Rehearsal-Free Modular and Compositional Continual Learning for Language Models

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

Abstract

Continual learning aims at incrementally acquiring new knowledge while not forgetting existing knowledge. To overcome catastrophic forgetting, methods are either rehearsal-based, i.e., store data examples from previous tasks for data replay, or isolate parameters dedicated to each task. However, rehearsal-based methods raise privacy and memory issues, and parameter-isolation continual learning does not consider interaction between tasks, thus hindering knowledge transfer. In this work, we propose MoCL, a rehearsal-free **Mo**dular and **C**ompositional Continual **L**earning framework which continually adds new modules to language models and composes them with existing modules. Experiments on various benchmarks show that MoCL outperforms state of the art and effectively facilitates knowledge transfer.

1 Introduction

To effectively deploy machine learning (ML) models in real-world settings, they need to adopt *continual learning* (CL), i.e., incrementally acquire, update and accumulate knowledge to evolve continually and stay effective over time (Chen and Liu, 2018). However, CL often suffers from *catastrophic forgetting* (McCloskey and Cohen, 1989): The knowledge learned at early stages of training is overwritten by subsequent model updates.

A commonly used strategy to mitigate catastrophic forgetting is to store training samples from prior tasks along the continual learning process and train the model jointly with samples from prior and current tasks (*rehearsal*) (Rebuffi et al., 2017). However, training samples of prior tasks are not always available due to storage or privacy constraints (Wang et al., 2023a).

Another line of work allocates task-specific parameters to overcome catastrophic forgetting, often referred to as *parameter isolation-based* CL. Although inter-task interference leads to catastrophic forgetting (Wang et al., 2023a), knowledge transfer across tasks could be promising. However, those approaches do not enable effective knowledge transfer. Recent parameter isolation-based methods either separately train task-specific modules, completely excluding knowledge transfer (Wang et al., 2023e), or progressively concatenate all previous task-specific modules with the current task module (Razdaibiedina et al., 2022), without considering if the interaction between tasks is "positive" (knowledge transfer boosting performance) or "negative" (knowledge interference hurting performance).

To address these challenges, we introduce MoCL, a **Mo**dular and **C**ompositional Continual **L**earning framework for language models. MoCL avoids catastrophic forgetting without storing additional data and facilitates effective knowledge transfer via module composition. Specifically, MoCL allocates task-specific parameters using parameter-efficient fine-tuning (PEFT) modules.[1] During training, MoCL continually adds new task-specific modules to language models. To avoid catastrophic forgetting, the task-specific module is frozen once the training on the respective task is finished. Additionally, MoCL facilitates knowledge transfer across tasks by composing existing and new modules based on task matching weights while learning the new task. In our evaluation on *near-domain* and *far-domain* continual learning benchmarks, MoCL outperforms state-of-the-art methods under the task-incremental learning setting where the task identities are available during testing. It further demonstrates strong abilities to transfer knowledge of previous tasks to the new tasks. Furthermore, the task matching strategy of MoCL enables task composition during testing. As a result, MoCL ef-

[1]We use two PEFT methods, prefix-tuning (Li and Liang, 2021) and LoRA (Hu et al., 2021) in this work to be consistent with prior works (see Section 5.2 for details). Other PEFT methods, such as Adapter (Houlsby et al., 2019), can also be combined with MoCL in general. We leave such exploration for future work.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers), pages 469–480

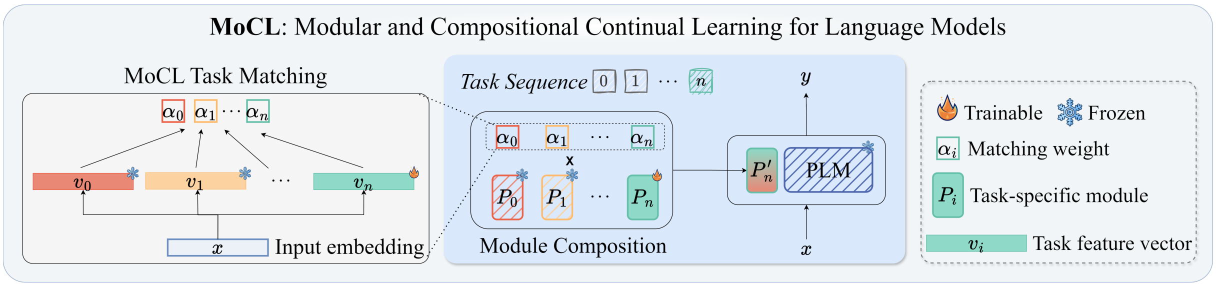

Figure 1: Overview of the MoCL framework for continual learning. MoCL continually adds new modules to language models and composes existing and new modules based on task matching weights for learning the new task.

fectively addresses the continual learning problem in the challenging class-incremental setting where task identities are not provided during testing.

