parameters θ_t
  for all $l = L-1, \ldots, 1$ and $W^{(l,t)} \in \mathcal{W}^{(l)}$ **do**
    Aggregate historical importance $\bar{R}(W^{(l,t)})$ using Eq. (12)
    Compute the raw gradient $\nabla_{W^{(l,t)}} \mathcal{L}_t$
    Modulate the gradient: $\widetilde{\nabla}_{W^{(l,t)}} \mathcal{L}_t = \left(\mathbf{1} - \bar{R}(W^{(l,t)})\right) \odot \nabla_{W^{(l,t)}} \mathcal{L}_t$
    Update $W^{(l,t)}$ using the modulated gradient
  end for
  Set $\theta_{t+1} \leftarrow \theta_t$
end for
Output: model parameters θ_T.

Table 3: The description and evaluation metrics of the TRACE.

Task	Dataset	Task Type	Evaluation Metric
Task 1	FOMC	Monetary policy stance classification	Accuracy
Task 2	MeetingBank	Meeting summarization	ROUGE-L
Task 3	Py150	Code completion	Edit Similarity
Task 4	ScienceQA	Science question answering	Accuracy
Task 5	NumGLUE-cm	Numerical reasoning	Accuracy
Task 6	NumGLUE-ds	Domain-specific numerical reasoning	Accuracy
Task 7	20Minuten	German text simplification	SARI

Table 4: Summary of experimental settings.

Backbone	Benchmark	#Tasks	Tuning	Task Type
Llama-3.2-3B-Instruct	Concept-10	10	Full fine-tuning	Short-answer QA
Llama-3.2-3B-Instruct	TRACE	7	Full fine-tuning	QA, Generation, Code, Math
Llama2-7b	Concept-10	10	LoRA	Short-answer QA
Llama2-7b	TRACE	7	LoRA	QA, Generation, Code, Math
Qwen2.5-3b	Concept-10	10	Full fine-tuning	Short-answer QA
Qwen2.5-3b	TRACE	7	Full fine-tuning	QA, Generation, Code, Math
Qwen2.5-7b	Concept-10	10	LoRA	Short-answer QA
Qwen2.5-7b	TRACE	7	LoRA	QA, Generation, Code, Math

A.8 Baseline Details

We consider seven baselines representing different strategies for mitigating catastrophic forgetting.

- **Sequential fine-tuning (SEQ).** Sequentially fine-tunes the model on each task without any explicit mechanism for preserving knowledge from previous tasks.
- **Replay-based methods (Replay, DER++). Replay** jointly trains on current-task data and buffered examples from previous tasks. **DER++** (Buzzega et al., 2020) further incorporates prediction consistency on replayed samples to preserve previous model behavior. In our experiments, we set the replay ratio to 0.1.
- **Parameter-importance-based methods (EWC, SNIP, Wanda). EWC** (Kirkpatrick et al., 2017) uses the diagonal Fisher information matrix to estimate parameter importance and penalize changes to important parameters. **SNIP** (Lee et al., 2019) uses connection sensitivity to measure the effect of individual parameters on the task loss and we adapt its criterion from network pruning to parameter selection for continual LLM fine-tuning. **Wanda** (Sun et al., 2024) combines weight magnitude with the norm of the corresponding input activations, and we use its score as a parameter-importance measure in LLMs.
- **Parameter freezing (Freeze). Freeze** (Zheng et al., 2025) freezes the bottom Transformer layers and updates only the remaining layers during continual fine-tuning.

A.9 Continual learning setting

Table 4 summarizes our continual learning settings. We use AdamW with a learning rate of 1×10^{-5} and weight decay of 5×10^{-4} for all experiments. Each task is trained for 10 epochs with a batch size of 8 on Concept-1K, and for 5 epochs with a batch size of 4 on TRACE. We use a fixed random seed of 42 throughout. All experiments are run on four NVIDIA RTX A6000 GPUs with 48 GB memory each. Training all models requires approximately 6 hours on TRACE and 2 hours on Concept-1K.

A.10 Evaluation metrics

We evaluate performance using four metrics: Accuracy, ROUGE-L, Edit Similarity, and SARI. Accuracy is based on exact match with the ground-truth answer. ROUGE-L evaluates summarization quality via the longest common subsequence between generated and reference summaries, while

Edit Similarity measures string-level similarity for code completion. SARI is used for text simplification to assess the quality of added, deleted, and retained content relative to the source and reference texts. The metric used for each task is reported in Table 3.

