Learn it or Leave it: Module Composition and Pruning for Continual Learning

Mingyang Wang[1,2,3] **Heike Adel**[4] **Lukas Lange**[1]
Jannik Strötgen[5] **Hinrich Schütze**[2,3]
[1]Bosch Center for Artificial Intelligence, Renningen, Germany
[2]LMU Munich, Germany [3]Munich Center for Machine Learning (MCML)
[4]Hochschule der Medien, Stuttgart, Germany
[5]Karlsruhe University of Applied Sciences, Germany
mingyang.wang2@de.bosch.com

Abstract

In real-world environments, continual learning is essential for machine learning models, as they need to acquire new knowledge incrementally without forgetting what they have already learned. While pretrained language models have shown impressive capabilities on various static tasks, applying them to continual learning poses significant challenges, including avoiding catastrophic forgetting, facilitating knowledge transfer, and maintaining parameter efficiency. In this paper, we introduce MoCL-P, a novel lightweight continual learning method that addresses these challenges simultaneously. Unlike traditional approaches that continuously expand parameters for newly arriving tasks, MoCL-P integrates task representation-guided module composition with adaptive pruning, effectively balancing knowledge integration and computational overhead. Our evaluation across three continual learning benchmarks with up to 176 tasks shows that MoCL-P achieves state-of-the-art performance and improves parameter efficiency by up to three times, demonstrating its potential for practical applications where resource requirements are constrained.

1 Introduction

Continual learning (CL) is a learning paradigm aiming at incrementally acquiring and integrating new knowledge over time without forgetting existing knowledge. This capability is essential for machine learning models to stay effective as they encounter dynamic and evolving real-world environments. While pretrained language models (PLMs) have demonstrated remarkable capabilities on various static tasks, adapting them for continual task learning remains challenging.

In particular, there are three notable challenges for continual learning. (1) Avoiding catastrophic forgetting: The newly learned information should not disrupt and degrade previously acquired knowledge (McCloskey and Cohen, 1989). (2) Facilitating knowledge transfer: The knowledge from past tasks should be reused for efficient learning of new tasks. (3) Maintaining parameter efficiency: The language models need to stay lightweight and effective even if the continual learning sequence scales to hundreds of tasks.

To mitigate catastrophic forgetting, a line of prior works adopt the idea of *parameter isolation* (Razdaibiedina et al., 2022; Wang et al., 2023d,e, 2024), which allocates isolated parameters dedicated for each task to avoid inter-task interference. While parameter isolation typically does not allow knowledge transfer across tasks (Wang et al., 2023d,e), there are attempts to address both challenges of catastrophic forgetting and knowledge transfer at the same time, e.g., by progressively concatenating (Razdaibiedina et al., 2022) or composing task-specific modules (Wang et al., 2024).

Despite their effectiveness in terms of task performance, parameter isolation methods do not scale well with the number of tasks. When the number of tasks in a continual learning sequence is growing into the hundreds, the progressive expansion of task-specific parameters leads to parameter inefficiency and significantly increases computational and storage costs.

In this paper, we address all three continual learning challenges simultaneously and introduce MoCL-P, a lightweight continual learning approach that leverages task representation-guided module composition and adaptive pruning. First, to avoid catastrophic forgetting, MoCL-P continually adds task-specific modules to PLMs for learning new tasks while keeping the modules frozen once the training on the respective tasks is finished. In addition, to enable knowledge transfer across tasks, MoCL-P allows the model to reuse existing knowledge via module composition. Finally, to keep the language model lightweight, MoCL-P adopts an adaptive pruning strategy by removing

Proceedings of the 9th Workshop on Representation Learning for NLP (RepL4NLP-2024), pages 163–176

modules with redundant information and retaining only the most salient modules throughout the continual learning process.

In our evaluation on three popular datasets as continual learning benchmarks with up to 176 tasks in the learning sequence, MoCL-P stands out by not only showing state-of-the-art performance but also outperforming prior algorithms in parameter efficiency by up to three times across benchmarks.

To the best of our knowledge, this is the first paper that tackles the three challenges of continual learning simultaneously: MoCL-P avoids catastrophic forgetting, allows knowledge transfer and ensures parameter efficiency. Thus, MoCL-P proposes a sustainable way for continual learning, allowing models to remain lightweight and effective as they evolve with accumulating tasks.

The code base for MoCL is available online.[1]

2 Related Work

2.1 Avoiding Catastrophic Forgetting in Continual Learning

A major challenge in continual learning is known as catastrophic forgetting, where newly learned information disrupts and degrades previously acquired knowledge (McCloskey and Cohen, 1989). Existing approaches to overcome this issue can be broadly divided into three categories (Wang et al., 2023a): (1) *Regularization*-based methods explicitly add regularization terms to the loss function to restrict model updates and preserve existing knowledge (Li and Hoiem, 2017; Kirkpatrick et al., 2017; Aljundi et al., 2018); (2) *Rehearsal*-based methods leverage a memory buffer to store real examples (Rebuffi et al., 2017; Rolnick et al., 2019; Zhang et al., 2022a) or generated pseudo-examples of past tasks for future rehearsal to avoid catastrophic forgetting (Shin et al., 2017; Su et al., 2019); (3) *Parameter isolation*-based methods construct task-specific parameters to prevent inter-task interference by either dynamically expanding model capacity or isolating existing model weights (Madotto et al., 2020; Zhang et al., 2022b; Razdaibiedina et al., 2022; Wang et al., 2023e,d, 2024).

Our method, MoCL-P, belongs to the parameter-isolation based category. We use task representation-guided module composition and adaptive pruning to effectively manage isolated parameters.

