TL-CL: Task And Language Incremental Continual Learning

Shrey Satapara
Dept. of Artificial Intelligence
Indian Institute of Technology Hyderabad
Hyderabad, India
ai22mtech02003@iith.ac.in

P. K. Srijith
Dept. of Computer Science and Engineering
Indian Institute of Technology Hyderabad
Hyderabad, India
srijith@cse.iith.ac.in

Abstract

This paper introduces and investigates the problem of Task and Language Incremental Continual Learning (TLCL), wherein a multilingual model is systematically updated to accommodate new tasks in previously learned languages or new languages for established tasks. This significant yet previously unexplored area holds substantial practical relevance as it mirrors the dynamic requirements of real-world applications. We benchmark a representative set of continual learning (CL) algorithms for TLCL. Furthermore, we propose Task and Language-Specific Adapters (TLSA), an adapter-based parameter-efficient fine-tuning strategy. TLSA facilitates cross-lingual and cross-task transfer and outperforms other parameter-efficient fine-tuning techniques. Crucially, TLSA reduces parameter growth stemming from saving adapters to linear complexity from polynomial complexity as it was with parameter isolation-based adapter tuning. We conducted experiments on several NLP tasks arising across several languages. We observed that TLSA outperforms all other parameter-efficient approaches without requiring access to historical data for replay.

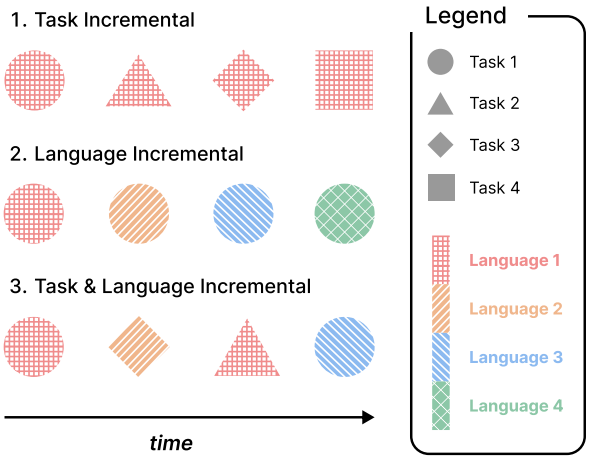

Figure 1: Experimental Setting of different CL scenarios (1) Task Incremental Continual Learning: Learning a sequence of distinct tasks appearing in the same language. While (2) Language Incremental Continual Learning focuses on sequentially learning the same tasks in new languages. (3) Task and language incremental continual learning enable both by learning new tasks for known languages, learning a new language for a known task or learning completely new tasks in a new language sequentially.

1 Introduction

A deployed NLP model may need to be periodically updated for various reasons, such as adapting to data shifts or accommodating newly available training data. Learning new tasks or languages is crucial for real-world NLP models (Wang et al., 2023b). For example, a customer service that handles specific tasks like order tracking and returns in English needs to extend its support for additional languages or handle more tasks like catering to payment issues.

Growing demand for multilingual support makes developing robust multilingual NLP models crucial. A multilingual model (Tang et al., 2020; Xue et al., 2021) is fundamentally a singular deep neural network trained across multiple languages. This unified approach simplifies the training and maintenance of NLP systems by consolidating multilingual capabilities into a single model. It optimizes performance across languages through shared learning architectures. Like monolingual models, multilingual models must be periodically updated to adapt to new tasks or languages. If these updates are not carefully managed, training on new data can reduce the model's ability to perform previously learned tasks or handle earlier languages effectively. This phenomenon is known as catastrophic forgetting (McCloskey and Cohen, 1989; French, 1993). To prevent catastrophic forgetting, a machine learning paradigm called continual learn-

Code Repository: https://github.com/ShreySatapara/TL-CL

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 12123–12142
November 12-16, 2024

ing (CL) (De Lange et al., 2021; Ke and Liu, 2022). CL aims to address the challenges of adapting a pre-trained language model (PLM) to a new task, language or domain while retaining previously obtained knowledge.