A.11 Overlap of Important Parameter Locations of Llama-3.2-Instruct-3B.

We further find that this task-dependent overlap pattern extends beyond the attention output projection to other parameter matrices in Llama-3.2-Instruct-3B, including the FFN up (Figure 4), down (Figure 5), and gate (Figure 6) projections, as well as the attention query (Figure 7), key (Figure 8), and value (Figure 9) projections. On Concept-1K, overlap is generally higher in later layers and typically peaks in the final layer. On TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) consistently exhibit higher overlap in deeper layers, whereas dissimilar tasks maintain lower overlap. These consistent trends across parameter matrices suggest that the observed pattern is not specific to a particular parameter.

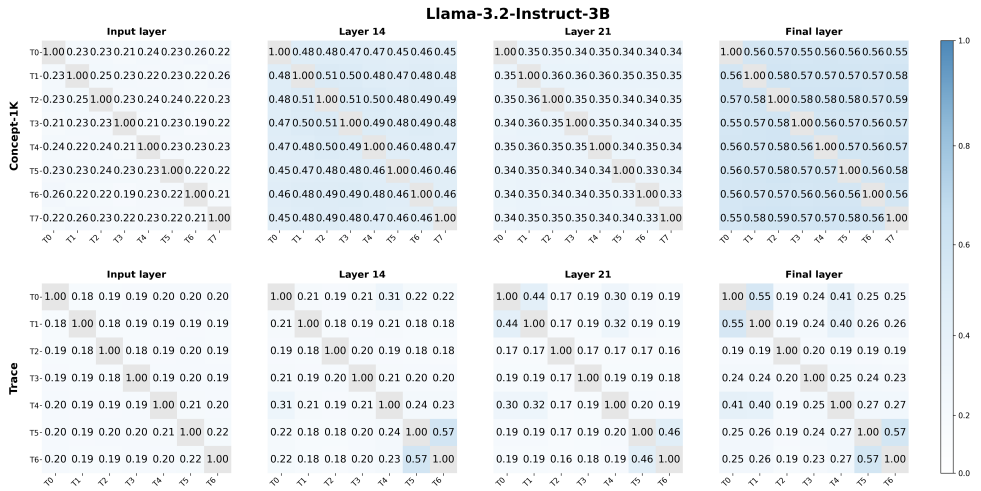

Figure 4: Overlap of important parameter locations in the feed-forward network **up** projection of Llama-3.2-Instruct-3B. We report pairwise overlap across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap varies across intermediate layers but becomes substantially higher in the final layer, while related TRACE tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks.

A.12 Overlap of Important Parameter Locations of Qwen2.5-3B

We observe a consistent task-dependent overlap pattern across the FFN up (Figure 11), down (Figure 12), and gate (Figure 13) projections and the attention query (Figure 14), key (Figure 15), and value (Figure 16) projections of Qwen2.5-3B. On Concept-1K, where tasks are closely related, important-parameter locations generally exhibit high overlap, although the layer-wise trend varies across projections: the FFN and value projections show relatively high overlap in earlier and intermediate layers, whereas the query and key projections exhibit particularly high overlap in deeper layers. On TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) consistently show higher overlap than dissimilar task pairs, with this distinction becoming especially pronounced in deeper layers. These results suggest that the task-dependent overlap pattern is broadly consistent across different parameter matrices and follows trends similar to those observed in Llama-3.2-Instruct-3B.