The code base for MoCL is available online.[2]

2 Related Work

In line with previous work (De Lange et al., 2021; Ke and Liu, 2022; Wang et al., 2023a), we group CL strategies into three categories. (i) *Regularization*-based methods add explicit regularization terms to preserve the knowledge of previous tasks (Li and Hoiem, 2017; Kirkpatrick et al., 2017; Aljundi et al., 2018). As regularizing knowledge tends to have suboptimal performance, it is often used in combination with other methods. (ii) *Rehearsal*-based methods address catastrophic forgetting by saving old training samples in a memory buffer (Rebuffi et al., 2017; Rolnick et al., 2019; Zhang et al., 2022a), or training generative models to provide pseudo samples of previous tasks (Shin et al., 2017; Su et al., 2019) for future rehearsal. (iii) *Parameter isolation*-based methods assign isolated parameters dedicated to each task along the CL process to prevent interference between tasks (Madotto et al., 2020; Zhang et al., 2022b; Razdaibiedina et al., 2022; Wang et al., 2023e,d).

Since rehearsal-based methods raise memory and data privacy issues, we focus on rehearsal-free CL methods. MoCL falls into the category of parameter isolation-based continual learning, i.e., we allocate task-specific parameters to avoid knowledge interference. In contrast to related work, we additionally encourage knowledge transfer considering the relatedness across tasks.

3 Continual Learning Basics / Notation

In this work, we focus on continual learning (CL) on a sequence of text classification tasks. Specifically, we denote the sequence of tasks as $\{T_1, \ldots, T_N\}$. Each task T_n contains a set of input samples $\{(x_n^i, y_n^i)\}$, where 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. A CL model aims to solve the series of tasks which arrive sequentially. The overarching goal is to optimize the model's average performance across all tasks after learning them in the sequence. As we focus on rehearsal-free continual learning, data from earlier tasks is not available when training later tasks, i.e., our model does not suffer from the aforementioned shortcomings of rehearsal-based methods, such as memory issues.

While in many benchmark settings, the task identity n is provided, it is not a realistic assumption that task identities are available in real-world setups. Thus, we consider two setups: task-incremental learning (TIL) and class-incremental learning (CIL). In TIL, the task identities are available in both training and testing. In CIL, the task identities are only provided during training.[3]

4 Method

We propose MoCL, a novel CL approach for language models to tackle catastrophic forgetting and enhance knowledge transfer at the same time.

Avoiding Catastrophic Forgetting. We utilize two representative PEFT methods, prefix-tuning (Li and Liang, 2021) and LoRA (Hu et al., 2021) for allocating task-specific parameters to LMs, avoiding catastrophic forgetting without storing data samples. In particular, MoCL adds a set of trainable PEFT parameters (prefix or LoRA) to the frozen pretrained language model (PLM) for downstream task fine-tuning. Instead of updating the whole

[2] https://github.com/boschresearch/MoCL-NAACL-2024

[3] For better readability, we also refer to the domain-incremental learning (DIL), where tasks have the same label space but different input distributions, with and without test-time task identities as CIL and TIL, respectively; see Appendix A.2 for a more rigorous definition.

model, only a small number of the PEFT parameters are trained. As illustrated in Figure 1, MoCL optimizes the task-specific modules and keeps the PLM frozen. For each task T_n in the sequence, we initialize a trainable module P_n for fine-tuning. After the training on one task is finished, the corresponding PEFT parameters are frozen to preserve the task-specific knowledge in the following training process, thus avoiding catastrophic forgetting.

Enabling Knowledge Transfer. MoCL introduces task feature vectors for task matching and composes old and new modules for learning. This composition strategy facilitates effective knowledge transfer, which is often ignored by prior work. In particular, while learning on T_n, the previously acquired knowledge, which is encoded in the respective PEFT module $(P_1, \ldots, P_{n-1})$, is reused via a weighted summation, denoted as $P'_n = \sum_{k=1}^{n} \alpha_k P_k$. Here, P_k is the module specific to the k^{th} task and α_k is the weight determining the contribution of P_k for new task learning. We detail its computation below. Finally, the composed module P'_n is combined with the PLM, consisting of all the module components up to the current task.

To calculate the contribution weights α_k of each task-specific module, we introduce trainable task feature vectors $V \in \mathbb{R}^{N \times D}$ to capture salient features of tasks in the CL sequence. Note that each task-specific vector $v \in \mathbb{R}^D$ has the same dimension as the input embeddings $x_n \in \mathbb{R}^D$ (i.e., the embeddings from the PLM encoder). Then, we calculate the cosine similarity between the input embeddings x_n and feature vectors up to the current n^{th} task $V[:n]$ as task matching scores $\alpha[:n] = \cos(x_n, V[:n])$.