[1] https://github.com/boschresearch/MoCL-Pruning

2.2 Transferring Knowledge in Continual Learning

Recent studies in continual learning demonstrate the effectiveness of parameter isolation methods in avoiding catastrophic forgetting (Razdaibiedina et al., 2022; Wang et al., 2023e,d, 2024). However, naive parameter isolation methods do not allow knowledge transfer across tasks, which leads to inefficient learning as the model cannot leverage previously acquired knowledge to facilitate learning new tasks. To address this, Yoon et al. (2017) and Zhu et al. (2022) attempt to first identify reusable modules and only add new parameters when necessary. Ke et al. (2021) and Wang et al. (2022) introduce knowledge-sharing modules to facilitate knowledge transfer while maintaining task-specific parameters to prevent interference. Razdaibiedina et al. (2022) progressively concatenate task-specific modules to incrementally build a composite model that leverages both new and existing knowledge. Wang et al. (2024) introduce a modular and compositional continual learning framework to compose the new module with existing ones based on task module matching.

2.3 Parameter-Efficient Continual Learning

With the ever-increasing number of parameters in PLMs, it becomes increasingly important to develop machine learning systems that are more scalable, practical, and resource-efficient. In the context of continual learning, this necessitates parameter-efficient approaches that can effectively integrate new knowledge without excessive computational and storage costs as the number of tasks in the continual learning sequence increases.

Recent advancements in continual learning integrate parameter isolation with parameter-efficient fine-tuning (PEFT), i.e., they allocate task-specific PEFT modules for learning and inference (Razdaibiedina et al., 2022; Wang et al., 2023e,d, 2024). Various PEFT techniques, such as adapter tuning (Houlsby et al., 2019), prefix tuning (Li and Liang, 2021), and LoRA (Huang, 2022), have been applied in continual learning. Although they reduce the number of training parameters to some extent by freezing the PLM and only updating the PEFT module parameters, it remains challenging to apply them to long-sequence benchmarks that consist of hundreds of tasks. The continuous expansion of task-specific modules leads to significant computational overhead as the number of tasks increases.

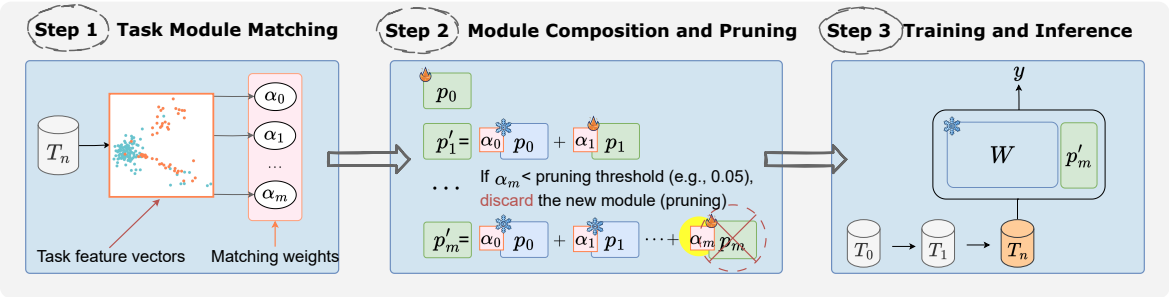

Figure 1: Overview of our proposed method MoCL-P for parameter-efficient continual learning. **Step 1**: We match the n-th task input with task feature vectors to determine the contribution of each existing module for learning the current task. **Step 2**: We compose the newly initialized module with existing ones and perform adaptive module pruning to preserve only the dominant modules. **Step 3**: Finally, we combine the composed module p'_m with the PLM for training and inference.

Our approach builds on the idea of Wang et al. (2024) by utilizing task representations for module composition, ensuring that the model effectively reuses relevant knowledge from previous tasks. Beyond that, we introduce an adaptive pruning strategy to keep the language model lightweight and effective throughout the continual learning process, thus making it scalable for continual learning scenarios with long task sequences.

3 Problem Definition

Continual learning focuses on addressing a series of tasks which arrive in a sequential order. The primary goal is to optimize the model's average performance across all tasks after learning them sequentially. Formally, the sequence of tasks is denoted as $\{T_1, \ldots, T_N\}$. Each task contains a set of input samples $\{(x_n^i, y_n^i)\}$. For the text classification tasks we study in this work, x_n^i is the input text, y_n^i is the ground-truth label, and $n \in \{1, \ldots, N\}$ is the task identity.

In this work, we focus on rehearsal-free continual learning, i.e., data from earlier tasks is not available when training later tasks. Therefore, our model does not suffer from the memory or privacy issues associated with rehearsal-based methods. We assume the task labels are provided during both training and testing, i.e., task-incremental continual learning (Wang et al., 2023a). However, MoCL-P can be adapted for class-incremental learning, where the task labels are not given during testing, with minor modifications following Wang et al. (2024). We leave the exploration of other continual learning settings for future work.

4 Method

In this section, we describe MoCL-P, our proposed CL approach for language models, as illustrated in Figure 1, which tackles catastrophic forgetting and enhances knowledge transfer with superior parameter efficiency at the same time.

4.1 Continual Learning with PEFT

We inherit the idea of parameter isolation with parameter-efficient fine-tuning (PEFT) introduced in prior work (Razdaibiedina et al., 2022; Wang et al., 2023d,e, 2024), which allocates trainable PEFT parameters for each task while keeping other parameters frozen.

We utilize *prefix-tuning* (Li and Liang, 2021) as the PEFT module in consistency with prior works.[2] For each task in the CL sequence, we add a set of trainable PEFT parameters, i.e., a task-specific module, to the pretrained language model (PLM) for downstream task fine-tuning. Instead of updating the whole model, only a small number of the PEFT parameters are optimized. Once training on one given task is completed, the corresponding PEFT module is frozen to preserve the task-specific knowledge in the subsequent training process, thus avoiding catastrophic forgetting.

4.2 Task Representation-Guided Module Matching

In contrast to completely isolating task-specific parameters during continual learning, which excludes knowledge transfer, we follow the idea of

[2]Other PEFT methods like Adapter (Houlsby et al., 2019) and LoRA (Hu et al., 2021) can also be combined with MoCL-P in general. We leave such exploration for future work.

task module composition introduced in Wang et al. (2024) to facilitate knowledge transfer.

