Learning to Route for Dynamic Adapter Composition in Continual Learning with Language Models

Vladimir Araujo, Marie-Francine Moens, Tinne Tuytelaars
KU Leuven
vladimir.araujo@kuleuven.be

Abstract

Parameter-efficient fine-tuning (PEFT) methods are increasingly used with pre-trained language models (PLMs) for continual learning (CL). These methods typically involve training a PEFT module for each new task and employing similarity-based selection to route modules during inference. However, they face two major limitations: 1) interference during module training with already learned modules and 2) suboptimal routing when composing modules. In this paper, we present L2R, a method that isolates the training of new PEFT modules to ensure their task specialization. L2R then learns to compose the learned modules by training a network of routers that leverages a small memory containing examples of previously seen tasks. We evaluate our method in two CL setups using various benchmarks. Our results demonstrate that L2R provides an effective composition of PEFT modules, leading to improved generalization and performance compared to other methods.

1 Introduction

CL aims to continuously learn new tasks without forgetting previously learned knowledge from old tasks (McCloskey and Cohen, 1989). Recent advancements in PLMs and PEFT methods have shown potential in CL for NLP (Zhou et al., 2024).

PEFT methods, like prompts or adapters, are lightweight modules that can be trained for downstream tasks while keeping the PLM frozen. Recent work shows that these methods can be competitive with, or even superior to, full fine-tuning of PLMs (Li and Liang, 2021; Hu et al., 2022). Additionally, research (Wang et al., 2023, 2024, 2022b; Smith et al., 2023) demonstrates their effectiveness in CL.

However, PEFT approaches for CL have two significant limitations: 1) When a new task arises, a new PEFT module is added, and the previously learned ones remain frozen but activated (Razdaibiedina et al., 2023). Previous modules may interfere with optimal task knowledge acquisition by the current PEFT module. 2) They rely on a similarity-based selection mechanism (Wang et al., 2024) that may not accurately represent a routing function that effectively composes the PEFT modules.

To address these issues, we introduce **L**earning **to Ro**ute for dynamic PEFT composition (L2R). L2R is a simple but effective method that trains PEFT modules in isolation to avoid interference from previous knowledge when learning new tasks (Wang et al., 2023). Then, it uses previous examples stored in a memory to learn a routing function for dynamically composing PEFT modules during inference, a process analogous to local adaptation (d'Autume et al., 2019). Our extensive evaluations show that L2R is competitive against other PEFT-based methods across benchmarks and CL setups.

Unlike existing methods that focus on developing routing mechanisms within the training phase, our method learns to route adapters prior to test-time inference. From a practitioner's perspective, the ultimate goal of CL is to develop a model that can learn new tasks while maintaining high performance (Prabhu et al., 2020). Our method addresses this by incrementally adding PEFT modules for new tasks. Additionally, in real-world applications, systems are primarily constrained by computational and time budgets rather than storage (Prabhu et al., 2023; Verwimp et al., 2024). By leveraging a memory along PEFT, our approach achieves a better trade-off between performance and efficiency.

2 Related Work

Continual Learning with PLMs CL in NLP has grown with the advent of robust PLMs from pre-training, beneficial for CL (Zhou et al., 2024). Existing NLP approaches include replay-based (d'Autume et al., 2019; Araujo et al., 2022b), meta-learning-based (Wang et al., 2020), amount others.

Recent techniques use PEFT for CL, adding new

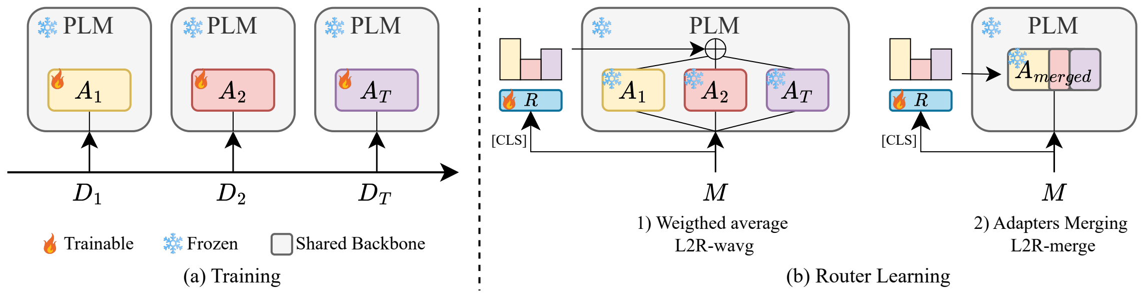

Figure 1: Overview of the L2R method. (a) Adapters A attached to the backbone are sequentially trained on a series of tasks D. Each adapter undergoes isolated training to prevent interference. (b) Before performing inference, our method utilizes a memory M to learn a routing function R, facilitating composition either by 1) computing a weighted average of the adapters' outputs or 2) merging the parameters of the adapters.