Training and Inference. The training objective for the n^{th} task is to find the PEFT module P_n and the task feature vector v_n that minimize the cross-entropy loss of training examples, and, at the same time, maximize the cosine similarity between v_n and the corresponding task input embeddings x_n:

$$\min_{P_n, v_n} - \sum_{x_n, y_n} \log p(y_n | x_n, P'_n, \theta) - \sum_{x_n} \cos(x_n, v_n) \quad (1)$$

During inference, as the task identities are available in the TIL setting, we directly select the task-specific module for inference. In the CIL setting, we use the matching scores between task inputs and the feature vectors for module composition. The resulting module is combined with the PLM for inference.

5 Experimental Setup

In this section, we describe our experimental setup.

5.1 Datasets

Dataset	Class	Task Type	Domain
AGNews	4	Topic classification	News
Yelp	5	Sentiment anlysis	Yelp reviews
Amazon	5	Sentiment anlysis	Amazon reviews
DBPedia	14	Topic classification	Wikipedia
Yahoo	10	Q&A	Yahoo Q&A

Table 1: Details of the MTL5 dataset.

Following Wang et al. (2023e), we distinguish benchmarks according to the domain similarity of tasks. As *near-domain* benchmarks, we use the Web-of-Science (WOS) document classification dataset (Kowsari et al., 2017) consisting of 7 tasks, and AfriSenti (Muhammad et al., 2023), a multilingual sentiment analysis dataset with 12 African languages. As *far-domain* benchmark, we use the widely adopted MTL5 dataset (de Masson D'Autume et al., 2019), including 5 text classification tasks. We summarize the details of MTL5 in Table 1. We adopt the same multiple task orders as the prior works for evaluation. Detailed task information is provided in Appendix A.1.

5.2 Training Details

We adopt four LMs for these datasets in line with prior works (Razdaibiedina et al., 2022; Wang et al., 2023d,e). We use encoder-based models for WOS, AfriSenti and MTL5 datasets (BERT (Devlin et al., 2018), AfroXLMR (Alabi et al., 2022) and BERT, respectively), the encoder-decoder T5 model (Raffel et al., 2020) as well as the decoder-based Llama 2-7B model (Touvron et al., 2023) for MTL5 under the few-shot setting. For all models except Llama 2,

Method	WOS	AfriSenti Orders			
		AVG	1	2	3
Sequential FT-F	47.15	6.17	5.62	6.52	6.30
Sequential FT-P	53.86	49.10	50.05	49.74	47.53
Per-task FT	82.78	52.41	52.41	52.41	52.41
ProgPrompt	89.93	49.07	50.16	46.74	50.30
EPI	77.83	43.10	41.49	42.65	45.16
MoCL (Ours)	**90.59**	**56.77**	**57.05**	**56.52**	**56.74**

Table 2: TIL results on near-domain WOS and AfriSenti datasets. MoCL outperforms existing continual learning methods on both datasets, suggesting MoCL effectively facilitates knowledge transfer across near-domain tasks.

we use prefix-tuning as the task-specific modules, and LoRA as the task modules on Llama 2. All design choices are kept consistent with previous works to ensure a fair comparison. The reported results are the average performance after training on all tasks consecutively. All results are averaged over three random seeds. The detailed experimental settings are provided in Appendix A.4.1.

5.3 Baselines

To compare different CL methods, we include the following baselines[4]: Sequential fine-tuning continuously fine-tunes the language model on the task sequence: **Sequential FT-F** means all model parameters are updated (fully fine-tuning),[5] while **Sequential FT-P** only fine-tunes the PEFT parameters; **Per-task FT** trains a separate PEFT module for each task; and the parameter isolation-based methods **ProgPrompt** (Razdaibiedina et al., 2022), **EPI** (Wang et al., 2023e) and O-LoRA (Wang et al., 2023d). A detailed description of these methods can be found in Appendix A.3.1.

6 Experimental Results

In this section, we discuss our experimental results.

6.1 MoCL for Task-Incremental Learning

Near-domain. As shown in Table 2, MoCL outperforms state-of-the-art methods on both benchmarks. It is 7.81 and 4.36 points better than training each task with an individual model (per-task FT), indicating it realizes effective knowledge transfer.

Since EPI consists of task identification and per-task fine-tuning, its performance depends on the task identification accuracy. While it achieves comparable results with per-task fine-tuning on WOS, the performance degrades on AfriSenti, where different languages could be harder to differentiate.

While MoCL achieves comparable results to ProgPrompt on WOS (0.66 percentage points better), the performance gap on AfriSenti is considerably higher (7.7 points better). We assume this is due to the suboptimal knowledge transfer of ProgPrompt, which we will analyze in Section 7.1.

Far-domain. Table 3 provides the results on MTL5 using BERT, T5 and Llama 2 models. MoCL again outperforms other CL methods in both cases across different task orders. Its advantage over per-task fine-tuning is less pronounced, which is due to the fact that far-domain tasks share weaker similarities.

6.2 MoCL for Class-Incremental Learning

Table 4 presents the class-incremental results. We compare MoCL only to EPI as they are the only two rehearsal-free CL methods applicable to this setting. Unlike EPI, our model has no explicit task identification component. Nevertheless, it still achieves better or competitive results.

