Multilingual Continual Learning Approaches for Text Classification

Karan Praharaj **Irina Matveeva**
Reveal
Chicago, IL
{kpraharaj,imatveeva}@revealdata.com

Abstract

Multilingual continual learning is important for models that are designed to be deployed over long periods of time and are required to be updated when new data becomes available. Such models are continually applied to new unseen data that can be in any of the supported languages. One challenge in this scenario is to ensure consistent performance of the model throughout the deployment lifecycle, beginning from the moment of first deployment. We empirically assess the strengths and shortcomings of some continual learning methods in a multilingual setting across two tasks.

1 Introduction

There is a substantial amount of research in continual learning that studies how large language models can be trained in multiple steps. Task-incremental learning (Kirkpatrick et al., 2017; Chaudhry et al., 2019; Lopez-Paz and Ranzato, 2017), class-incremental learning (Wu et al., 2019), domain-incremental learning (Wang et al., 2022) and language-incremental learning (Badola et al., 2022; Castellucci et al., 2021; M'hamdi et al., 2022; Praharaj and Matveeva, 2022) have been extensively studied (van de Ven and Tolias, 2019) in the realm of continual learning. The motivation is for the same model to be trained on various tasks/classes/domains/languages not only to avoid having individual models for each task, class, domain, or language but also to improve the overall model performance.

One of the challenges in continual learning is to counteract the tendency of large language models to "forget" previously learned information when trained on new data. In multilingual models, the performance of the model on languages fine-tuned in the past tends to decrease or can even result in catastrophic forgetting (Mccloskey and Cohen, 1989; French, 1999).

Task-incremental learning, class-incremental learning, and domain-incremental learning research typically focus on few-step continual learning, where the model is incrementally fine-tuned with a few tasks, classes, or domains. while language-incremental learning or multilingual continual learning offers an opportunity to do continual learning over dozens of steps. We will use the term "multilingual continual learning" in this paper.

This work addresses the performance of training multilingual models over many fine-tuning steps. We analyze the main existing approaches, provide a summary of their performance on two multilingual datasets and provide an analysis of the trade-offs for selecting one approach over another.

For this work, we develop a test scenario that is representative of the practical setting: we update multilingual classification models over a large number of steps with small amounts of training data, and the training data is in different languages.

We consider three dimensions in which the approaches differ: the amount of previously seen data that is used in fine-tuning steps, the training time, and the fine-tuning of data in different languages. The main contributions are as follows:

- We study different multilingual continual learning approaches on two datasets with a long sequence of training steps and provide an empirical analysis of the strengths and weaknesses of these approaches.
- We show that the performance on a two-class versus a multi-class dataset is very different. We also show that more research is needed for some approaches before they can be used for continual learning over long sequences of training data.
- We provide recommendations for different model deployment scenarios depending on

Proceedings of Recent Advances in Natural Language Processing, pages 864–870
Varna, Sep 4–6, 2023
https://doi.org/10.26615/978-954-452-092-2_093

the availability of resources.

2 Related Work

There are multiple approaches to language-incremental learning such as Praharaj and Matveeva (2022); Castellucci et al. (2021); Badola et al. (2022); Yang et al. (2021); M'hamdi et al. (2022); Pfeiffer et al. (2022). In this paper, we provide a comparison of the main methods in application to continual learning over many steps and discuss considerations for selecting one approach over another.

One group of existing studies considered how to construct the training data at each step: joining all training data from all previous fine-tuning steps vs using only the training data from the current step. Joint training of all languages (M'hamdi et al., 2022) or domains (Ozler et al., 2020) has been shown to work better than sequential training using only the current data. Since joint tuning does not comply with possible privacy constraints, we consider both approaches in our study, joint fine-tuning with all training data and sequential fine-tuning with only current training data.