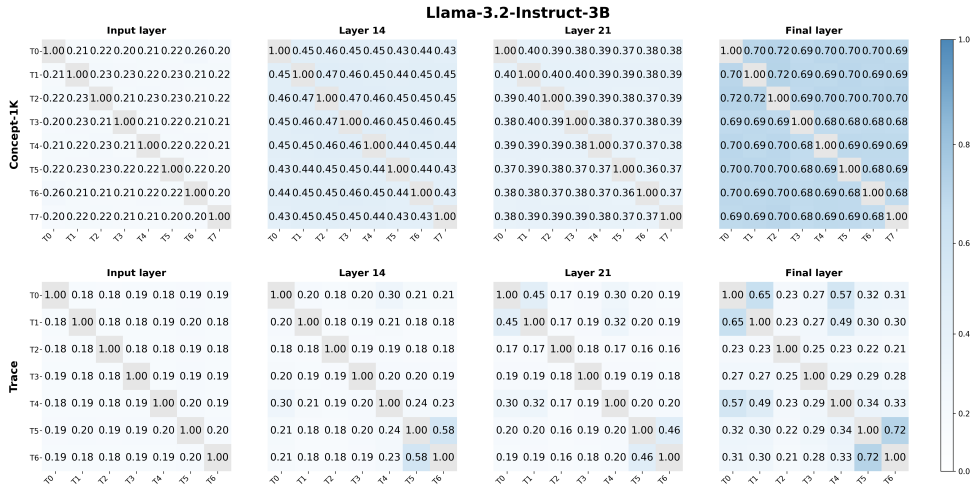

Figure 5: Overlap of important parameter locations in the feed-forward network **down** projection of Llama-3.2-Instruct-3B. We report pairwise overlap across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap varies across intermediate layers but becomes substantially higher in the final layer, while related TRACE tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks.

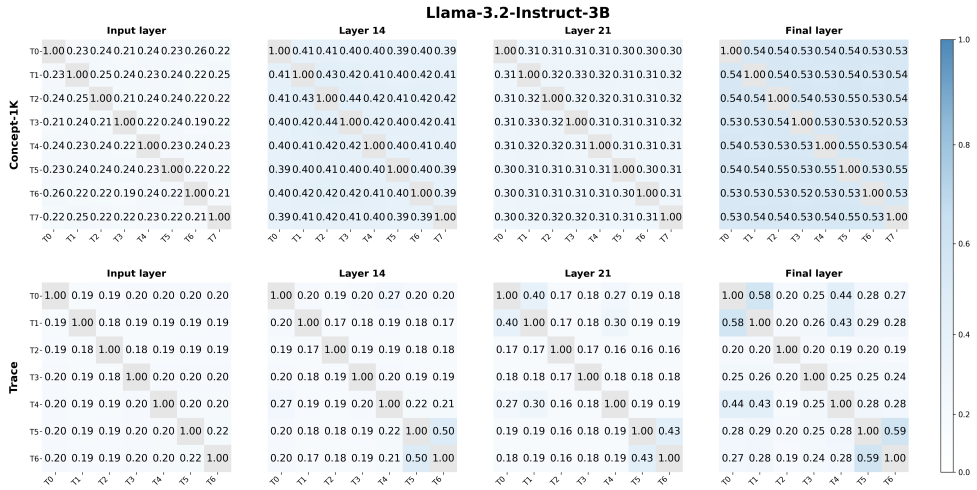

Figure 6: Overlap of important parameter locations in the feed-forward network **gate** projection of Llama-3.2-Instruct-3B. We report pairwise overlap across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap varies across intermediate layers but becomes substantially higher in the final layer, while related TRACE tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks.

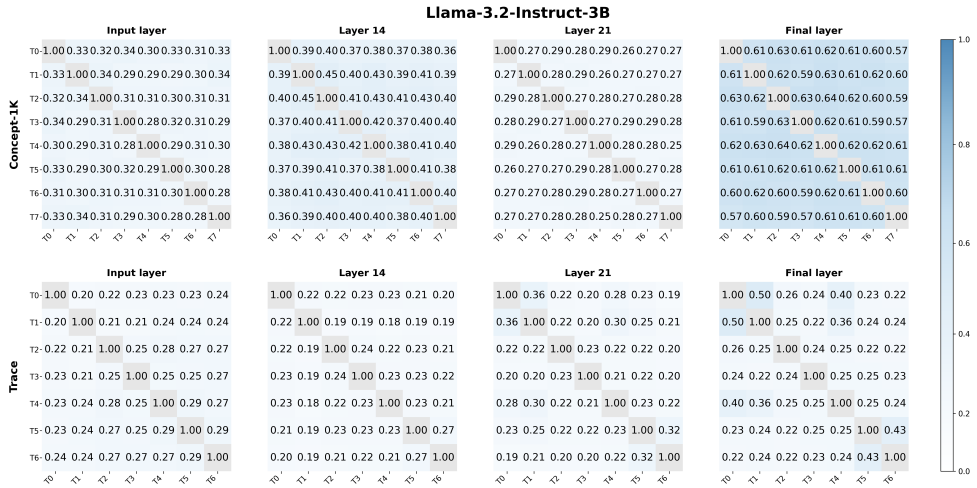

Figure 7: Overlap of important parameter locations in the attention **query** projection of Llama-3.2-Instruct-3B. We report pairwise overlap across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap varies across intermediate layers but becomes substantially higher in the final layer, while related TRACE tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks.