7 Analysis

In this section, we analyze MoCL's forward transfer capability and its matching weights distribution.

7.1 Forward Transfer Analysis

We calculate the forward transfer scores (FWT) (Wang et al., 2023a) of MoCL and ProgPrompt in the TIL setting (see Table 5).[6] The FWT metric evaluates the average influence of all previous tasks on the current task:

$$\mathrm{FWT} = \frac{1}{N-1}\sum_{j=2}^{N}(a_{i,i} - \tilde{a}_i), \qquad (2)$$

where N is the number of tasks in the continual learning sequence, $a_{i,i}$ denotes the performance evaluated on the i-th task after incremental learning on the first i tasks, $\tilde{a}_i$ is the task performance of a randomly initialized reference model trained with dataset D_i. The results show that ProgPrompt suffers from catastrophic forgetting on AfriSenti ($\mathrm{FWT} < 0$) and explain the performance gap in Table 2. We assume the reason is negative interference between some of the languages, as observed in Wang et al. (2023c). ProgPrompt suffers from such interference as it concatenates all previous task-specific modules with the current module, without considering task interaction. In contrast, MoCL composes task modules via task matching, thus avoiding negative interference between tasks while exploiting similarities for knowledge transfer.

On the far-domain MTL5 dataset, MoCL still achieves higher scores than ProgPrompt. This suggests that our approach is better at transferring knowledge on various benchmarks, even with different levels of task similarities.

[4]For consistency, we include the results of baseline methods compatible with multiple base models used in this work. Results of other baselines which are specifically designed for certain LMs can be found in Appendix A.4.

[5]We did not run the Sequential FT-F experiments on Llama 2 because of the computational overhead and its poor performance in other experimental setups.

[6]As mentioned in 6.1, EPI consists of task identification and per-task FT. Thus, with given task IDs, EPI is identical to per-task FT, thus, includes no knowledge transfer ($\mathrm{FWT} = 0$).

Method	MTL5 (BERT) Orders					MTL5 (T5) Orders			
	AVG	1	2	3	4	AVG	1	2	3
Sequential FT-F	14.8	27.8	26.7	4.5	18.4	28.5	18.9	24.9	41.7
Sequential FT-P	66.5	66.4	65.7	65.4	68.5	27.2	24.6	30.3	25.0
Per-task FT	79.0	79.0	79.0	79.0	79.0	75.1	75.1	75.1	75.1
ProgPrompt $^\diamond$	77.9	78.0	77.9	77.9	77.9	75.1	75.0	75.0	75.1
EPI †	77.3	77.4	77.3	77.2	77.4	56.4	49.7	54.1	65.3
MoCL (Ours)	**79.4**	**79.3**	**79.6**	**79.2**	**79.4**	**75.9**	**75.6**	**75.4**	**76.7**

Method	MTL5 (Llama 2) Orders			
	AVG	1	2	3
Sequential FT-P	26.7	28.8	27.4	26.6
Per-task FT	76.6	76.6	76.6	76.6
EPI	48.4	48.1	48.0	49.0
O-LoRA ‡	76.1	76.8	75.7	75.7
MoCL (Ours)	**78.2**	**78.4**	**77.7**	**78.4**

Table 3: TIL results on the far-domain MTL5 dataset with BERT, T5 and Llama 2 as the base model. The superscripts $^\diamond$, † and ‡ indicate that results are taken from Razdaibiedina et al. (2022), Wang et al. (2023e) and Wang et al. (2023d), respectively.[7]

CIL	Datasets			
	WOS	AfriSenti	MTL5-BERT	MTL5-T5
EPI	77.83	43.10	**77.3**	56.4
MoCL (Ours)	**79.23**	**45.62**	74.1	**56.8**

Table 4: CIL results. We only compare MoCL and EPI as they are the only two rehearsal-free approaches that support this challenging task setting.

FWT	Datasets			
	WOS	AfriSenti	MTL5-BERT	MTL5-T5
ProgPrompt	8.4	-3.5	-0.3	0
MoCL (Ours)	**8.9**	**4.8**	**0.3**	**0.3**

Table 5: Forward transfer (FWT) score comparison between ProgPrompt and MoCL across datasets.

7.2 Task Matching Weights Visualization

In Figure 2, we visualize the task matching weight distribution of MoCL on the AfriSenti dataset[8] exemplarily with task order 2 (see Table 6) under the TIL setting. As MoCL performs per-instance task matching and module composition, we average the weights across all examples from a given task (i.e., language). As introduced in Section 4, while learning on the n^{th} task, we calculate the cosine similarity between the input embeddings and task feature vectors up to the current n^{th} task. Therefore, the heatmap only has the lower left part.

Certain task-specific modules, such as *ma*, *kr*, and *ha*, exhibit utility across a wide range of other tasks, while others, like *pcm*, demonstrate utility exclusivity in their respective tasks. Moreover, we observe that there is a pronounced sparsity in the learned weight distributions. Our task matching paradigm can be considered a mixture-of-experts strategy where we use task-specific experts as the mixture components. Such a sparsity suggests that we can potentially reduce the number of experts, instead of using experts specific to each task. This can be an interesting direction for future work.