For sequential fine-tuning, we consider Praharaj and Matveeva (2022). They present an analysis of sequential training with multilingual BERT (Devlin et al., 2019) and show that a combination of translation augmentation and specialized training methodology facilitates stable continual learning performance over many multilingual steps. We consider their approach for our study because they also used a large number of fine-tuning steps. We adapt their approach to generating long training sequences with some modifications.

Adapter-based methods such as Houlsby et al. (2019); Ke et al. (2021); Pfeiffer et al. (2020) were proposed as another methodology for robust continual learning. More recently, Pfeiffer et al. (2022) have proposed a parameter modularization approach in pre-training (X-MOD) that enables positive transfer between languages while also reducing negative interference between them. We include X-MOD as one of the approaches in our analysis.

Memory-based approaches such as Chaudhry et al. (2019); Lopez-Paz and Ranzato (2017); Scialom et al. (2022) have also been explored to mitigate forgetting. Such methods make use of an *episodic memory* or a cache that stores a subset of data from previous tasks. These examples are then used for training along with the current examples in the current optimization step. We don't consider them in our study. To represent approaches that assume access to the previous training data, we use joint fine-tuning.

3 Compared Approaches

We focus on four approaches in this survey. Joint training (*Joint*), joint-incremental training (*Joint-Inc*), sequential fine-tuning with a specialized optimization regime (*SeqFT-SO*) (Praharaj and Matveeva, 2022) and pre-trained adapters for individual languages (*Ada-SeqFT*) (Pfeiffer et al., 2022). Out of the four approaches we consider here, only *SeqFT-SO* was studied for continual multi-lingual learning. Therefore we don't have many related research results. We implemented each approach and used the best practice parameters from the literature for each of them. This section outlines how we implemented these approaches.

Joint training. (*Joint*) Training the base model on all data in the language sequence simultaneously in only one step. Previous literature on continual learning has shown that joint training outperforms most sequential training methods. This is a non-incremental baseline since there is no continual learning aspect here.

Joint-incremental training. (*Joint-Inc*) The model is trained incrementally, collecting data from each step. At each step, all previously available training data is combined for fine-tuning, and the base model is fine-tuned. This means that the training set size and the training time grow over time. This approach performed well on task incremental learning (M'hamdi et al., 2022). At the last step of the *Joint-Inc* training, all data is available, and so it will be the same as the *Joint* baseline.

Both approaches, *Joint* and *Joint-inc* require access to training data from previous steps. While combining training data leads to better results, it may not be possible to store data from previous steps due to privacy concerns. Both approaches combine data from all languages in each training step and don't handle each language individually.

Adapters. (*Ada-SeqFT* (Pfeiffer et al., 2022)) This approach uses language-specific modules called adapters during fine-tuning with an aim to

disentangle the linguistic component from the task information component so as to mitigate negative interference between languages. The training and inference cost remains constant regardless of the number of languages involved because only one module is used at a time. However, since a dedicated module is learned for each language, adding a new language results in an increase in the total number of parameters. We incorporate this method in a continual learning setting by adding a language adapter during training and inference of the designated train/test languages. For example, at a given step in the sequence, if the training data arrives in Spanish, we would "plug" in the pre-trained Spanish adapter during fine-tuning. At test time, we would use an English adapter for a test set in English, a Spanish adapter for a test set in Spanish, etc. As recommended by the authors, we freeze the adapter weights during fine-tuning and only update the weights shared by all languages. The *Ada-SeqFT* has specialized procedures for training data in each language, unlike all other approaches we consider here. We use adapters with sequential fine-tuning, which means for each training step, we use only the current training data. This approach can be used when data privacy is important.