modules as tasks arise while freezing previous ones to maintain a robust backbone and leverage past knowledge (Razdaibiedina et al., 2023). This approach has been extended to learn the composition of PEFT modules based on similarity (Wang et al., 2024). While effective, these methods may suffer from interference during learning new modules and suboptimal module composition during inference.

Our work focuses on the PEFT setup. We rely on the parameter isolation strategy (Wang et al., 2023; Cheng et al., 2024) to let the PLM learn task-specific modules independently. We also adopt a memory as in replay-based methods, not to rehearse examples but to learn a routing function for effective module composition before inference.

Local Adaptation Memory-based parameter adaptation (MbPA) (Sprechmann et al., 2018), also known as local adaptation, allows a model to deal with changes and shifts in data distributions. In CL, it leverages labeled data stored in memory for brief fine-tuning before making predictions to improve model performance (d'Autume et al., 2019).

Our work builds on the concept of local adaptation to maintain model generalization and performance across tasks in CL. Rather than retraining model components, we introduce a new component (a router) that learns to dynamically compose appropriate PEFT modules based on input.

Learning to Route In mixture-of-experts (MoE) models, routing activates subnetworks based on their specialized capabilities for prediction (Shazeer et al., 2017a). Recently, adapters—a class of PEFT methods—have been employed to implement MoE-style models (Wang et al., 2022a) and routing functions (Ponti et al., 2023).

Existing approaches in CL use task centroids (Cheng et al., 2024), learnable task vectors (Wang et al., 2024), or record task distributions (Wang et al., 2023) to select modules. However, these act as a proxy for routing and may not effectively direct input to modules based on their specialties.

Our work builds on studies showing the benefits of a well-learned router in MoE models (Dikkala et al., 2023). We propose a router learning approach that takes place prior to inference and enables dynamic utilization of modules by the model.

3 Method

We consider a CL setup for NLP (d'Autume et al., 2019), where a model needs to learn from a sequence of T tasks: $D = \{D_1, ..., D_T\}$. Each task t consists of a new data distribution $D_t = (X_t, Y_t)$, where X_t are the input instances and Y_t are the labels. A model in this setup consists of a PLM that processes input x and a classifier for predicting class y. We extend this setup with PEFT modules, specifically adapters, which consistently achieve superior predictive performance than other techniques in CL (Wistuba et al., 2024).

Task-specific Adapters L2R employs a set of adapters $A = \{A_1, ..., A_T\}$ that learn task-specific knowledge from a data stream (Figure 1a). Specifically, an adapter A_t is activated and trained when a new task D_t emerges. Previous adapters are deactivated, allowing A_t to learn in isolation and specialize in task t (Wang et al., 2023). We keep the backbone frozen and train only the adapter A_t on the current task t. To simplify the explanation, we omit the fact that each task uses L adapters that match the number of PLM layers.

Adapters offer significant benefits (Wistuba et al., 2024). Their robust modeling capabilities allow them to effectively capture distribution-specific details from each task and the high parameter efficiency enables efficient training of our method whenever a new task emerges.

Memory Our method is equipped with a non-parametric memory $M = \{(x_i, y_i)\}$ that stores input-output pairs of previously seen training examples. A similar component is used in replay methods (d'Autume et al., 2019), where memory is used to perform replay during training. However, our memory is leveraged prior to test inference to learn a routing function resembling the local adaptation process to improve generalization.

We determine the memory capacity as a percentage of the total dataset size and populate it by randomly sampling examples from the training data $M = \{\mathtt{sample}(D_1), ..., \mathtt{sample}(D_T)\}$. Random sampling effectively captures the global distribution of the data stream and has proven successful in experience replay (Araujo et al., 2022a), so we hypothesize that it will similarly support router training. While more advanced memory population methods could enhance router performance (Hurtado et al., 2023; Hayes and Kanan, 2021), we leave this exploration for future work.