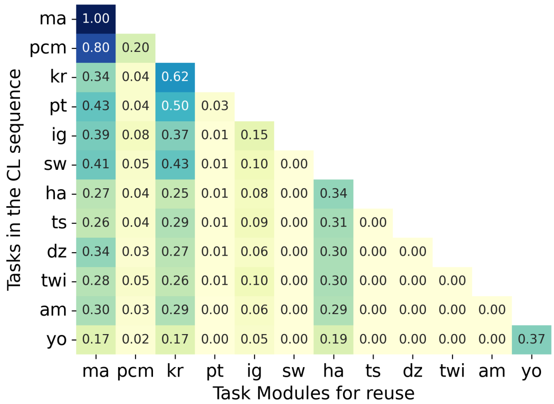

Figure 2: Visualization on the task matching weights of MoCL on the AfriSenti dataset (Task order 2). The heatmap entries quantify the extent of contribution from each task-specific module (denoted on the x-axis) to the subsequent tasks (represented on the y-axis).

8 Conclusion

In this paper, we introduced MoCL, a modular and compositional continual learning framework for language models, effectively addressing the critical challenges of catastrophic forgetting and knowledge transfer in continual learning. Our broad evaluations across various benchmarks demonstrated MoCL's superior performance compared to existing state-of-the-art methods and showed its proficiency in knowledge transfer from previous tasks.

[7] Among these baseline methods, ProgPrompt is only applicable with prefix-tuning as the PEFT module. O-LoRA is specifically designed for LoRA as the PEFT module.

[8] We provide weights visualization on other datasets in Appendix A.4

Limitations

One limitation of our work is the scope of evaluation. While MoCL is generally applicable to a wide range of tasks, we primarily focus on text classification tasks following prior work. Further experiments with other types of NLP tasks, especially generative tasks is left as a future work direction.

Besides, the continual learning datasets we study in this work include at most 12 tasks in a sequence. As the continual learning sequence scales to dozens or hundreds of tasks, continually initializing a new PEFT module for each task would largely increase the computational and storage cost. In Section 7.2, we observe that the learned weight distribution is notably sparse, suggesting that we could reduce the number of task modules instead of assigning a specific module for each task. It would be an interesting future work direction to utilize some module pruning strategies for more efficient continual learning.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive feedback, which was very valuable for improving our work. This work was partially supported by Deutsche Forschungsgemeinschaft (project SCHU 2246/14-1).

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. *arXiv preprint arXiv:2204.06487*.

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.

Zhiyuan Chen and Bing Liu. 2018. Continual learning and catastrophic forgetting. In *Lifelong Machine Learning*, pages 55–75. Springer.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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*.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2736–2746, Online. Association for Computational Linguistics.

Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*.

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.

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. IEEE.

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*.

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*.

Chengwei Qin and Shafiq Joty. 2021. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. *arXiv preprint arXiv:2110.07298*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

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.

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

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 Schütze. 2023c. Nlnde at semeval-2023 task 12: Adaptive pretraining and source language selection for low-resource multilingual sentiment analysis. *arXiv preprint arXiv:2305.00090*.

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.

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*.

A Appendix

A.1 Dataset Information

Here we give detailed information of the datasets we use with in this work. For *near-domain* benchmarks, we use Web-of-Science (WOS) and AfriSenti. WOS is originally a hierarchical document classification datasets which collects published papers in 7 different domains, which are biochemistry, civil engineering, computer science, electrical engineering, medical science, mechanical engineering and psychology. These domains corresponds to 7 high-level classes for document classification, and there are several low-level subclasses under each high-level class. Following Wang et al. (2023e), we organize 7 continual learning tasks according to these high-level classes. AfriSenti is a multilingual sentiment analysis dataset which covers 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).

For *far-domain* benchmarks, we adopt the commonly used MTL5 dataset, consisting of 5 text classification tasks. Detailed task information is given in Table 1. We experiment with BERT-base and T5-large models on this dataset in line with prior work (Razdaibiedina et al., 2022). For BERT-based experiments, we uses the same train and test sets following prior work such as ProgPrompt (Razdaibiedina et al., 2022) and EPI (Wang et al., 2023e), consisting of 115,000 training and 7,600 text samples for each task. For T5- and Llama 2-based experiments, 4 out of these 5 tasks (except Yelp) are used in line with Razdaibiedina et al. (2022) and (Wang et al., 2023d), with 16 samples per task for training and the test sets are unchanged.

Following prior work, we report F1 score on the AfriSenti dataset (Muhammad et al., 2023; Wang et al., 2023b) and accuracy on WOS and MTL5 datasets (de Masson D'Autume et al., 2019; Razdaibiedina et al., 2022; Wang et al., 2023e). We use different task orders for each dataset to evaluate the robustness of continual learning methods against changing task orders. The task orders used are summarzied in Table 6.