Sequential Fine-tuning. (*SeqFT-SO* (Praharaj and Matveeva, 2022)) Sequential fine-tuning uses only the current training data in each fine-tuning step. Once the training data from a particular step is used to fine-tune the model, it has to be discarded. This approach can be used when the training data cannot be stored due to privacy considerations. Praharaj and Matveeva (2022) showed that with an appropriate training regime *SeqFT-SO* avoids catastrophic forgetting and allows the model to improve for 50 incremental fine-tuning steps. The difference to the *Joint-inc* approach is that here we don't store the additional training data after the incremental fine-tuning is done. The main difference to *Ada-SeqFT* is that here the same model is fine-tuned with all supported languages.

To summarize, for *Joint-Inc*, the training data increases in size after each step, and so the training time increases. *Ada-SeqFT* and *SeqFT-SO* use only the current training data to sequentially fine-tune the model so the training set size does not grow over time. *Joint-Inc* and *SeqFT-SO* use training data for all languages to fine-tune the full model, whereas *Ada-SeqFT* fine-tunes only the language-specific adapter at each fine-tuning step.

4 Experiments

4.1 Data

Datasets We use two datasets for the evaluation: sentiment classification MARC (Keung et al., 2020) and intent classification MTOP (Li et al., 2021). MARC is a multilingual dataset of customer reviews for various product categories. The MTOP dataset is an almost parallel task-oriented semantic parsing dataset for two tasks: intent classification and slot filling. We use the intent classification data for our evaluation.

For MARC, we transformed the multi-class data into binary class data by combining 4-star and 5-star reviews as positive sentiment reviews and 1-star and 2-star reviews as negative sentiment reviews. We use data from five languages for products in four categories, resulting in 60 possible language-category combinations that may occur in the sequence. The languages used are German, English, French, Chinese, and Japanese. The categories used are *apparel*, *home*, *musical instruments*, and *sports*. At each step, the training data is from a particular language-category combination. Though categories vary from step to step, the classification problem remains the same - sentiment classification.

The MTOP intent classification task has 117 classes across six languages. We use all languages included in MTOP, German, English, French, Spanish, Hindi, and Thai. We filter out any classes that do not feature in all six languages or do not have at least four examples in each language. This leaves us with 113 classes in the training set which span domains such as alarm (e.g., *SET_ALARM*, music (e.g., *PLAY_SONG*), messaging (e.g. *SEND_MESSAGE*), weather (e.g. *GET_WEATHER*), recipes (e.g. *GET_INFO_RECIPES*), etc. This is a more challenging task compared to MARC in terms of the number of classes. At each step, all classes are present in the training data. So from step to step, only the language of the data varies. This ensures that the learning is exclusively language-incremental and not class-incremental.

4.2 Experimental Setup

In our continual multilingual learning scenario, a pre-trained model is sequentially fine-tuned using training data in different languages over multiple

steps. For our experimentation, we assume that the set of training languages is fixed. We define this set of languages as $\mathcal{L} = \mathcal{L}_0, \mathcal{L}_1, ..., \mathcal{L}_K$. We also assume that in each step, the data is exclusively in one language.

Base Model We begin with a pre-trained multilingual model $\mathcal{M}_b$. For joint and sequential fine-tuning, we use mBERT (Devlin et al., 2019) as our base model. For X-MOD, the pre-trained weights made available are an extension for the XLM-RoBERTa (Conneau et al., 2020). The batch size used was 32 with a learning rate of $3e-5$. For sequential fine-tuning, we apply a layer-wise learning rate decay of 0.95. We train all runs (for all methods) over three epochs. For MARC, we set the maximum sequence length to 512 (for all methods), whereas for MTOP, we set the maximum sequence length to 128 (for all methods).

Intermediate pre-training Analogous to the *inception stage* followed by Badola et al. (2022), we first initialize the base model $\mathcal{M}_b$ by fine-tuning it on some task data on all expected languages. We call this process *intermediate pre-training* (IPT). This step could be thought of as setting up the model and endowing it with task knowledge before it is deployed and incrementally trained on new incoming data. We consider this IPT model $\mathcal{M}_0$ to be starting model for our continual fine-tuning steps.