Memory-based Router Learning Once the adapters A have been trained, they can be combined for inference (Figure 1b). For this, we implement a router network R within the model. Note that, for simplicity, we omit the detail that a separate R is added for each of the L layers. This router computes the probability of the input being sent to each adapter. This provides two advantages: 1) the routers learn to route based on input structure, and 2) can produce different combinations at different layers based on the model's abstraction hierarchy.

In practice, the router takes the [CLS] representation as input to generate an allocation vector $z = R(h_{[\text{CLS}]})$, which is then used to compute the output distribution. Rather than applying Softmax across adapters, where modules compete for activation, we adopt an approach inspired by Ponti et al. (2023), emphasizing task-level modularity that respects task hierarchy, with more complex tasks encompassing simpler ones as sub-tasks. In this case, the router would output a binary scalar for each adapter, indicating whether it is active for a given input. As a binary vector is not differentiable, this is implemented as a collection of Bernoulli distributions, relaxed through a Gumbel-sigmoid to ensure stochasticity: $\hat{z}_t = \sigma \log \left[\frac{\sigma(z_t)u}{(1-\sigma(z_t))(1-u)}\right]$, where $u \sim \mathtt{Uniform}(0, 1)$.

To obtain a competent router, we propose a memory-based router learning process. Inspired by local adaptation, we leverage the elements in memory to train the router network parameters. However, unlike local adaptation, which adjusts the model for every inference example, our method performs this process only once after the training phase to learn the routing functions, enabling inference on any example without further adaptation.

Adapter Composition Given routing probabilities, the model can compose its adapters in different ways. We consider two options: 1) a weighted average of their outputs (Shazeer et al., 2017b), denoted as L2R-wavg, and 2) merging adapter parameters via a weighted average of their weights (Wortsman et al., 2022), referred to as L2R-merge.

Note that these approaches differ slightly. The first uses all adapters for inference and returns a weighted average of the hidden state outputs: $H_{wavg} = \sum_{t=1}^{T} \hat{z}_t * H_t$, where $H_t = A_t(x)$, while the second merges all adapters into a single one to be used to produce a unique hidden state output: $A_{merge} = \sum_{t=1}^{T} \hat{z}_t * A_t$.

4 Experimental Setup

Benchmarks We adopt two CL setups: 1) Class-Incremental Learning (CIL) and 2) Task-Incremental Learning (TIL). Both setups aim to incrementally learn new tasks (and thus new classes), but TIL always has access to task identity, making CIL a more challenging and realistic scenario.

Based on this, we consider three benchmarks. *MTL5* (d'Autume et al., 2019): 5 text classification tasks. *WOS* (Kowsari et al., 2017): 7 document classification tasks. *AfriSenti* (Muhammad et al., 2023): 12 (multilingual) sentiment analysis tasks. More details and training orders are in Appendix A.

Baselines We focus our experimentation on comparing PEFT-based CL methods. *Lower-Bound* trains task-specific adapters, summing their outputs during inference. *Upper-Bound* trains task-specific adapters, using only the corresponding adapter for inference. *ProgPrompt* (Razdaibiedina et al., 2023) progressively adds new prompts for new tasks. All prompts are used for inference (only for TIL). *DAM* (Cheng et al., 2024) learns task-specific adapters and creates task vectors by averaging training example representations. At inference, it uses similarity scores between the input and task vectors to route (only for CIL). *EPI* (Wang et al., 2023) learns

Setup	Method	MTL5	WOS	AfriSenti
	Lower-Bound	16.2	15.07	33.64
	Upper-Bound	79.4	90.06	62.16
	DAM	27.4	10.00	48.78
	EPI†	77.3	77.83	43.10
CIL	MoCL†	73.8	79.23	45.61
	L2R-wavg	**78.0**	79.90	**60.04**
	L2R-merge	77.7	**79.98**	52.82
	ProgPrompt†	77.9	89.93	49.07
	EPI†	77.3	77.83	43.10
TIL	MoCL†	**79.4**	**90.59**	56.77
	L2R-wavg	**79.4**	89.24	**62.62**
	L2R-merge	79.3	89.16	53.97

Table 1: Accuracy results on MTL5, WOS, and AfriSenti benchmarks for (top) CIL and (bottom) TIL setups. We present the average performance across all orders. † indicates results from (Wang et al., 2024).