To this end, we utilize task representations for task module matching, and consequently for composing old and new modules for learning. The module matching aims to determine the contribution of each existing module to learning the current task, i.e., to what extent previously learned modules can be reused for the current task.

We introduce trainable feature vectors $V \in \mathbb{R}^{N \times D}$ as task representations to capture the features of each task in the CL sequence.[3] We set the dimension of each task feature vector $v \in \mathbb{R}^D$ to the same value as the dimension of the input embeddings $x_n \in \mathbb{R}^D$. Then, we calculate the cosine similarity between the input embeddings x_n and each feature vector v_i up to the current task as the matching score $\alpha_i = \cos(x_n, v_i)$. Consequently, we get the module matching weights $\{\alpha_0, \alpha_1, ...\}$ for module composition (details will be introduced in Section 4.3) to reuse existing knowledge.

4.3 Module Composition with Adaptive Pruning

When the CL learning sequence scales to dozens or hundreds of tasks, the need for efficiency increases. Continuously expanding the module pool to assign a PEFT module to each task, as done in prior works (Wang et al., 2023e; Razdaibiedina et al., 2022; Wang et al., 2024), leads to large computational costs. In contrast, we employ an adaptive pruning strategy to make our approach scalable in scenarios with long task sequences.

In particular, our pruning strategy aims at preserving only those modules that add new and valuable information to the set of already selected modules. Given a set of selected modules $\{P_0, \ldots, P_{m-1}\}$ from previous tasks and a new task T_n, $(m - 1 \ll n)$, we initialize a trainable module P_m and add it temporarily to the model. For each instance[4] x_n^i of the current task T_n, we compute the matching weights $\{\alpha_0, \ldots, \alpha_m\}$ by matching x_n with all task feature vectors $\{v_0, \ldots, v_m\}$ from our current set of modules. Specifically, we calculate the cosine similarity between x_n and $\{v_0, \ldots, v_m\}$ as module matching weights $\alpha_{0:m}$ as detailed in Section 4.2.

[3]Note that MoCL-P is agnostic to different types of task representations. In addition to the trainable feature vectors, other static task representations such as task embeddings or Gaussian task distributions can also be combined with MoCL-P. We analyze these options in Section 6.2.

[4]For simplicity, we refer to this as x_n in the following.

Then, we compose the new and old modules via a weighted sum: $P'_m = \sum_{k=0}^{m} \alpha_k P_k$. Finally, the composed module P'_m is combined with the PLM, consisting of all the selected module components up to the current task.

After the training on T_n is finished (specifically, the training of the PEFT module P_m and the task feature vector v_m), we compare α_m, the matching weight of the new module P_m, with a threshold[5] to decide whether to prune P_m or leave it in the set of existing modules. The intuition is that large matching weights indicate new and valuable information, while task modules with small matching weights do not contribute new information and, thus can, be discarded.

4.4 Training and Inference

The training objective for the n-th task in the continual learning sequence is to find the PEFT module P_m and the task feature vector v_m that minimize the cross-entropy loss of training examples, and, at the same time, maximize the cosine similarity between the task-specific feature vector v_m and the corresponding task input embeddings x_n:

$$\min_{P_m, v_m} - \sum_{x_n, y_n} \log p(y_n | x_n, P'_n, \theta) - \sum_{x_n} \cos(x_n, v_m)$$

Here $P'_n = \sum_{k=1}^{m} \alpha_k P_k$ is the weighted summation of the new trainable task module and the existing frozen task modules as introduced in Section 4.3. During inference, MoCL-P performs per-instance task module matching and composition. The resulting module is combined with the PLM for inference.

5 Experimental Setup

In this section, we describe datasets, training details and baselines for our experiments.

5.1 Datasets

To evaluate the performance of our method and the effectiveness of its module pruning functionality, we experiment with three continual learning benchmarks, each with long task sequences. Following prior work (Razdaibiedina et al., 2022; Wang et al., 2024), we use MTL15, a multi-task continual learning benchmark comprising 15 classification tasks, and AfriSenti (Muhammad et al., 2023), a multilingual sentiment analysis dataset that includes 12 low-resource African languages. Additionally, we

[5]The threshold is a tunable hyperparameter.

Method	AfriSenti					WikiAnn				
	AVG	O1	O2	O3	**# Params**	**AVG**	O1	O2	O3	**# Params**
Seq FT (F)	6.17	5.62	6.52	6.30	560M	14.50	3.44	23.36	16.70	110M
Seq FT	49.10	50.05	49.74	47.53	0.4M	67.99	68.25	65.05	70.66	0.1M
Per-task FT	52.41	52.41	52.41	52.41	4.5M	71.22	71.22	71.22	71.22	24.8M
ProgPrompt	49.07	50.16	46.74	50.30	4.5M	73.20	73.24	73.22	73.15	24.8M
EPI	43.10	41.49	42.65	45.16	4.5M	67.34	67.72	67.12	67.18	24.8M
MoCL	59.31	59.56	58.98	59.40	4.5M	73.80	73.78	73.81	73.82	24.9M
MoCL-P (Ours)	**59.41**	59.52	58.97	59.76	2.2M$\pm$0.4	**73.91**	73.94	73.94	73.86	8.0M$\pm$0.1

Table 1: Summary of the continual learning results on two multilingual benchmarks: AfriSenti and WikiAnn NER, with 12 and 176 languages in the task sequence, respectively. We compare MoCL-P with various baseline methods, and show the average model performance (AVG) across different task orders (O1, O2, O3) with the number of trainable parameters (# Params) used by each method. MoCL-P outperforms other methods with significantly fewer parameters, demonstrating its superiority in both model performance and parameter efficiency in long-sequence continual learning scenarios.

include WikiAnn (Pan et al., 2017), a multilingual named entity recognition (NER) dataset covering 176 languages; its long task sequence provides an adequate testbed for the pruning ability of our approach.