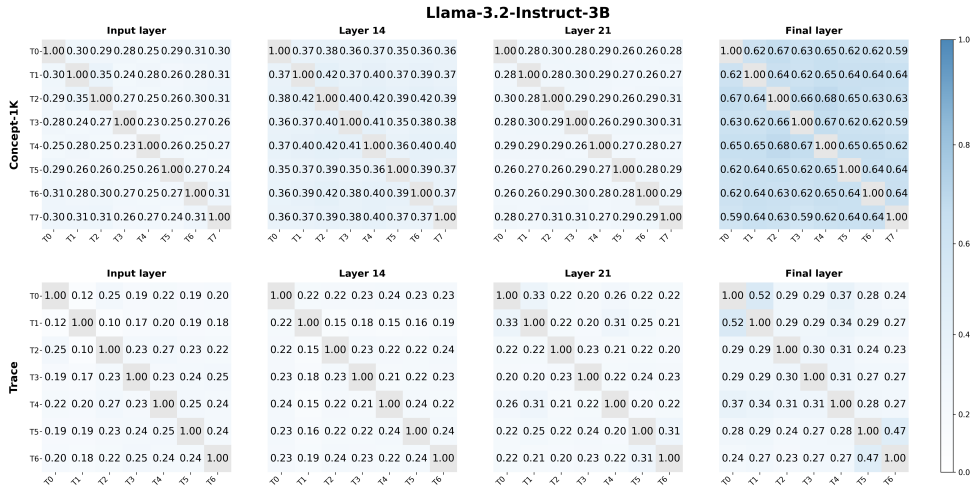

Figure 8: Overlap of important parameter locations in the attention **key** projection of Llama-3.2-Instruct-3B. We report pairwise overlap across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap varies across intermediate layers but becomes substantially higher in the final layer, while related TRACE tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks.

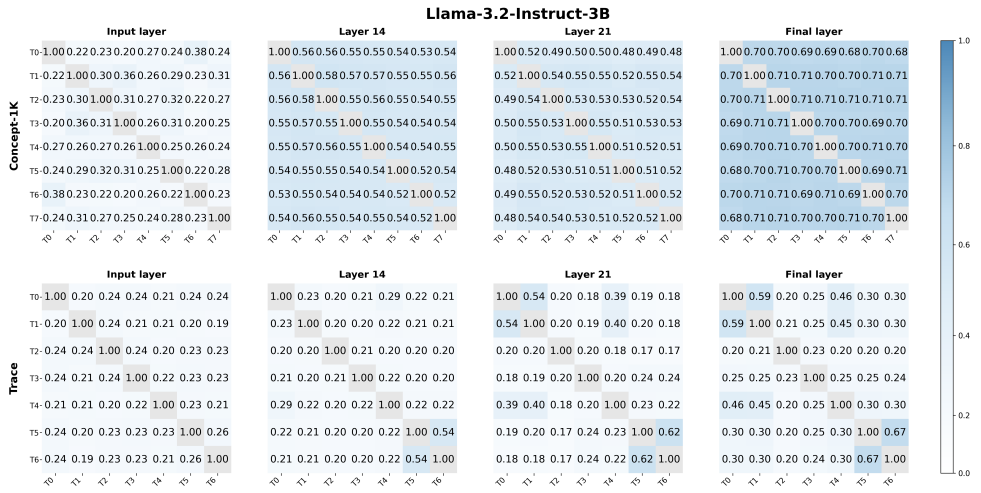

Figure 9: Overlap of important parameter locations in the attention **value** projection of Llama-3.2-Instruct-3B. We report pairwise overlap across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap varies across intermediate layers but becomes substantially higher in the final layer, while related TRACE tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks.