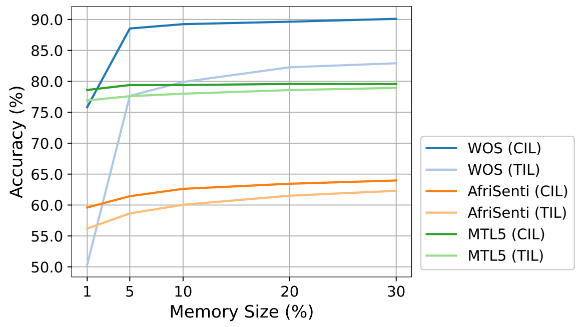

Figure 2: L2R-wavg performance across tasks and CL setups. Results for order 3 are shown for MTL5 and AfriSenti, and for order 1 for WOS.

task-specific prompts and records task distributions. At inference, it selects the nearest task's prompt based on input comparison with all distributions. *MoCL* (Wang et al., 2024) progressively adds new prompts and learns a task vector. At inference, it uses task vectors to compute similarity scores to combine the prompts.

Implementation Details We use BERT (Devlin et al., 2019) for MTL5 and WOS while AfroXLMR (Alabi et al., 2022) for AfriSenti. To implement our adapters, We adopt LoRA (Hu et al., 2022) due to its efficiency and performance. Our memory is similar to that of (d'Autume et al., 2019), but we accumulate only 10% of each task's, training dataset a negligible amount when using efficient storage formats. Our router network consists of a linear transformation followed by a Gumbel-sigmoid (Maddison et al., 2017). In CIL, routers are trained using all memory examples to route inputs universally across tasks. In TIL, leveraging task identity, routers are trained with task-specific memory elements. For more details, see Appendix B.

5 Results

In this section, we present our results, summarized in Table 1, which reports the average performance across orders for each benchmark. Full results are provided in Appendix C. Additionally, a brief efficiency analysis is available in D.

Results on CIL Table 1 presents the results under the CIL setup. Both L2R-wavg and L2R-merge outperform comparable baselines such as DAM, EPI, and MoCL. DAM shows the worst performance of the group, as it relies on task vectors created from training data, which are insufficient for properly routing the adapters. However, for Afrisenti, DAM achieves competitive results since this benchmark resembles a domain incremental learning scenario where task vectors effectively help distinguish task identity. EPI and MoCL perform better than DAM but still fall short of our method despite their sophisticated techniques. This highlights the importance of a well-learned router, which enables the router's ability to intelligently route and confer a significant performance advantage (Dikkala et al., 2023).

Regarding L2R versions, we observe that L2R-wavg has a clear advantage over L2R-merge. This may be due to interference between the parameters of multiple adapters, which can cause performance drops when they are merged (Yadav et al., 2024).

Results on TIL Table 1 presents the results under the TIL setup. L2R-wavg performs best in the MTL5 and AfriSenti benchmarks, while it lags behind in the WOS benchmark. Interestingly, both L2R-wavg and MoCL reach the Upper-Bound performance, suggesting that the learned routing function likely emulates the full activation of the corresponding adapter or an effective combination of the adapters. We hypothesize that L2R-wavg tends toward the latter, as evidenced by its performance in AfriSenti, where it surpasses the Upper-Bound.

In the WOS benchmark, our methods perform competitively but fall behind MoCL. We attribute this to the amount of data used to train the routing function. WOS consists of very small datasets, resulting in a memory populated with approximately 100 elements per task, which may not be sufficient for router learning. In fact, Dikkala et al. (2023) have shown that as the number of examples increases, the routing function reaches its optimal performance. We explore this in the next section.

Impact of Memory Size Figure 2 shows the performance for all benchmarks with memory sizes