We report macro-weighted F1 scores on the AfriSenti benchmark, accuracy on MTL15, and micro-weighted F1 scores on WikiAnn. On the MTL15 benchmark, we select 1000 random samples per class for training each task and hold out 500 samples per class for validation.[6] We explore three task orders for each benchmark, adopting the same multiple task orders as the prior work. Please refer to Appendix A.1 for more details about the benchmarks and task orders.

5.2 Training Details

We deploy three LMs for these datasets, in line with prior work (Razdaibiedina et al., 2022; Wang et al., 2024). We use encoder-based models for AfriSenti and WikiAnn NER (AfroXLM and BERT, respectively), and the encoder-decoder model T5 for MTL15. Prefix-tuning is used as the task-specific modules for all deployed models. All design choices are consistent with previous works to ensure a fair comparison. The reported results represent the average performance after training on all tasks consecutively and are averaged over three random seeds. The detailed experimental settings are provided in Appendix A.2.1.

[6]All design choices of MoCL-P are kept consistent with previous works (Wang et al., 2023b,d, 2024) to ensure a fair comparison.

5.3 Baselines

To compare different CL methods, we include the following baselines: (1) Sequential fine-tuning continuously fine-tunes the language model on the task sequence: (a) **Seq FT (F)** refers to all model parameters are updated (fully fine-tuning), while (b) **Seq FT** only fine-tunes the PEFT parameters; (2) **Per-task FT** trains a separate PEFT module for each task; and the parameter isolation-based methods (3) **ProgPrompt** (Razdaibiedina et al., 2022) assigns task-specific parameters and progressively concatenates modules of all tasks to encourage knowledge transfer; (4) **EPI** (Wang et al., 2023e) introduces a non-parametric task identification technique to select modules for task training and inference; (5) **O-LoRA** (Wang et al., 2023d) learns tasks in different low-rank vector spaces that are kept orthogonal to each other to minimize interference; and (6) **MoCL** (Wang et al., 2024) introduces a modular and compositional framework that progressively expands task-specific modules and composes the new module with existing ones to facilitate knowledge transfer. A detailed description of these methods can be found in Appendix A.2.2.

6 Results and Analysis

In this section, we present and analyze our experimental results.

6.1 Overall Results

Table 1 shows the performance of MoCL-P and other baseline methods on the AfriSenti and WikiAnn benchmarks. MoCL-P consistently outperforms the baselines while significantly reducing the number of trainable parameters. Using

only 50% and 30% of the trainable parameters compared to other CL methods on Afrisenti and Wikiann respectively, MoCL-P showcases an exceptional balance of efficiency and performance. In the MTL15 benchmark, as illustrated in Table 2, MoCL-P also shows superior performance. As mentioned in prior work (Wang et al., 2024), tasks in this benchmark share lower similarity compared to AfriSenti and WikiAnn, resulting in weaker reusability of task modules. Therefore, we do not observe a significant drop in the number of trainable parameters here as seen in the other benchmarks. However, we still achieve a 25% reduction in parameter size while maintaining final performance.

Overall, MoCL-P demonstrates its superiority in efficiently managing the continual learning process without the substantial parameter overhead. The competitive performance of MoCL-P across different benchmarks highlights its robust adaptability and scalability to the continual learning sequence up to 176 tasks long.

Method	MTL15				
	AVG	O1	O2	O3	# Params
Seq FT-F	7.4	7.4	7.4	7.5	770M
Seq FT-P	64.7	69.9	58.9	65.1	1.4M
Per-task FT	80.5	80.5	80.5	80.5	21.1M
ProgPrompt	77.9	78.0	77.7	77.9	21.1M
EPI	65.4	62.8	65.3	68.1	21.1M
O-LoRA	69.6	78.0	77.7	77.9	33.8M
MoCL	**82.5**	82.9	82.8	81.9	21.1M
MoCL-P (Ours)	**82.5**	83.0	82.7	81.8	15.6M$\pm$1.1

Table 2: Summary of the continual learning results on the MTL15 benchmark with the T5-large model. MoCL-P achieves the best average performance (AVG) while using fewer parameters (# Params), demonstrating its effectiveness on the multi-task CL benchmark.

6.2 Task Representation Comparison

In this work, we adopt learnable task feature vectors as task representation, and based on these, we perform module composition and pruning. In Section 6.1, we demonstrate the effectiveness of this design choice. While this is not the only option for task representations, in this section, we experiment with two other types of task representation: (1) using Gaussian distributions to model the input embeddings of each task (*w/ Gaussian*) and (2) calculating the mean of the input embeddings of each task for task representations(*w/ Embed mean*).

Table 3 provides the results of using different task representation options for module composition and pruning. A significant performance drop occurs when using Gaussian distributions or the mean of task input embeddings as the task representation. In most cases, their performance is worse than the Per-Task FT baseline, indicating that using these task representations for module composition leads to performance degradation rather than beneficial knowledge transfer across tasks.

	AfriSenti		WikiAnn	
	AVG	# Params	AVG	# Params
Per-task FT	49.10	4.5M	71.22	24.8M
MoCL	59.31	4.5M	73.80	24.9M
w/ Gaussian	42.25	4.5M	67.38	24.8M
w/ Embed mean	52.63	4.5M	70.12	24.8M
MoCL-P (Ours)	**59.41**	2.2M$\pm$0.4	**73.91**	8.0M$\pm$0.1
w/ Gaussian	42.15	4.1M$\pm$0.0	67.46	5.4M$\pm$0.3
w/ Embed mean	52.13	3.9M$\pm$0.2	70.21	20.3M$\pm$0.7

Table 3: Performance comparison of different task representation methods for module composition and pruning. Specifically, we use Gaussian distribution to model the input embeddings of each task (*w/ Gaussian rep*) and calculate the mean of the input embeddings of each task (*w/ Embed mean*). Notably, the use of these variations results in substantially lower performance compared to the original MoCL and MoCL-P which utilizes learnable feature vectors as task representations.