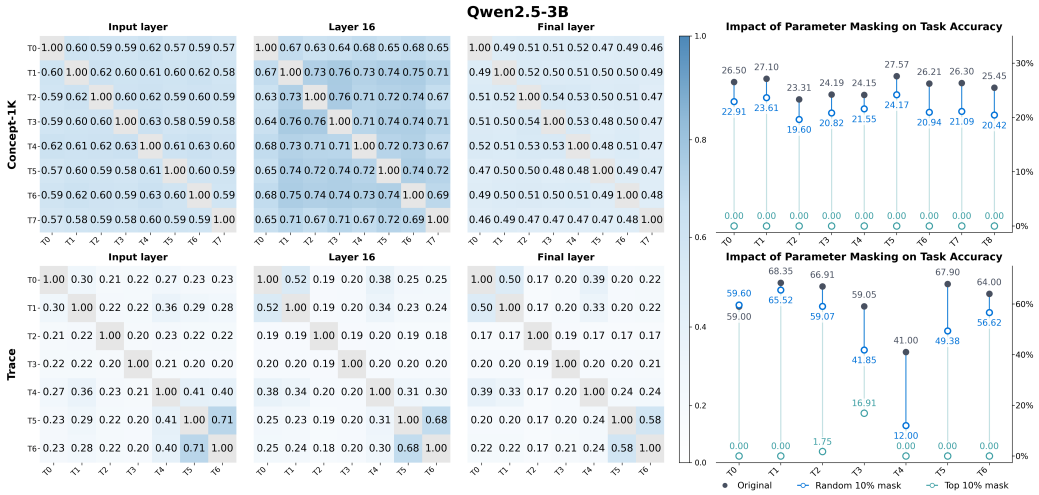

Figure 10: Overlap of important parameter locations in the attention output projection of Qwen2.5-3B. We report pairwise overlaps across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. Concept-1K shows consistently high overlap across layers, while on TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) exhibit higher overlap in deeper layers than dissimilar tasks. Right: Black denotes independently fine-tuned models, blue random masking of 10% of parameters, and cyan masking of the top-10% important parameters. Masking important parameters causes a larger performance drop than random masking.

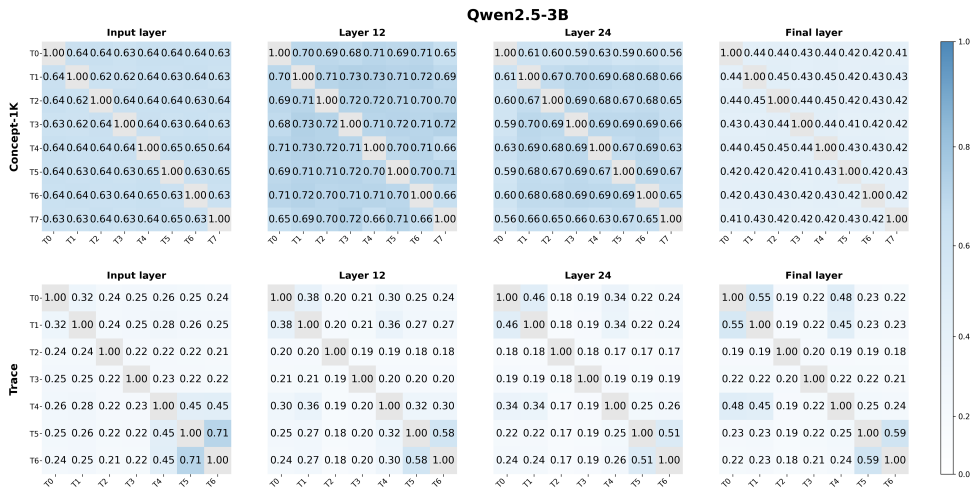

Figure 11: Overlap of important parameter locations in the feed-forward network **up** projection of Qwen2.5-3B. We report pairwise overlaps across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap remains high across layers but decreases in the final layer. On TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks in deeper layers.