A.2 Continual Learning Setting Details

Beyond the general formulation as introduced in Section 3, continual learning can be categorized into several detailed settings,[9] according to the distinction between incremental data batches and task identity availability. *Task-incremental learning* (TIL) refers to the scenario where the tasks have disjoint label space. Task identities are provided in both training and testing. This is the most studied continual learning scenario and also the easiest case of continual learning tasks.

Class-incremental learning (CIL) is a more challenging continual learning scenario where the task identities are not available during testing. The tasks still have disjoint label space and task identities are available during training.

Domain-incremental learning (DIL) assumes the class labels are the same across all tasks and the inputs are from different domains. Whether task identities are given during testing or not, it all belongs to this category. Strictly speaking, the AfriSenti benchmark used in this work belongs to the DIL category. In this multilingual sentiment analysis dataset, the data of different tasks (languages) is considered to have different input distributions, while the label space is shared across tasks (languages). In this work, we aim to evaluate MoCL in settings where the task identities are provided and are not provided during testing. We also consider the evaluation setting on AfriSenti as task-incremental learning and class-incremental learning, respectively. In our experiments, we assume tasks have disjoint label spaces, i.e., their classification heads are different. In this way, we use the AfriSenti benchmark for TIL and CIL evaluation as well.

A.3 Experimental Setup Details

In this section, we give more detailed information about the baseline methods we used in this work and the implementation details for experiments.

A.3.1 Baseline Methods

In Sections 6 and A.4, we evaluate MoCL and prior continual learning methods on different benchmark datasets. Here we give a more detailed description of the baseline methods used in this work.

MBPA++ (de Masson D'Autume et al., 2019): introduces an episodic memory model that performs sparse experience replay and local adaptation to continuously learn and reuse previously acquired knowledge.

[9]We focus on some commonly studied continual learning settings here, for a more comprehensive categorization of continual learning settings please refer to (Wang et al., 2023a).

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
WOS	1	BERT	1 → 2 → 3 → 4 → 5 → 6 → 7
MTL5	1	BERT	ag → yelp → amazon → yahoo → db
	2	BERT	yelp → yahoo → amazon → db → agnews
	3	BERT	db → yahoo → ag → amazon → yelp
	4	BERT	yelp → ag → db → amazon → yahoo
MTL5	1	T5, Llama 2	db → amazon → yahoo → ag
	2	T5, Llama 2	db → amazon → ag → yahoo
	3	T5, Llama 2	yahoo → amazon → ag → db

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

Method	RF	PE	CI	KT
EWC (Kirkpatrick et al., 2017)	✓			✓
MBPA++ (de Masson D'Autume et al., 2019)			✓	✓
IDBR (Huang et al., 2021)			✓	✓
LFPT5 (Qin and Joty, 2021)		✓		✓
ProgPrompt (Razdaibiedina et al., 2022)	✓	✓		✓
EPI (Wang et al., 2023e)	✓	✓	✓	
O-LoRA (Wang et al., 2023d)	✓	✓	✓	
MoCL (Ours)	✓	✓	✓	✓

Table 7: Comparison between MoCL and existing CL approaches. RF: rehearsal-free; PE: parameter-efficient; CI: applicable to class-incremental learning, KT: enabled knowledge transfer.

IDBR (Huang et al., 2021): disentangles the text embeddings into task-generic space and task-specific space and further regularizes them differently. It also leverages data replay and two auxiliary tasks for effective continual learning. Due to its architectural design, it is only applicable to encoder-based transformer models for classification tasks.

LFPT5 (Qin and Joty, 2021): a continual learning approach based on the T5 model. It continuously trains a soft prompt to solve the task and generate pseudo samples for data replay.

ProgPrompt (Razdaibiedina et al., 2022): a parameter isolation-based continual learning method which assigns task-specific parameters to avoid catastrophic forgetting. During continual learning, ProgPrompt progressively concatenates all task-specific modules to encourage forward transfer. Task identities are always required during training and testing.

EPI (Wang et al., 2023e): a parameter isolation-based method applicable to the class-incremental learning setting. EPI introduces a non-parametric task identification module that identifies tasks during testing. Given reliable task identification, the CIL performance could be comparable with TIL, where the ground truth task identities are given.

O-LoRA (Wang et al., 2023d): a parameter isolation-based method which 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 the past tasks. However, the orthogonality of the gradient subspace for individual tasks also limits knowledge transfer between tasks.

As discussed in the main paper, ProgPrompt and EPI are two closely related prior work to MoCL. ProgPrompt concatenates all previously learned parameters with the current learnable to encourage knowledge transfer while ignoring different levels of relatedness across tasks: There might be knowledge interference or transfer between different pairs of tasks. EPI focus on the class-incremental learning setting and the task-specific parameters are completely isolated, i.e., there is no knowledge transfer in their approach. In contrast, MoCL assigns different weights to previously learned task-specific modules based on the relatedness between tasks, therefore deftly balancing knowledge interference or transfer and leading to more effective knowledge transfer.