We believe that this degradation is due to the fact that both of these task representations are static and are solely based on the input embeddings. In contrast, MoCL-P utilizes trainable task feature vectors, meaning the model can automatically learn to capture the salient task features necessary for effective module composition. Trainable task representations are a better choice because not all information in the input embedding is relevant for module composition. To effectively capture reusability between task modules, the model must focus on the salient features while ignoring irrelevant ones. Static task representations, which are purely based on input embeddings, fail to achieve this selective focus.

6.3 Ablation Study: Varying the Training Epochs for Task Feature Vectors

To substantiate our assumption introduced in Section 6.2, we additionally conduct ablation experiments on WikiAnn where we vary the training epochs for task feature vectors in MoCL-P. As illustrated in Figure 3, training the task feature vectors for different epochs shows a clear pattern: the model performance improves significantly with the initial increase of the number of training epochs.

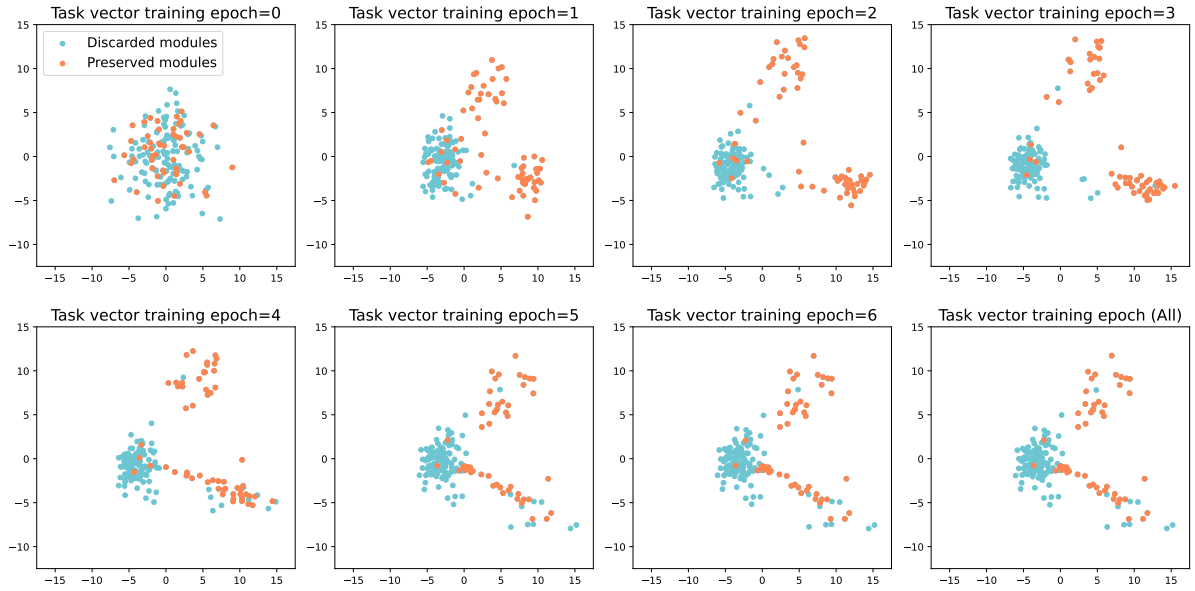

Figure 2: Visualization of task feature vectors on the WikiAnn benchmark using PCA for dimensionality reduction. We vary the training epochs for task feature vectors in MoCL-P. As the training epochs increase, the task feature vectors spread out from the initial dense cluster. Notably, the feature vectors of preserved modules (in orange) are spread across a large area while the discarded modules (in blue) form a dense cluster, indicating their redundancy.

Beyond a certain point (epoch = 4), additional training does not yield further benefits and converges towards a performance plateau. This observation suggests that by allowing the model to adapt these vectors over several epochs, MoCL-P can more accurately identify and leverage the most relevant features for module composition. This underscores the critical importance of the trainable nature of task feature vectors in MoCL-P.

In Figure 2, we visualize the task feature vectors at different training epochs on the WikiAnn dataset, which includes a total of 176 tasks. The colors represent two categories of task modules: those that are eventually discarded (blue) and those that are preserved (orange) through the learning process. Initially (training epoch = 0), the vectors are evenly distributed around the origin since they are uniformly initialized. As the training epochs increase, the task vectors spread out and become more distinct, suggesting that the model captures distinct features of tasks and utilizes them for module pruning. Notably, the feature vectors of preserved modules are spread across a large area, while the discarded modules form a dense cluster, indicating their redundancy. The embeddings of task feature vectors stabilize by epoch 5, indicating a convergence in the task representation learning process. These observed patterns demonstrate the effectiveness of our strategy of using trainable task representations for module composition and pruning, which helps in preserving only the most salient modules for continual learning.

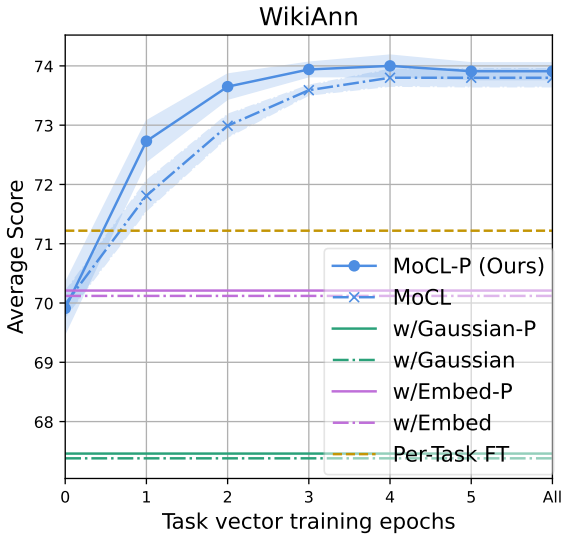

Figure 3: Experimental results with different training epochs for task feature vectors in MoCL-P. The model performance improves rapidly with the initial increase of the number of training epochs and converges towards a performance plateau after epochs > 4. MoCL-P achieves significantly better performance than other task representation options. It highlights the advantage of trainable task representations in capturing salient task features for effective module composition.