Training sequence creation $\mathcal{M}_0$ is fine-tuned over multiple stages to create incremental versions $\mathcal{M}_i$ where $i = 0...N$. In each fine-tuning step the training data $\mathcal{D}_i$ is in a language $\mathcal{L}_j$, where $0 \leq j \leq K$. We randomly generate sequences of training data to simulate a sequential fine-tuning scenario using the method from Praharaj and Matveeva (2022). For the MARC dataset, the training data for each step comes from a language-category combination, and the classes of positive-negative sentiment remain the same. For the MTOP dataset, at each step, all classes are represented in the data, and only the language changes across steps. Table 1 shows the sequences of the training data for each step and for each dataset.

For each dataset, we generate three random sequences. For MARC, we train over 24 steps, and for MTOP we train over 20 steps. We define a *step* as one iteration of fine-tuning the weights of the model and then evaluating it on the test data. For *Joint-Seq* at each step, we train $\mathcal{M}_0$ with the combined training data.

We would like to point out that for *Joint-Inc*, we did not train each step in the sequence. We only trained it at the points that we show on the plots in Figure 1. We did this because the training set size and the training time increased. And since the performance trend seemed to be very clear, it did not make sense to use resources for that. For the other sequential approaches, we fine-tune the models at every step in the sequence.

Task	Seq.	Steps
MARC	1	zh, de, en, fr, fr, en, de, en, fr, jp, zh, zh, jp, de, jp, de, fr, zh, jp, en
	2	jp, zh, de, en, fr, zh, fr, jp, de, fr, de, jp, en, de, fr, zh, en, zh, jp, en
	3	fr, zh, en, jp, en, jp, zh, fr, de, jp, zh, en, en, de, zh, fr, jp, de, de, fr
MTOP	1	fr, hi, es, th, es, en, de, en, de, fr, hi, es, hi, en, th, fr, hi, de, es, th, fr, de, th, en
	2	th, hi, fr, de, es, en, es, hi, th, de, fr, es, en, de, th, en, fr, hi, hi, es, fr, th, de, en
	3	en, fr, th, hi, de, es, fr, hi, en, th, es, de, fr, hi, th, en, de, es, es, de, fr, en, hi, th

Table 1: Training sequences by dataset. We generate three random sequences each for both datasets. Each comma-separated entry represents the language of the training set for that step in the sequence.

Test data and evaluation metric We use the original test splits for both MARC and MTOP. Each language has a separate test set. At each step, we evaluate the model on all test sets. This means at each step we evaluate the model on all languages and use the same test sets at each evaluation step. We use average accuracy over all test sets as our evaluation metric.

5 Results

We provide a comparison of the performance of the methods over the MARC and MTOP datasets in Figure 1. For each dataset, we generated three sequences. For these plots, we took the average accuracy at each step across all three sequences, and we show the average accuracy results for all four approaches.

The first important observation is that the different training sequences for each dataset show similar performance trends. This suggests that the approaches are largely stable and show only a small performance variation due to the variation in the order of the languages in the training data, with the exception of *Ada-SeqFT* on the MTOP dataset. This is important because the order of the training data in each sequence is different. For MARC, each sequence has a different order of language-category combinations, and for MTOP, the order of languages is different. The sequence details are provided in Table 1. This means that the performance is stable under variations of the training data. Another observation is that approaches are