A.4 Additional Experimental Results

In this section, we give additional experimental results, including the additional baseline results, MoCL's per-task results on the three datasets, and the weight distribution on AfriSenti for module composition.

Method	MTL5 (BERT) Orders				
	AVG	1	2	3	4
Sequential FT-F	14.8	27.8	26.7	4.5	18.4
Sequential FT-P	66.5	66.4	65.7	65.4	68.5
Per-task FT	79.0	79.0	79.0	79.0	79.0
*MBPA++**	*70.6*	*70.8*	*70.9*	*70.2*	*70.7*
*IDBR**	*76.3*	*75.9*	*76.2*	*76.4*	*76.7*
ProgPrompt*	77.9	78.0	77.9	77.9	77.9
EPI*	77.3	77.4	77.3	77.2	77.4
MoCL (Ours)	**79.4**	**79.3**	**79.6**	**79.2**	**79.4**

Method	MTL5 (T5) Orders			
	AVG	1	2	3
Sequential FT-F	28.5	18.9	24.9	41.7
Sequential FT-P	27.2	24.6	30.3	25.0
Per-task FT	75.1	75.1	75.1	75.1
*LFPT5**	*52.7*	*47.6*	*52.6*	*57.9*
ProgPrompt*	75.1	75.0	75.0	75.1
EPI	56.4	49.7	54.1	65.3
MoCL (Ours)	**75.9**	**75.6**	**75.4**	**76.7**

Table 8: TIL results with additional baseline methods on far-domain MTL5 with BERT and T5 as the base model. * indicates that results are taken from the corresponding papers. MoCL still outperforms all existing work in both evaluation settings.

Additional baselines. In Section 5.3 and 6, we only include methods that are applicable across models for consistency reasons. In Table 8, we provide results with three additional continual learning methods, where IDBR (Huang et al., 2021) and MBPA++ (de Masson D'Autume et al., 2019) are BERT-based continual learning methods, while LFPT5 (Qin and Joty, 2021) is specifically designed for the T5 language model. In both evaluation settings, MoCL consistently shows better performance than prior work, demonstrating the effectiveness of our proposed approach.

Per-task results. From Table 9 to 12, we give the detailed per-task results on the aforementioned datasets under task-incremental learning and class-incremental learning settings.

WOS per-task results								
order 1	AVG	1	2	3	4	5	6	7
TIL	90.59	91.86	95.72	80.05	93.25	95.09	93.60	84.54
CIL	79.23	70.57	93.36	58.74	86.67	91.29	87.82	66.19

Table 9: Detailed per-task results on the WOS dataset under TIL and CIL settings.

Task matching weights visualization. In Section 7.2, we visualized the task matching weights produced by MoCL on the AfriSenti dataset (Figure 2). In Figures 3, 4 and 5, we provide the visualization

AfriSenti per-task results							
order1	AVG	am	dz	ha	ig	kr	ma
TIL	57.05	58.52	58.58	66.83	56.92	63.68	48.68
CIL	45.57	63.56	52.88	47.06	26.15	52.16	40.28
order1		pcm	pt	sw	ts	twi	yo
TIL		60.59	64.27	57.24	42.97	46.56	59.77
CIL		56.98	36.71	28.80	38.10	44.21	60.00
order2	AVG	ma	pcm	kr	pt	ig	sw
TIL	56.52	47.41	58.51	65.15	61.38	54.47	55.19
CIL	44.32	40.56	57.12	47.53	35.22	25.44	29.21
order2		ha	ts	dz	twi	am	yo
TIL		67.27	44.45	61.20	45.40	58.32	59.53
CIL		44.49	40.33	46.24	41.82	64.91	59.03
order3	AVG	am	dz	ha	ma	ig	kr
TIL	56.74	58.52	58.58	66.83	50.05	54.20	59.90
CIL	46.95	46.00	39.34	57.76	45.17	47.08	49.89
order3		sw	ts	twi	yo	pcm	pt
TIL		57.47	42.60	44.83	60.01	60.17	64.71
CIL		53.56	23.24	34.61	49.19	53.50	CIL

Table 10: Detailed per-task results on the AfriSenti dataset under TIL and CIL settings.

MTL5-BERT per-task results						
order1	AVG	agnews	yelp	amazon	yahoo	db
TIL	79.31	94.13	64.41	61.67	77.14	99.19
CIL	73.02	93.39	62.75	39.13	72.30	97.52
order2	AVG	yelp	amazon	yahoo	db	agnews
TIL	79.64	64.43	62.50	78.03	99.23	94.03
CIL	74.00	62.69	44.91	70.98	99.14	92.26
order3	AVG	db	yahoo	agnews	amazon	yelp
TIL	79.20	99.23	77.72	94.03	61.78	63.24
CIL	74.75	98.40	72.19	92.97	53.82	59.57
order4	AVG	yelp	agnews	db	amazon	yahoo
TIL	79.61	64.43	94.37	99.20	62.04	77.99
CIL	73.55	62.54	93.41	98.98	47.75	65.07

Table 11: Detailed per-task results on the MTL5 dataset using BERT as the base language model under TIL and CIL settings.