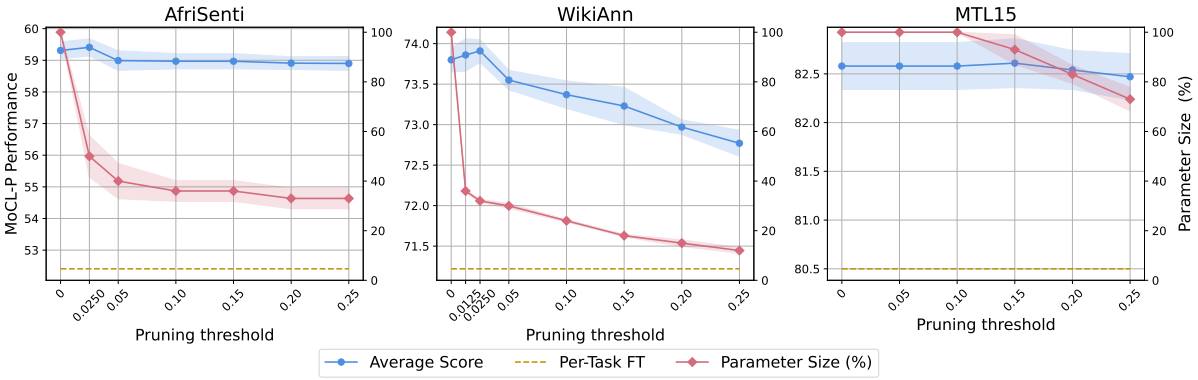

Figure 4: Impact of pruning thresholds on the performance and parameter size of MoCL-P across different benchmarks. The performance of MoCL-P exhibits robustness to different thresholds, with a significant reduction in the number of trainable parameters. This demonstrates that MoCL-P can maintain effective performance despite using fewer parameters.

6.4 Ablation Study: Varying the Pruning Threshold

In this section, we study the impact of using different thresholds on the performance of MoCL-P. As introduced in Section 4.3, we compare the matching weight of the newly initialized task module α_m with the pre-specified threshold α_{ths}, if $\alpha_m < \alpha_{ths}$, then we discard the newly learned module. We vary α_{ths} from 0 to 0.25 for the three benchmarks used in this work.

The results are shown in Figure 4. The figure illustrates how varying the pruning threshold affects both the average performance and the parameter size across different benchmarks.

For the model performance, we observe that the initial increase in the pruning threshold leads to a performance increase on all three benchmarks. This indicates that excluding the redundant modules benefits performance. As the threshold continues to increase, the average performance on AfriSenti and MTL15 remains relatively stable, while the performance on WikiAnn drops, possibly due to the loss of information in potentially useful modules. Additionally, it is worth noting that the performance of MoCL-P is consistently and significantly better than the Per-Task FT baseline, suggesting that MoCL-P achieves effective knowledge transfer at different pruning thresholds.

Furthermore, for the parameter size, a significant reduction is observed as the threshold increases on all three benchmarks. This demonstrates the superiority of MoCL-P on parameter efficiency. We observe that the parameter size decreases more pronounced and faster on AfriSenti and WikiAnn, while it decreases less and more slowly on MTL15. We believe it is due to the characteristics of the benchmarks. As mentioned in Section 6.1, tasks in this benchmark share a lower similarity, therefore, most task modules are highly specialized to these distinct tasks and cannot be discarded.

We choose different pruning thresholds for different benchmarks reported in Table 1. For each benchmark, we select the pruning threshold that best balances performance and parameter size to report the results in Table 1. Specifically, we use $\alpha_{ths} = 0.025$ for AfriSenti and WikiAnn, and $\alpha_{ths} = 0.25$ for MTL15. With these thresholds, MoCL-P achieves equally good performance with only 50%, 30%, and 75% of the number of trainable parameters compared to MoCL without pruning on these three benchmarks, respectively.

7 Conclusion

In this paper, we introduce MoCL-P, a novel continual learning approach that addresses the core challenges of catastrophic forgetting, knowledge transfer, and parameter efficiency in continual learning. We utilize learnable task representations for module composition and adaptive pruning, maintaining a lightweight model while achieving state-of-the-art performance across various benchmarks. Notably, MoCL-P scales effectively to long continual learning sequences, handling up to 176 tasks without compromising performance. These experimental results showcase MoCL-P's potential to enhance practical machine learning applications by effectively managing computational costs, thus providing a scalable and efficient solution for real-world scenarios where minimum resource requirements are crucial.

Limitations

While MoCL-P demonstrates significant advancements in continual learning, our study has some limitations that should be addressed in future work. First, we only use the long sequence multilingual benchmark, i.e., WikiAnn with 176 tasks, in this work due to the lack of existing long sequence multi-task benchmarks. The absence of these benchmarks limits the evaluation of MoCL-P's performance across diverse multi-task scenarios. Building a long sequence multi-task benchmark for continual learning would be an interesting research direction, although it is beyond the scope of this work. Second, as we follow the evaluation setup from prior works, we do not include generative tasks for evaluation. Therefore, we may not capture the potential of MoCL-P in a wider range of continual learning challenges. Including generative tasks in future evaluations would provide a more comprehensive understanding of MoCL-P's capabilities.

References

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

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Xiaolei Huang. 2022. Easy adaptation to mitigate gender bias in multilingual text classification. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 717–723, Seattle, United States. Association for Computational Linguistics.

Zixuan Ke, Hu Xu, and Bing Liu. 2021. Adapting BERT for continual learning of a sequence of aspect sentiment classification tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4746–4755, Online. Association for Computational Linguistics.

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

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.

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

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, and Zhiguang Wang. 2020. Continual learning in task-oriented dialogue systems. *arXiv preprint arXiv:2012.15504*.