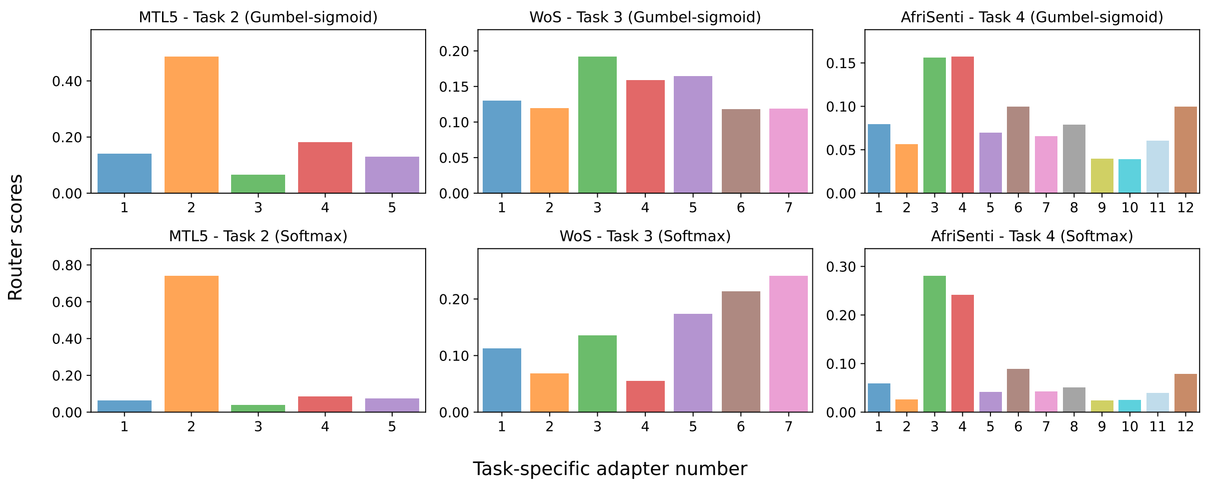

Figure 3: Average router scores for task 2 of MTL5 (order 4), task 3 of WOS (order 1), and task 4 of AfriSenti (order 1) using Gumbel-sigmoid (top) and Softmax (bottom). Scores were computed on the test sets.

Method: L2R-wavg	MTL5	WOS	AfriSenti
Gumbel-sigmoid	**78.0**	**79.90**	**60.04**
Softmax	77.8	60.74	59.90

Table 2: Accuracy results on MTL5, WOS, and AfriSenti benchmarks for the CIL setup of L2R-wavg using Gumbel-sigmoid and Softmax.

ranging from 1% to 30%. As expected, we find that the performance of L2R-wavg increases with larger memory sizes. Notably, the performance on WOS improves to match that of MoCL, supporting our assertion about the importance of using more examples for effective router adaptation. For MTL5 and AfriSenti, their performance gets close to the Upper-Bound, demonstrating that L2R may be learning better compositions for the adapters, leading to high performance across tasks.

Impact of Gumbel-sigmoid Distribution As discussed in section 3, we use Gumbel-sigmoid distribution during the training of the routers to theoretically enhance task-level modularity over activation competency offered by Softmax. To validate this, we compare the resulting routing scores from L2R-wavg in CIL with the ones from a version trained using Softmax function.

Table 2 shows the results, highlighting that using Gumbel-sigmoid during the training of the routing function consistently outperforms the use of Softmax activation. The improvement is observed across all benchmarks, with accuracy gains ranging from 0.1 to 19.2 points. The WOS benchmark exhibits the most significant improvement, likely because it consists of more tasks, which allows for more modules to be leveraged.

Figure 3 shows router probabilities for the test sets of MTL5 (order 4), WoS (order 1), and AfriSenti (order 1). On the left, for MTL5 task 2 (news classification), adapter 2 is the most activated in both model versions. However, the Softmax version predominantly uses adapter 2, while the Gumbel-sigmoid version also leverages adapters 1 and 4, which specialize in sentiment analysis.

In the middle, for the WOS task 3, both models activate multiple adapters because all tasks are document categorization across various domains. With Gumbel-sigmoid, adapter 3 (the correct one) is primarily activated, while the Softmax version strongly activates adapter 7, with less use of others.

On the right, for AfriSenti task 4 (sentiment analysis with varying input languages), the Gumbel-sigmoid version strongly activates adapters 3 and 4, with lighter activation of the rest. Adapters 3 and 4 correspond to Hausa and Igbo, both spoken in Nigeria, explaining their activation. Adapter 4 is slightly more activated, showing the router's ability to detect the target task while using other adapters. In contrast, the Softmax version primarily activates adapter 3, with less activation of others.

6 Conclusion

We introduced L2R, a method for routing in CL that enables PLMs to dynamically combine adapters, improving generalization and performance. Experiments across benchmarks demonstrate L2R's effectiveness in both TIL and the challenging CIL setting. We also show that larger memory enhances routing function learning and that L2R effectively leverages diverse adapter combinations.