Most previous works deal with the challenge of forgetting mitigation for task incremental continual learning (TICL) (Khan et al., 2022; Razdaibiedina et al., 2023; Sun et al., 2019; Wang et al., 2023a). Another line of research deals with CL over language sequences but is mostly limited to language incremental learning for neural machine translation (Gu et al., 2022; Bapna et al., 2019). There is scarce literature on language incremental continual learning (LICL) on other downstream tasks. M'hamdi et al. (2023) studied cross-lingual CL on individual downstream tasks from knowledge preservation, accumulation and generalization perspective. Badola et al. (2023) proposed a parameter-efficient fine-tuning approach using Adapters and language similarity metrics.

The intertwined complexities of task and language incremental continual learning (TLCL) remain unexplored despite the need. We fill that gap in this paper. In the TLCL setting, a pre-trained, multilingual model must be continually updated with a new task for an already-seen language or a new language for an already-seen task. Given a shared multilingual model, the goal of updating is to adapt new task-language[1] pair while retaining knowledge obtained from previously learned task-language data. Figure 1 shows three different types of incremental learning.

We propose Task and Language-Specific Adapters (TLSA), a novel parameter-efficient fine-tuning approach that leverages adapters for continual learning (CL) over tasks and languages. This method enables cross-lingual and cross-task transfer between different task-language pairs. Additionally, TLSA reduces the number of adapters that need to be stored, compared to standard parameter isolation-based fine-tuning using adapters.

We conduct experiments using four different tasks, namely classification (cls), natural language inference (nli), context-based question answering (qa) and summarization (summ), each in four languages: English (en), Spanish (es), Hindi (hi) and Arabic (ar). We comprehensively analyze a wide variety of successful CL algorithms from previous literature investigated in a CL context including (a) regularization (Kirkpatrick et al., 2017), (b) Experience Replay (Chaudhry et al., 2019) and (c) Parameter Isolation (Khan et al., 2022) techniques. The experiments demonstrate the superior performance of TLSA over other parameter-efficient fine tuning approaches.

Our main contributions are:

1. We propose a new problem setup, Task and language incremental continual learning (TLCL), where a multilingual model has to sequentially learn tasks with different objectives from different languages while retaining previous knowledge.

2. We propose TLSA, a novel adapter-based parameter-efficient fine-tuning strategy which enables cross-lingual and cross-task transfer. It outperforms all other parameter-efficient finetuning techniques while reducing the number of adapters that need to be stored.

3. In our experiments, we explore different CL scenarios, ranging from complete task-language data availability to highly constrained single-language settings, to simulate the complexities of real-world deployments.

2 Related Works

From an experimental setup perspective, existing CL in NLP (Biesialska et al., 2020) can be broadly divided into TICL and LICL. Researchers have broadly used replay-based techniques (de Masson D'Autume et al., 2019; Han et al., 2020; Lopez-Paz and Ranzato, 2017) that store a subset of samples for future rehearsal via experience replay. Additionally, Regularisation-based strategies (Kirkpatrick et al., 2017; Farajtabar et al., 2020) which restrict changes in model parameters to avoid interference with already learned tasks. Architecture-based (Razdaibiedina et al., 2023; Wang et al., 2023d) CL methods learn separate sets of parameters for each task.

Task incremental continual learning TICL (De Lange et al., 2021) aims to develop algorithms to accumulate knowledge on non-stationary data for different tasks. Some previous works for TICL in NLP include LAMOL (Sun et al., 2019), which proposes a pseudo-replay-based approach where it learns downstream tasks. It also learns to generate training data for downstream tasks simultaneously. Parameter isolation-based methods such as Wang

[1]Throughout paper task-language term refers to a task in particular language

et al. (2023c) learn a small set of private parameters for each task with a shared and frozen pretrained model, while Khan et al. (2022) proposes training separate sets of adapters for each downstream task. Progressive Prompts (Razdaibiedina et al., 2023) trains new soft prompts for each task and sequentially concatenates them with the previously learned prompts. Wang et al. (2022) also trains a separate set of adapters for task-oriented dialogue systems.