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

Shamsuddeen Hassan Muhammad, Idris Abdulmumin, Abinew Ali Ayele, Nedjma Ousidhoum, David Ifeoluwa Adelani, Seid Muhie Yimam, Ibrahim Sa'id Ahmad, Meriem Beloucif, Saif Mohammad, Sebastian Ruder, et al. 2023. Afrisenti: A twitter sentiment analysis benchmark for african languages. *arXiv preprint arXiv:2302.08956*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages.

In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2022. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

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

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32.

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

Xin Su, Shangqi Guo, Tian Tan, and Feng Chen. 2019. Generative memory for lifelong learning. *IEEE transactions on neural networks and learning systems*, 31(6):1884–1898.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023a. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schuetze. 2023b. GradSim: Gradient-based language grouping for effective multilingual training. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schuetze. 2024. Rehearsal-free modular and compositional continual learning for language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, Mexico City, Mexico. Association for Computational Linguistics.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. 2023c. NLNDE at SemEval-2023 task 12: Adaptive pretraining and source language selection for low-resource multilingual sentiment analysis. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 488–497, Toronto, Canada. Association for Computational Linguistics.

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

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and Wenqiu Zeng. 2023e. 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, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. 2022. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022a. Continual sequence generation with adaptive compositional modules. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3653–3667, Dublin, Ireland. Association for Computational Linguistics.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022b. Continual sequence generation with adaptive compositional modules. *arXiv preprint arXiv:2203.10652*.

Qi Zhu, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang. 2022. Continual prompt tuning for dialog state tracking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1124–1137, Dublin, Ireland. Association for Computational Linguistics.

A Appendix

A.1 Dataset Information

Here we provide detailed information on the datasets used in this work. The MTL15 benchmark consists of 15 classification tasks, combining five datasets from the standard CL benchmark MTL5 (AG News, Amazon reviews, Yelp reviews, DBpedia, and Yahoo Answers) (Zhang et al., 2015), four tasks from the GLUE benchmark (MNLI, QQP, RTE, SST2) (Wang et al., 2018), five tasks from the SuperGLUE benchmark (WiC, CB, COPA, MultiRC, BoolQ), and the IMDB movie reviews dataset (Maas et al., 2011). Details of the MTL15 benchmark are provided in Table 4. Following Wang et al. (2024), we use AfriSenti (Muhammad et al., 2023; Wang et al., 2023c), a multilingual sentiment analysis dataset covering 12 low-resource African languages, including 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). Additionally, to further evaluate the module pruning capability of MoCL-P, we include WikiAnn, a multilingual named entity recognition (NER) dataset that covers 176 languages. The long task sequence in WikiAnn provides an adequate testbed for evaluating the pruning functionality of MoCL-P. Due to space constraints, we do not list the names of the 176 languages and their corresponding abbreviations. The specific language information is available at https://huggingface.co/datasets/wikiann.

We use different task orders for each dataset to evaluate the robustness of continual learning methods against changing task orders. For the MTL15 and AfriSenti benchmarks, we follow the task orders used in prior works, while for the WikiAnn benchmarks, we generate three random task orders for evaluation. The task orders used are summarized in Table 6.

A.2 Experiment Details

In this section, we provide the implementation details for the experiments and a detailed description of the baseline methods used in this work.

A.2.1 Implementation Details

We use the AdamW optimizer (Loshchilov and Hutter, 2017) for all experiments. We choose the same maximum sequence length and prefix length as prior work (Razdaibiedina et al., 2022; Wang et al., 2023e). Table 5 provides detailed hyperparameter choices for MoCL-P across different datasets. The training was performed on Nvidia A100 GPUs.[7]

A.2.2 Baseline Methods

In Section 6, we evaluate MoCL-P and prior continual learning methods on different benchmark datasets. Here, we provide a more detailed description of the baseline methods used in this work.

ProgPrompt (Razdaibiedina et al., 2022): A parameter isolation-based continual learning method that assigns task-specific parameters to avoid catastrophic forgetting. During continual learning, ProgPrompt progressively concatenates all task-specific modules to encourage forward transfer.

EPI (Wang et al., 2023e): A parameter isolation-based method applicable to the class-incremental learning setting (CIL), where task identities are not given during inference. EPI introduces a non-parametric task identification module that identifies tasks during testing. Given reliable task identification, the CIL performance of EPI could be comparable to TIL, where the ground truth task identities are given during inference.

O-LoRA (Wang et al., 2023d): A parameter isolation-based method that learns tasks in different low-rank vector spaces that are kept orthogonal to each other to minimize interference. It mitigates catastrophic forgetting by constraining the gradient update of the current task to be orthogonal to the gradient space of past tasks. However, the orthogonality of the gradient subspace for individual tasks also limits knowledge transfer between tasks.

MoCL (Wang et al., 2024): Introduces a modular and compositional continual learning framework to compose the new module with existing ones based on task module matching. This compositional strategy enables effective knowledge transfer by considering task interaction.

As discussed in Section 2, we build on the idea of MoCL (Wang et al., 2024) by utilizing task representations for module composition, ensuring that the model effectively reuses relevant knowledge from previous tasks. Beyond that, we introduce an adaptive pruning strategy to keep the language model lightweight and effective throughout the continual learning process, making it scalable for continual learning scenarios with long task sequences.

[7] All experiments ran on a carbon-neutral GPU cluster.

Dataset name	Category	Task	Domain
Yelp	MTL5	sentiment analysis	Yelp reviews
Amazon	MTL5	sentiment analysis	Amazon reviews
DBpedia	MTL5	topic classification	Wikipedia
Yahoo	MTL5	QA	Yahoo Q&A
AG News	MTL5	topic classification	news
MNLI	GLUE	NLI	various
QQP	GLUE	paraphrase detection	Quora
RTE	GLUE	NLI	news, Wikipedia
SST2	GLUE	sentiment analysis	movie reviews
WiC	SuperGLUE	word sense disambiguation	lexical databases
CB	SuperGLUE	NLI	various
COPA	SuperGLUE	QA	blogs, encyclopedia
BoolQ	SuperGLUE	boolean QA	Wikipedia
MultiRC	SuperGLUE	QA	various
IMDB	Other	sentiment analysis	movie reviews

Table 4: The details of 15 datasets used in the MTL15 benchmark. NLI denotes natural language inference, and QA denotes questions and answers task.