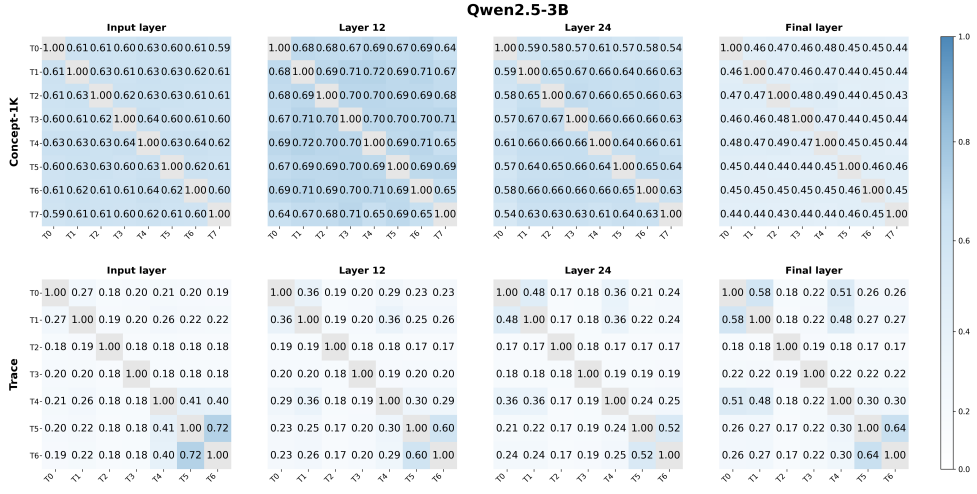

Figure 12: Overlap of important parameter locations in the feed-forward network **down** projection of Qwen2.5-3B. We report pairwise overlaps across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap remains high across layers but decreases in the final layer. On TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks in deeper layers.

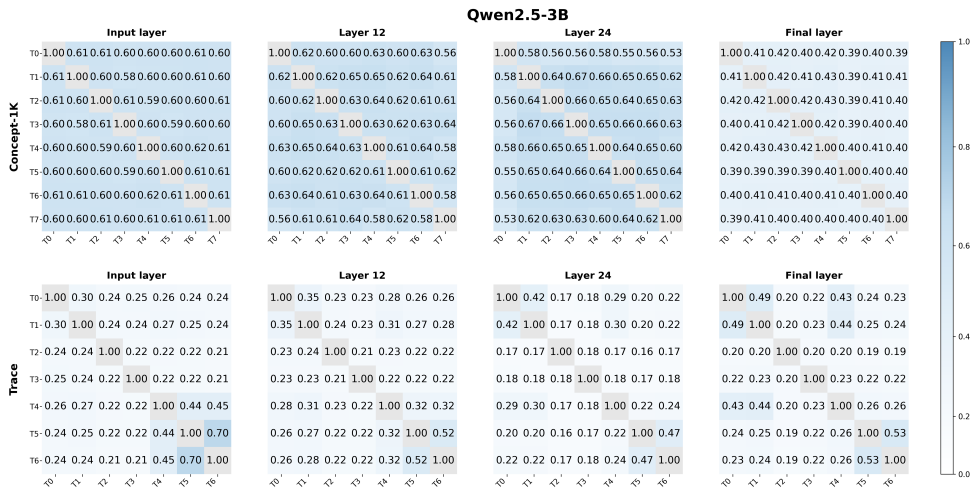

Figure 13: Overlap of important parameter locations in the feed-forward network **gate** projection of Qwen2.5-3B. We report pairwise overlaps across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap remains high across layers but decreases in the final layer. On TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) show higher overlap than dissimilar tasks in deeper layers.

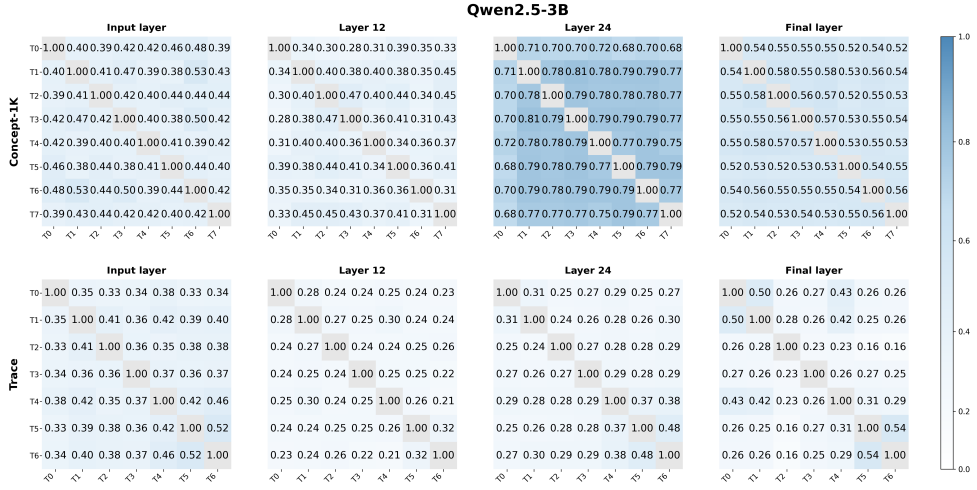

Figure 14: Overlap of important parameter locations in the attention query projection of Qwen2.5-3B. We report pairwise overlaps across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap is highest in the deeper layers, particularly around Layer 24, and remains relatively high in the final layer. On TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) exhibit higher overlap than dissimilar tasks, especially in deeper layers.