Language incremental continual learning (LICL) also known as multilingual continual learning (MCL) (M'hamdi et al., 2023), studies multilingual modelling from a CL perspective. Many researchers have studied language incremental CL for neural machine translation (NMT) (Berard, 2021) tasks. Berard (2021); Chuang et al. (2020); Gu et al. (2022) and Escolano et al. (2019) propose a method by replacing shared vocabulary with small language-specific vocabulary and fine-tuning embeddings on the new language parallel data. Chuang et al. (2020) suggests using knowledge distillation for continual NMT. M'hamdi et al. (2023) explores cross-lingual continual learning from knowledge preservation and cross-lingual generalization perspective.Badola et al. (2023) proposes a sparse fine-tuning approach for LICL.

Parameter-efficient finetuning: Parameter-efficient fine-tuning (PEFT) has recently garnered attention from many researchers, especially for large language models. Recent work on PEFT has shown that training a subset of parameters can achieve full model performance (Karimi Mahabadi et al., 2021; Li and Liang, 2021; Houlsby et al., 2019a). PEFT methods like adapters have shown promising results for zero-shot cross-lingual transfer (Pfeiffer et al., 2020b) as well as continual learning (Ke et al., 2021; Bapna et al., 2019).

3 Problem Definition And Background

This section formally defines task and language incremental continual learning (TLCL). To the best of our knowledge, we are the first to study TLCL, where the focus is on continually learning both new tasks and new languages.

3.1 Problem Setup

Our proposed problem setting of task and language incremental continual learning involves a sequence of downstream problems, where each problem is to solve a task in a language. We assume that there exists a sequence of $|K|$ downstream problems each associated with a dataset $\mathcal{D} = \{\mathcal{D}_{tl}\}_{(t,l)\in K}$, where $K \subseteq (T \times L)$, $T \subseteq \mathbb{T}$ (set of all tasks), and $L \subseteq \mathbb{L}$ (set of all languages). The dataset $\mathcal{D}_{t,l}$ is specific to a unique pair of task t and language l, and contains data points $(X^{t,l}, Y^{t,l}) = \{X_i^{t,l}, Y_i^{t,l}\}_{i=1}^{N^{t,l}}$ where $X_i^{t,l}$ is an input in language l and $Y_i^{t,l}$ is the corresponding output for task t, with $N^{t,l}$ being the number of the data points for task t in language l.

As with all CL settings, datasets for all task-language pairs are not available at once; rather, they arrive sequentially one after another. We need to train the model on each task-language data successively without forgetting previously learned task-language combinations under finite memory constraints.

3.2 Continual Learning Methods

We briefly highlight previous approaches in continual learning that are relevant to our methodology.

Elastic weight consolidation (EWC): The core idea behind EWC (Kirkpatrick et al., 2017) is to protect model weights that are important to previous tasks. This is achieved by adding a regularization term to the loss function, which constrains weight updates for crucial parameters while allowing the model to learn new tasks. This is achieved through weighing by the Fisher Information Matrix.

Adapter (Houlsby et al., 2019b) based approaches are lightweight alternatives to full model fine-tuning. Adapters are small feed-forward neural networks in an encoder-decoder configuration that are introduced at every transformer layer. They are generally small in size relative to the back-bone transformer. Adapters are fine-tuned on downstream tasks while keeping the backbone model frozen. We describe adapters and EWC in further detail in Appendix B

4 Methodology

In continual learning, a significant challenge is forgetting, which arises due to updates in models' parameters, which may overwrite information that was crucial for previous task-language pairs. Traditional CL methods like ER (Chaudhry et al., 2019), and EWC (Kirkpatrick et al., 2017) mitigate this to an extent but require fine-tuning the entire model for each new task-language pair, which is computationally expensive. Parameter Isolation (PI) based parameter efficient fine-tuning (PEFT) meth-

ods like Adapters address forgetting and computational costs by training only a subset of parameters.

To address our problem setup, we can consider an adapter-based fine-tuning approach on a multilingual pre-trained model, where we train separate adapters for each task-language pair. However, this can limit transfer learning opportunities and lead to a polynomial increase in the number of adapters that need to be stored when a new task or language is added. To address these issues, we propose Task and Language-Specific Adapters (TLSA), which support cross-lingual and cross-task transfer while reducing number of adapter sets that need to be stored from polynomial to linear if a new task or language is introduced.