	Hyperparameters		
	AfriSenti-AfroXLMR	WikiAnn-BERT	MTL15-T5
Epochs	40	5	40
Early stop patience	5	N/A	5
Batch size	8	32	8
Learning rate	2e-4	1e-3	5e-2
Max. sequence length	256	128	512
Prefix length	16	8	10

Table 5: Hyperparameters used in this work across different CL experiments.

Table 6: The different orders of task sequences used for continual learning experiments.

Dataset	Order	Model	Task Sequence
AfriSenti	1	AfroXLMR	am → dz → ha → ig → kr → ma → pcm → pt → sw → ts → twi → yo
	2	AfroXLMR	ma → pcm → kr → pt → ig → sw → ha → ts → dz → twi → am → yo
	3	AfroXLMR	am → dz → ha → ma → ig → kr → sw → ts → twi → yo → pcm → pt
MTL15	1	T5	mnli → cb → wic → copa → qqp → boolq → rte → imdb → yelp → amazon → sst2 → dbpedia → ag → multirc → yahoo
	2	T5	multirc → boolq → wic → mnli → cb → copa → qqp → rte → imdb → sst2 → dbpedia → ag → yelp → amazon → yahoo
	3	T5	yelp → amazon → mnli → cb → copa → qqp → rte → imdb → sst2 → dbpedia → ag → yahoo → multirc → boolq → wic
WikiAnn	1	BERT	ga → fi → sco → bs → co → pnb → eu → vls → os → de → hy → mwl → ca → or → wa → rw → simple → tl → crh → lij → min → ko → scn → an → mk → hi → ug → ext → sl → sw → nap → et → wuu → uz → mzn → ast → jv → su → ilo → csb → cdo → tk → ckb → lv → ur → th → am → kn → pms → ba → tt → pl → vec → ru → cs → ne → bn → es → fy → fiu-vro → bo → mt → fr → mr → nn → bar → ang → no → fo → el → qu → fa → eml → kk → tr → pt → km → dv → hsb → rm → ta → fur → war → frr → ps → io → da → zh-yue → ms → cv → diq → mn → lb → cy → sa → ig → oc → hu → arc → ln → ku → hr → nds → az → ar → ce → lt → zea → it → zh-classical → be-x-old → mi → ia → is → la → sv → nl → gd → pa → xmf → ksh → zh-min-nan → lmo → tg → sh → eo → zh → te → he → vep → as → yi → cbk-zam → yo → ro → ace → id → jbo → nov → bg → map-bms → be → sr → sah → ml → my → vo → so → gu → br → gl → ka → li → pdc → ky → bat-smg → als → mg → szl → gn → ceb → vi → sq → mhr → ay → en → bh → uk → gan → sk → si → hak → af → ja → arz → sd

Continued on next page

Table 6 – continued from previous page

Dataset	Order	Model	Task Sequence
	2	BERT	wuu → cy → mwl → eu → gn → scn → ka → pdc → it → ro → pnb → ig → tl → sah → is → ga → ml → wa → vo → simple → hr → dv → mn → csb → sl → gl → fy → bn → tg → fr → th → vls → arz → zh-classical → ln → tr → su → min → si → ur → sr → et → eo → sh → li → fiu-vro → rw → no → mg → mr → oc → nap → yi → pa → lt → ug → co → tt → sv → uk → so → ext → ky → ru → kk → sa → la → el → hsb → be-x-old → bg → pt → bh → br → mt → ne → id → te → cv → fo → cdo → bs → lij → sw → he → ceb → hak → es → kn → mk → am → or → ms → az → als → my → ce → os → ca → tk → diq → zh → fi → jbo → mhr → ay → pms → rm → zea → en → zh-yue → sco → ang → bo → ar → ia → zh-min-nan → ckb → fa → crh → as → yo → szl → fur → hi → eml → mi → lb → de → bat-smg → uz → lv → nov → ast → cs → hy → sk → sq → be → xmf → af → ps → qu → da → ja → vep → ku → mzn → nl → vec → map-bms → ace → io → gu → bar → ilo → km → arc → cbk-zam → pl → ksh → war → gd → ba → lmo → gan → ko → an → frr → vi → hu → jv → sd → nds → nn → tas
	3	BERT	tl → sah → ckb → qu → az → ast → mr → eo → wa → zh-classical → fiu-vro → eu → nl → map-bms → id → szl → mi → io → lt → war → my → bat-smg → jv → en → zh-min-nan → sh → su → frr → am → hu → hy → zh → ps → hi → tg → pl → nov → dv → min → jbo → diq → ksh → gn → vec → nds → lij → pdc → os → rw → als → sq → fi → da → sr → ru → uz → fr → scn → tt → bh → bn → mwl → et → hsb → kn → rm → nn → mhr → bg → sd → ko → la → ka → de → he → pt → cs → hr → tk → cy → co → or → csb → bar → mt → vo → oc → simple → ml → bs → km → sk → ang → br → xmf → ay → zea → ln → sco → ku → ilo → lv → mzn → zh-yue → gan → ta → gl → ca → hak → mg → ne → ur → cbk-zam → uk → mn → fy → ba → nap → kk → yo → tr → so → fo → ug → ace → fur → pa → lmo → it → be-x-old → sa → arc → ig → lb → ms → th → cv → arz → bo → el → eml → gd → pnb → cdo → ky → af → vls → be → ga → es → yi → si → ext → gu → mk → ja → is → no → ceb → ro → sv → ar → an → te → sl → sw → wuu → pms → fa → vi → as → ce → vep → li → ia → crh