Limitations

Our method focuses on developing an effective routing function to enhance generalization and performance on downstream tasks learned from a stream. To achieve this, we use a non-parametric memory to store previous examples, similar to replay-based methods. This approach may be limited in environments with storage constraints or data privacy concerns.

Our experimental setup primarily focuses on small-scale PLMs, which is a limitation since exploring large language models (LLMs) would be desirable given their dominance in the current NLP landscape. Although we have not tested our method with LLMs due to computational resource limitations, we anticipate similar results.

7 Acknowledgements

This work was funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme: CALCULUS project (Grant Agreement No. 788506) and KeepOnLearning project (Grant Agreement No. 101021347).

References

Jesujoba O. Alabi, David Ifeoluwa Adelani, Marius Mosbach, and Dietrich Klakow. 2022. Adapting pre-trained language models to African languages via multilingual adaptive fine-tuning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4336–4349, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Vladimir Araujo, Helena Balabin, Julio Hurtado, Alvaro Soto, and Marie-Francine Moens. 2022a. How relevant is selective memory population in lifelong language learning? In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 154–160, Online. Association for Computational Linguistics.

Vladimir Araujo, Julio Hurtado, Alvaro Soto, and Marie-Francine Moens. 2022b. Entropy-based stability-plasticity for lifelong learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3721–3728.

Feng Cheng, Ziyang Wang, Yi-Lin Sung, Yan-Bo Lin, Mohit Bansal, and Gedas Bertasius. 2024. Dynamic adapter merging for continual video question-answering learning.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nishanth Dikkala, Nikhil Ghosh, Raghu Meka, Rina Panigrahy, Nikhil Vyas, and Xin Wang. 2023. On the benefits of learning to route in mixture-of-experts models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9376–9396, Singapore. Association for Computational Linguistics.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Tyler L Hayes and Christopher Kanan. 2021. Selective replay enhances learning in online continual analogical reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3502–3512.

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

Julio Hurtado, Alain Raymond-Sáez, Vladimir Araujo, Vincenzo Lomonaco, Alvaro Soto, and Davide Bacciu. 2023. Memory population in continual learning via outlier elimination. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 3481–3490.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 364–371.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press.

Shamsuddeen Muhammad, Idris Abdulmumin, Abinew Ayele, Nedjma Ousidhoum, David Adelani, Seid Yimam, Ibrahim Ahmad, Meriem Beloucif, Saif Mohammad, Sebastian Ruder, Oumaima Hourrane, Alipio Jorge, Pavel Brazdil, Felermino Ali, Davis David, Salomey Osei, Bello Shehu-Bello, Falalu Lawan, Tajuddeen Gwadabe, Samuel Rutunda, Tadesse Belay, Wendimu Messelle, Hailu Balcha, Sisay Chala, Hagos Gebremichael, Bernard Opoku, and Stephen Arthur. 2023. AfriSenti: A Twitter sentiment analysis benchmark for African languages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13968–13981, Singapore. Association for Computational Linguistics.

Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2023. Combining parameter-efficient modules for task-level generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 687–702, Dubrovnik, Croatia. Association for Computational Linguistics.

A. Prabhu, H. Al Kader Hammoud, P. Dokania, P. S. Torr, S. Lim, B. Ghanem, and A. Bibi. 2023. Computationally budgeted continual learning: What does matter? In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3698–3707, Los Alamitos, CA, USA. IEEE Computer Society.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II*, page 524–540, Berlin, Heidelberg. Springer-Verlag.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017a. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017b. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *Preprint*, arXiv:1701.06538.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. 2023. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11909–11919.

Pablo Sprechmann, Siddhant Jayakumar, Jack Rae, Alexander Pritzel, Adria Puigdomenech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. 2018. Memory-based parameter adaptation. In *International Conference on Learning Representations*.

Eli Verwimp, Rahaf Aljundi, Shai Ben-David, Matthias Bethge, Andrea Cossu, Alexander Gepperth, Tyler L. Hayes, Eyke Hüllermeier, Christopher Kanan, Dhireesha Kudithipudi, Christoph H. Lampert, Martin Mundt, Razvan Pascanu, Adrian Popescu, Andreas S. Tolias, Joost van de Weijer, Bing Liu, Vincenzo Lomonaco, Tinne Tuytelaars, and Gido M van de Ven. 2024. Continual learning: Applications and the road forward. *Transactions on Machine Learning Research*.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. 2024. Rehearsal-free modular and compositional continual learning for language models. *Preprint*, arXiv:2404.00790.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022a. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and Wenqiu Zeng. 2023. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10933–10946, Toronto, Canada. Association for Computational Linguistics.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022b. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 139–149.