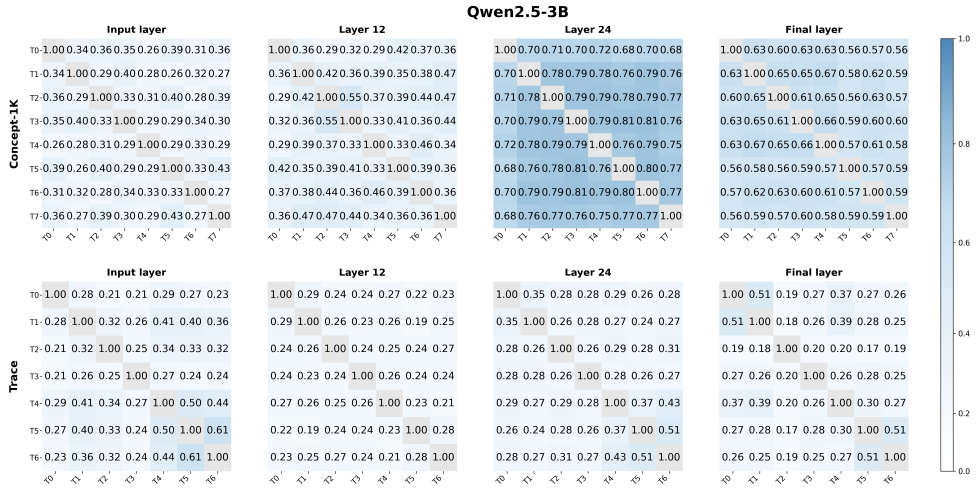

Figure 15: Overlap of important parameter locations in the attention key projection of Qwen2.5-3B. We report pairwise overlaps across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap is highest in the deeper layers, particularly around Layer 24, and remains relatively high in the final layer. On TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) exhibit higher overlap than dissimilar tasks, especially in deeper layers.

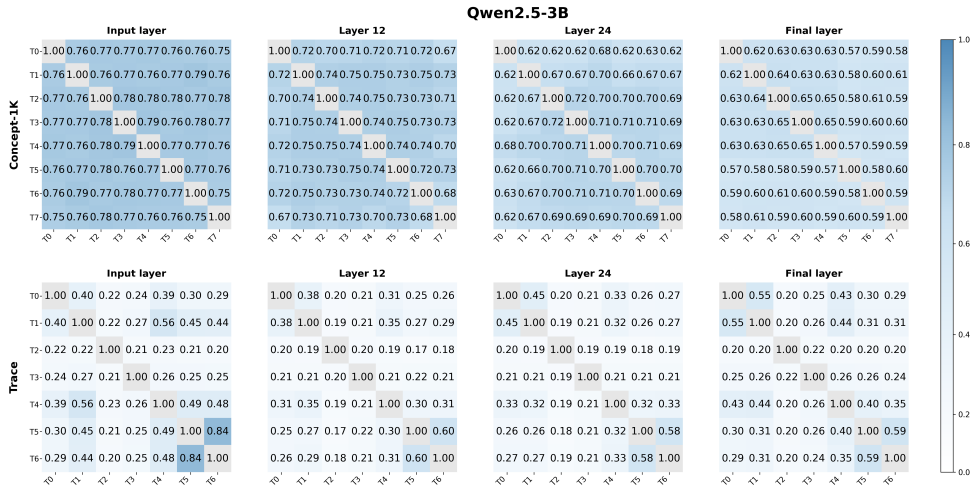

Figure 16: Overlap of important parameter locations in the attention value projection of Qwen2.5-3B. We report pairwise overlaps across layers on Concept-1K (top) and TRACE (bottom), using the top-10% important parameters. On Concept-1K, overlap remains consistently high across layers and is slightly higher in earlier layers. On TRACE, related tasks (e.g., Tasks 0, 1 and 5, 6) exhibit higher overlap than dissimilar tasks, particularly in the final layer.