MTL5-T5 per-task results					
order1	AVG	db	amazon	yahoo	agnews
TIL	75.59	98.27	47.88	70.84	85.31
CIL	51.15	40.86	11.34	67.58	84.84
order2	AVG	db	amazon	agnews	yahoo
TIL	75.37	98.18	47.99	84.69	70.64
CIL	47.84	32.04	8.91	79.84	70.59
order3	AVG	yahoo	amazon	agnews	db
TIL	76.70	71.42	51.09	86.25	97.99
CIL	71.47	67.75	48.37	73.92	95.82

Table 12: Detailed per-task results on the MTL5 dataset using T5 as the base language model under TIL and CIL settings.

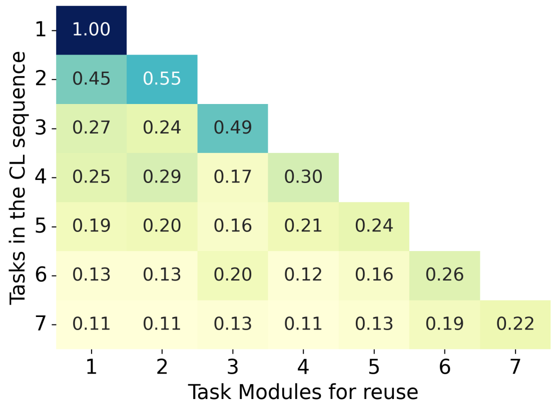

Figure 3: Visualization on the task matching weights of MoCL on the WOS dataset.

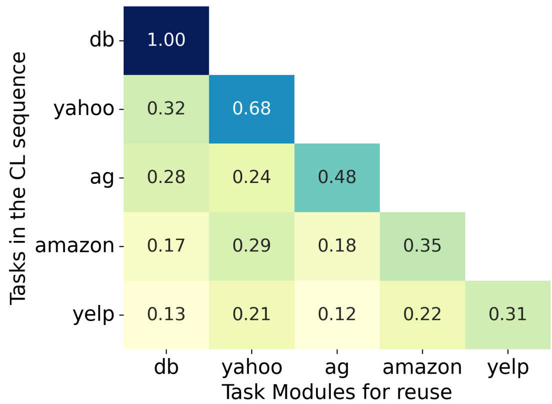

Figure 4: Visualization on the task matching weights of MoCL on the MTL5 dataset with BERT as the base model (Task order 3).

on the other datasets. We randomly pick one task order for each dataset for space reasons. As described in Section 7.2, the heatmap entries quantify the extent of contribution from each task-specific module (denoted on the x-axis) to the subsequent tasks (represented on the y-axis).

We observe a different distribution of weights on the two types of benchmarks, i.e., near-domain and far-domain. On the near-domain datasets, i.e., AfriSenti and WOS, as shown in Figure 2 and 3, the subsequent tasks tend to reuse modules of previous tasks. Whereas on the far-domain MTL5 dataset, Figure 4 and 5 show that the task-specific modules always have higher weights, i.e., the highest values on the diagonal of the heatmap. This is related to the nature of these benchmarks: The tasks from the near-domain benchmark are more related to each other, so there is a tendency for the tasks to reuse existing knowledge from previous modules.

Figure 5: Visualization on the task matching weights of MoCL on the MTL5 dataset with T5 as the base model (Task order 2).

While the tasks from the far-domain dataset are dissimilar, and thus the task-specific modules have higher weights.

A.4.1 Implementation Details

We use the AdamW optimizer (Loshchilov and Hutter, 2017) and the batch size of 8 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 13 gives detailed hyperparameter choices of MoCL across different datasets. The training was performed on Nvidia A100 GPUs.[10]

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

Hyperparameters	
WOS-BERT	
Epochs	40
Early stop patience	5
Learning rate	3e-2
Max. sequence len.	256
Prefix len.	16
AfriSenti-AfroXLMR	
Epochs	40
Early stop patience	5
Learning rate	2e-4
Max. sequence len.	128
Prefix len.	8
MTL5-BERT	
Epochs	40
Early stop patience	5
Learning rate	8e-4 (db), 1e-3 (yahoo) 2e-3 (others)
Max. sequence len.	256
Prefix len.	20
MTL5-T5	
Epochs	40
Early stop patience	5
Learning rate	2e-2 (yahoo, db) 5e-2 (others)
Max. sequence len.	512
Prefix len.	50
MTL5-Llama 2	
Epochs	40
Early stop patience	5
Learning rate	1e-3
Max. sequence len.	512
LoRA rank	4

Table 13: Hyperparameters used in this work across different CL experiments.