4.1 Task And Language Specific Adapters

We propose Task and Language-Specific Adapters (TLSA), which aim to facilitate CL across tasks and languages. The core idea of the approach is the segregation of task-specific and language-specific information through separate adapters. Starting with a pretrained model with parameters θ_b, we add task-specific and language-specific adapters with parameters θ_t and θ_l respectively for each downstream task. By training task-specific adapter parameters (θ_t) for each task across multiple languages with EWC regularization, these adapters can accumulate task-specific knowledge while remaining language-agnostic. conversely, by training language-specific adapter parameters (θ_l) across all tasks for a given language with EWC regularization, these adapters can accumulate language-specific knowledge while remaining task-agnostic. The key elements of TLSA are as follows:

Task-specific adapters: When a new task t is introduced, the corresponding task-specific θ_t is employed to learn new task. If the same task t reappears with a different language, the previously learned task-specific θ_t is fine-tuned using Elastic Weight Consolidation (EWC) regularization. The same task-specific adapters are shared across all languages for a particular task, enabling cross-lingual transfer between tasks.

Language-specific adapters: When a new language l is presented, the corresponding language-specific adapter parameter θ_l is employed. If the same language l reappears with a different task t, the previously learned θ_l is fine-tuned using EWC regularization. Here, the same language-specific adapters are shared across all tasks for each language, enabling cross-task knowledge transfer.

We define the optimization objective for task t and language l in the following way:

$$\underset{\theta_t,\theta_l}{\operatorname{argmin}}(\mathcal{L}(Y^{t,l}, F(X^{t,l}, \theta_b, \theta_t, \theta_l) \quad (1)$$

$$+\alpha[\mathbb{1}_{t\in S_{tasks}} R(\theta_t, \theta_t^*) + \mathbb{1}_{l\in S_{langs}} R(\theta_l, \theta_l^*)])$$

where, S_{tasks} is the set of previously seen tasks. S_{langs} denotes set of previously seen languages. $R(\cdot,\cdot)$ denotes EWC regularization (Kirkpatrick et al., 2017) (details in the Equation B.1) and α is the regularization strength. Here, θ_t^* is the adapter parameters trained for task t until the previous task-language pair. Similarly, θ_l^* denotes language-specific adapter parameters for l trained until the previous task-language pair. The fisher information for each task and language-specific adapters is calculated separately after finishing training on each task-language data in sequence. Appendix A Algorithm 1 shows the end-to-end training process for TLSA. During Inference for a task in a particular language, respective task-specific and language-specific adapters are used. Figure 2 illustrates positioning of the task and language-specific adapters in TLSA. We observe that the task-specific and language-specific adapters can vary in number and are applied to different transformer blocks; more details on this are available in subsection 5.6.

For the parameter isolation-based setup with $|T|$ tasks and $|L|$ languages, the total number of adapters required is expressed as $|T| \times |L| \times (n_a N)$ where N denotes the total number of encoder and decoder layers, and n_a denotes no. of adapters to be added into one encoder layer plus one decoder layer. In this setup, parameter growth can be described as $\mathcal{O}(|T| \times |L|)$, indicating polynomial growth as new tasks or languages are added. For the TLSA setup, the required number of adapters is given by: $|T| \times (n_a(N - N_l)) + |L| \times (n_a N_l)$, where N_l is the number of layers containing language-specific adapters. The parameter growth in this model is $\mathcal{O}(|T| + |L|)$, signifying linear growth with the addition of new tasks or languages.

While TLSA enables knowledge transfer across tasks and languages, EWC regularization term need not be very effective in dealing with catastrophic forgetting. In contrast, TLSA with parameter isolation (TLSA_{PI}) overcome forgetting by saving adapters after training on each task-language pair and reusing them during inference for those pairs. Though TLSA_{PI} can be very effective, it introduces a tradeoff between memory usage and performance compared to TLSA.