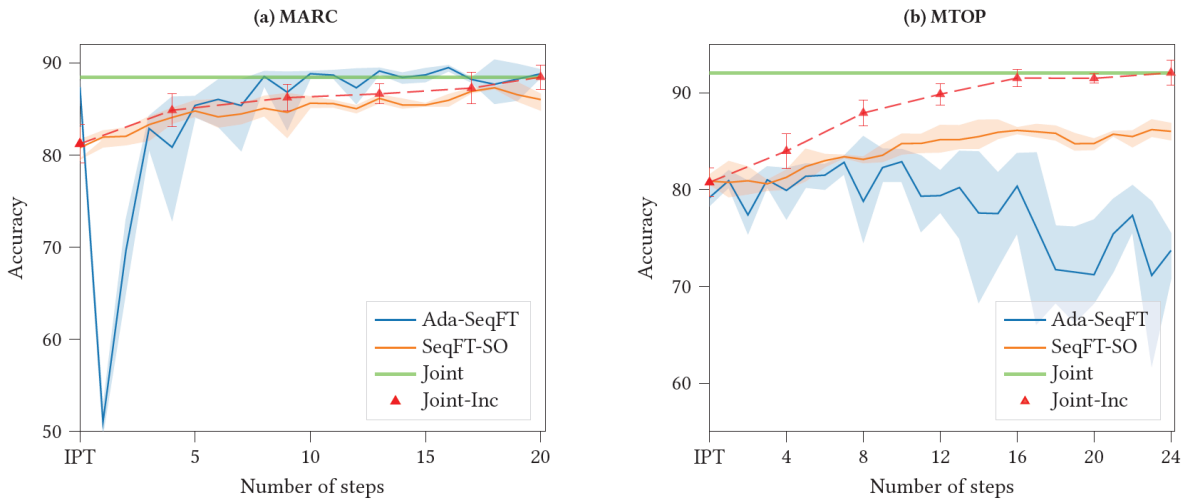

Figure 1: Comparison of average accuracies at each step on MARC (left) and MTOP (right) for the four considered methods. Average accuracy is computed for the same test data at each step. Step 0 is the model after intermediate pre-training (IPT).

stable when languages from different language families are mixed in the training data. This holds for both – a two-class MARC dataset and 113 class MTOP dataset. Again, with the exception of *Ada-SeqFT* on the MTOP dataset, we address it below. Let's take the example of *SeqFT-SO* on the MARC dataset. We see in Table 1 that the first three steps for MARC for sequence 1 are Chinese (zh), German (de), and English (en). For sequence 2, we had Japanese (jp), Chinese (zh), and German (de). For sequence 3, we had French (fr), Chinese (zh), and English (en). On the plot in Figure 1, we show the average over the three sequences, and the confidence interval is small, which means the average accuracies for each sequence at step 3 are comparable.

This is an important result. Since in the practical setting, users have no control over the sequence of languages that will be used for fine-tuning, it is important that the model performance does not depend on any particular sequence of languages.

We also investigate the role of the size of the training data. *Joint* is the baseline for when all data is available, and as expected, it performs higher in both datasets. *Joint-Inc* performance improves with every step as the training set size grows for both datasets. This is in line with expectations and existing results from the literature. On the MARC dataset, the performance of all methods seems to converge after 20 steps. *SeqFT-SO* performance is just below *Joint-Inc* and *Ada-SeqFT* after the initial 15 steps performs the same as the *Joint* baseline and slightly outperforms the other two sequential approaches. On the MTOP dataset, we see a wide difference between the approaches. In this case, the *Joint-Inc* outperforms the other two sequential approaches and reaches the same accuracy as the *Joint* baseline at step 22, before all training data that is used for *Joint* is available to it. The performance of *SeqFT-SO*, on the other hand, improves much more gradually.

Another set of interesting results is about the performance of *Ada-SeqFT*. Adapters are supposed to provide better handling for each language, and this can lead to significant improvements. As we mentioned, *Ada-SeqFT* has the best performance on the MARC data set and even performs the same as the *Joint* baseline. However, *Ada-SeqFT* performs worst on the MTOP dataset. We see that after a brief upward trend, the average accuracy starts dropping. This phenomenon was observed in all three sequences, even though the drop begins at different points: at step 8 for sequence 1, at step 18 for sequence 2, and at step 13 for sequence 3. This explains the wider confidence interval for *Ada-SeqFT* after step 8. It seems that X-MOD is prone to catastrophic forgetting when there are many classes in the data. Another interesting observation about *Ada-SeqFT* is on the MARC dataset, the average accuracy after the first non-IPT step collapses to as low as 50%. This was observed in all three sequences. Although the performance recovers at the next step, in a practical model deployment, even a one-time drop is undesirable.