Zirui Wang, Sanket Vaibhav Mehta, Barnabas Poczos, and Jaime Carbonell. 2020. Efficient meta lifelong-learning with limited memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 535–548, Online. Association for Computational Linguistics.

Martin Wistuba, Prabhu Teja Sivaprasad, Lukas Balles, and Giovanni Zappella. 2024. Choice of peft technique in continual learning: Prompt tuning is not all you need. *Preprint*, arXiv:2406.03216.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2024. Ties-merging: resolving interference when merging models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. 2024. Continual learning with pre-trained models: A survey. *Preprint*, arXiv:2401.16386.

A Additional Benchmark Details

MTL5 is a benchmark for text classification. It consists of five datasets from (Zhang et al., 2015). AGNews classification, Yelp sentiment analysis, Amazon sentiment analysis, DBPedia article classification, and Yahoo questions and answers categorization. Both sentiment analysis tasks share the same labels. In line with d'Autume et al. (2019), we use 575,000 training and 38,000 test examples with 33 classes from all datasets using 4 task orders:

(i) AGNews → Yelp → Amazon → Yahoo → DBpedia
(ii) Yelp → Yahoo → Amazon → DBpedia → AGNews
(iii) DBPedia → Yahoo → AGNews → Amazon → Yelp
(iv) Yelp → AGNews → DBPedia → Amazon → Yahoo

Web-of-science (WOS) (Kowsari et al., 2017) initially began as a hierarchical dataset for categorizing documents. It includes research papers from seven distinct fields: biochemistry, civil engineering, computer science, electrical engineering, medical science, mechanical engineering, and psychology. Each of these fields represents a high-level category for document classification, with multiple subcategories within each field. This dataset has about 11,967 instances and 33 classes. In line with Wang et al. (2023), we split to train/val/test set in the ratio of 0.6:0.2:0.2 and structured 7 sequential learning tasks based on the high-level categories of the dataset:

(i) 1 → 2 → 3 → 4 → 5 → 6 → 7

AfriSenti (Muhammad et al., 2023) is a multilingual sentiment analysis dataset comprising 12 low-resource African languages. These languages include Amharic (am), Algerian Arabic (dz), Hausa (ha), Igbo (ig), Kinyarwanda (kr), Moroccan Arabic (ma), Nigerian Pidgin (pcm), Mozambican Portuguese (pt), Swahili (sw), Xitsonga (ts), Twi (twi), and Yoruba (yo). The dataset contains over 110,000 annotated tweets and spans three sentiment classes across all languages. In line with (Wang et al., 2023), we use 3 task orders:

(i) am → dz → ha → ig → kr → ma → pcm → pt → sw → ts → twi → yo
(ii) ma → pcm → kr → pt → ig → sw → ha → ts → dz → twi → am → yo
(iii) am → dz → ha → ma → ig → kr → sw → ts → twi → yo → pcm → pt

Following Wang et al. (2023), we categorize benchmarks based on task domain similarity. WOS and Afrisenti serve as near-domain benchmarks due to their closely related tasks. MTL5, on the other hand, represents far-domain benchmarks where distinct domain boundaries among tasks are evident.

B Additional Implementation Details

As backbones, we use BERT-base (Devlin et al., 2019) for MTL5 and WOS and AfroXLMR-base (Alabi et al., 2022) for AfriSenti. Additionally, we adopt LoRA (Hu et al., 2022) as our PEFT modules, using a rank of 8 and dropout of 0.1. We applied LoRA to W_q and W_v, and it has been demonstrated to be a competitive configuration. These adapters allow efficient fine-tuning, representing less than 0.7% of the PLM's parameters in our experiments.

Our memory is non-parametric, which means that it stores raw examples with labels sampled from training examples. We only sample 10% of each task's training dataset, representing less than 3.8MB per dataset in our experiments, as we use Parquet, an efficient storage format.

During the training phase, we use a batch size of 8 and a learning rate of 3e-4. Additionally, we use a linear scheduler with warmup and AdamW optimizer. On the other hand, we use the same configuration to train the router networks. However, we use a learning rate of 3e-4 or 3e-5 for MTL5 and AfriSenti, and a learning rate of 3e-3 or 3e-4 for WOS.

C Full Results

Table 3 and Table 4 show the full results for CIL and TIL setups, respectively. We observe that our method consistently outperforms the best baseline (MoCL) across orders in CIL. Interestingly, our model achieves a similar performance across orders. This is because the isolation strategy helps the model to have a very specialized adapter regardless of the order of training. However, order (iii,iv) of MTL5 and order (ii) of AfriSenti show slightly superior performance, indicating that memory population methods could be leveraged to further improve performance (Araujo et al., 2022a).

Regarding TIL experiments, we find that L2R-wavg obtains a similar performance to MoCL and Upper-Bound. Interestingly, L2R-merge lags behind the performance of MoCL, potentially due to the need for more data to properly train its router, as shown in our experiments. This is more evident for our L2R-merge, which has also been shown to be less effective to L2R-wavg possible for interference produced by merging the adapters.

	MTL5					WOS	AfriSenti			
Method	**i**	**ii**	**iii**	**iv**	**Avg**	**i**	**i**	**ii**	**iii**	**Avg**
Lower-Bound	16.2	16.2	16.2	16.2	16.2	15.07	33.64	33.64	33.64	33.64
Upper-Bound	79.4	79.4	79.4	79.4	79.4	90.06	62.16	62.16	62.16	62.16
DAM	27.4	27.4	27.4	27.4	27.4	10.00	48.78	48.78	48.78	48.78
EPI†	77.4	77.3	77.2	77.4	77.3	77.83	42.96	42.97	43.36	43.10
MoCL†	73.0	74.0	74.8	73.6	73.8	79.23	45.57	44.32	46.95	45.61
L2R-wavg	77.8	77.9	78.2	78.1	**78.0**	**79.90**	59.14	60.8	60.18	**60.04**
L2R-merge	77.7	77.7	77.6	77.8	**77.7**	**79.98**	53.36	52.07	53.03	**52.82**

Table 3: Results on MTL5, WOS, and AfriSenti for CIL. † indicates that the results come from (Wang et al., 2024).

	MTL5					WOS	AfriSenti			
Method	**i**	**ii**	**iii**	**iv**	**Avg**	**i**	**i**	**ii**	**iii**	**Avg**
Lower-Bound	16.2	16.2	16.2	16.2	16.2	15.07	33.64	33.64	33.64	33.64
Upper-Bound	79.4	79.4	79.4	79.4	79.4	90.06	62.16	62.16	62.16	62.16
ProgPrompt†	78.0	77.9	77.9	77.9	77.9	89.93	50.16	46.74	50.30	49.07
EPI†	77.4	77.3	77.2	77.4	77.3	77.83	41.49	42.65	45.16	43.10
MoCL†	79.3	79.6	79.2	79.4	**79.4**	**90.59**	57.05	56.52	56.74	56.77
L2R-wavg	79.4	79.4	79.5	79.4	**79.4**	89.24	62.68	62.78	62.39	**62.62**
L2R-merge	79.2	79.3	79.4	79.3	79.3	89.16	54.06	53.3	54.55	53.97

Table 4: Results on MTL5, WOS, and AfriSenti for TIL. † indicates that the results come from (Wang et al., 2024).

D Efficiency Comparison: FLOPs Analysis

We compute the theoretical amount of FLOPs (floating-point operations) of our methods and some baselines for the MTL5 benchmark (i.e., 5 tasks). In Table 5, we observe BERT for classification as the more efficient model. This is expected as all the other methods augment BERT with PEFT modules. Our methods are the second most efficient models. L2R-wave is the less efficient one, as it always uses all the adapters and composes the input them. L2R-merge and DAM have the same amount of FLOPs as these models merge the adapters in one, making the computation more efficient. Although MoCL is the direct competitor to our methods in performance, it is not in terms of efficiency as this model extends the input, resulting in more operations.

Method	FLOPs
BERT	325.42 GFLOPS
DAM	326.49 GFLOPS
MoCL	477.85 GFLOPS
L2R-wavg	330.80 GFLOPS
L2R-merge	326.49 GFLOPS

Table 5: Theoretical FLOPs when processing a batch size 1 and a sequence of 128 tokens.