In summary,

- For the two class MARC dataset, there is no difference between using all available training data (*Joint* and *Joint-Inc*) versus only the current training data (*Ada-SeqFT* and *SeqFT-SO*). On the other hand, for the 113-class MTOP dataset, combining all available training data results in much higher accuracy. If privacy is not a consideration and training data can be stored and used throughout the multi-step fine-tuning, it is beneficial to use the *Joint-Inc* approach for multi-class datasets.

- All approaches are stable with respect to the order and the types of languages in the training sequence. This is an important positive result. *Ada-SeqFT* underperforms on MTOP, but it does not seem to be related to the languages.

- For the two class MARC data set *Ada-SeqFT* has the best performance. But on the MTOP dataset, it exhibits catastrophic forgetting. It appears that the adapters approach needs more research when dozens of fine-tuning steps are applied.

- The training set size increases for the *Joint-Inc* approach, and the training time increases accordingly. If longer training time is acceptable, it is beneficial to use the *Joint-Inc* approach for multiclass datasets.

We provide the following recommendation. We recommend using *SeqFT-SO* for two class data. It has a shorter training time because it uses only current training data and is compliant with privacy considerations. If there are data privacy considerations or training time considerations with multiclass data, *SeqFT-SO* is the recommended robust approach that improves model performance over time. If there are no privacy constraints, *Joint-Inc* is expected to provide better performance for multiclass data. *Ada-SeqFT* requires more research as it exhibits unstable performance in step 3 on the two-class dataset and catastrophic forgetting on the multiclass dataset.

6 Conclusion

We provided a comprehensive study of approaches to multilingual continual learning. We carry out multi-step training using a two-class and a 113-class dataset. We consider three dimensions of their difference: the amount of previously seen data that is used in fine-tuning steps, the training time, and handling the fine-tuning for data in different languages. The main result is that all approaches are stable with respect to the order and type of languages in multi-step training data sequences. The adapters approach needs more research to be reliably used in multilingual continual learning, especially for multiclass data. Joining all previous training data results in the best performance. However, if there are privacy constraints or training time constraints, sequential incremental learning is a robust alternative.

References

Kartikeya Badola, Shachi Dave, and Partha Talukdar. 2022. Parameter-efficient finetuning for robust continual multilingual learning.

Giuseppe Castellucci, Simone Filice, Danilo Croce, and Roberto Basili. 2021. Learning to solve NLP tasks in an incremental number of languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 837–847, Online. Association for Computational Linguistics.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. 2019. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

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

Robert French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020. The multilingual Amazon reviews corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4563–4568, 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, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, Online. Association for Computational Linguistics.

David Lopez-Paz and Marc' Aurelio Ranzato. 2017. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Michael Mccloskey and Neil J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169.

Meryem M'hamdi, Xiang Ren, and Jonathan May. 2022. Cross-lingual lifelong learning.

Kadir Bulut Ozler, Kate Kenski, Steve Rains, Yotam Shmargad, Kevin Coe, and Steven Bethard. 2020. Fine-tuning for multi-domain and multi-label uncivil language detection. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 28–33, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Karan Praharaj and Irina Matveeva. 2022. On robust incremental learning over many multilingual steps.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners.

Gido M. van de Ven and Andreas S. Tolias. 2019. Three scenarios for continual learning.

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. 2022. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In *Advances in Neural Information Processing Systems*.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mu Yang, Shaojin Ding, Tianlong Chen, Tong Wang, and Zhangyang Wang. 2021. Towards lifelong learning of multilingual text-to-speech synthesis.