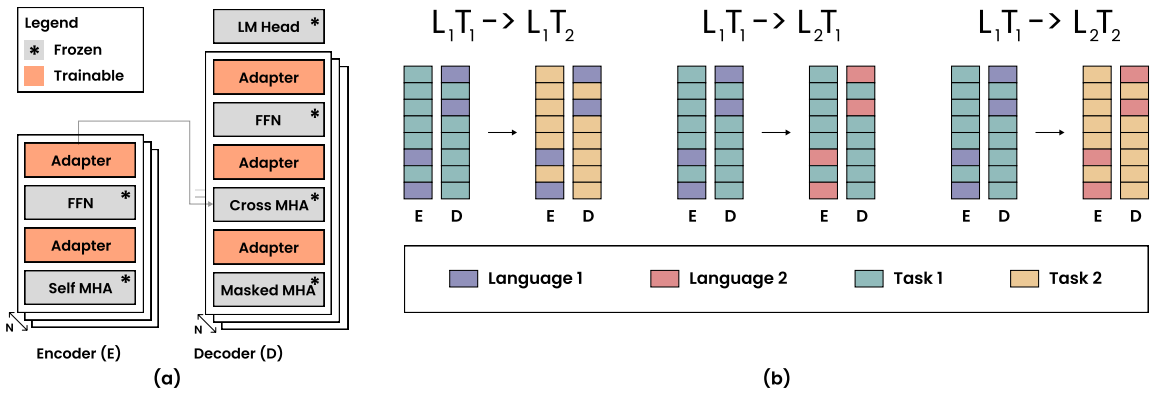

Figure 2: (a) The positions where adapters are added in each transformer encoder and decoder layer, here N indicates no. of encoder and decoder layers stacked on top of each other. (b) Three different scenarios for new task-language pair in the sequence. Here, different colours represent encoder (E) or decoder (D) containing task-specific and language-specific adapters for the respective task and language mentioned in the legend. Here, L_1, L_2, T_1, and T_2 mean language 1, language 2, task 1, and task 2, respectively. Here, each box represents a transformer layer, where either task-specific or language-specific adapters are added to all positions illustrated in Figure (a).

5 Experimental Setup

5.1 Tasks And Datasets

To evaluate our approach, we consider four tasks, namely text classification (cls), natural language inference (nli), context-based question answering (qa) and text summarization ($summ$). Each task in the four languages of English (en), Spanish (es), Hindi (hi) and Arabic (ar). For Natural language inference and question answering, we utilize the XNLI (Conneau et al., 2018) and the XQuAD (Artetxe et al., 2020) datasets, respectively. For languages other than English, we adopt the translations by XTREME (Hu et al., 2020). As for Text Summarization, data is acquired from the XLSum (Hasan et al., 2021) dataset. Finally, to evaluate text classification, we choose Augustyniak et al. (2023)'s sentiment analysis dataset. As the dataset contains Hindi in its romanised form, we combine datasets from OdiaGenAI[2] and Kaggle[3] for the text classification task in Hindi.

5.2 Dataset Construction

We combine a few different datasets as mentioned in subsection 5.1. Some datasets have many training examples; for example, XNLI has 4.5 million training samples. To reduce the expensive computational cost of our experiments, we randomly sample 8000 samples for training; for validation and testing, we try to keep 1000 and 2500 data points for each task-language pair data. However, for some task-language pairs, validation and testing of the data is less due as sufficient examples are unavailable. We use XQuAD for test answering, and training and validation were created using translations provided by (Hu et al., 2020). Dataset statistics for all task-language-specific data is available in Table C.4

To make better use of pre-trained knowledge by reformulating tasks(Liu et al., 2023), we unify all tasks into question-answering tasks. This allows the model to leverage signals across tasks and languages effectively, as suggested by Zhao and Schütze (2021). We use English prompts for all task-language pairs for better cross-lingual cross-task transfer as suggested by (Fu et al., 2022). The exact prompts used for each task are available in Table C.5.

We test these methods for three distinct scenarios for TLCL: (1) Complete Task-Language Sequences: In this setup, we assume that task data for each task is available in all languages. For example, if there are two tasks qa and nli both in two languages en and hi, we have data available for all four task-language pairs (qa-en, nli-en, qa-hi, nli-hi). The second scenario is (2) Partial Task-Language Availability, where task data is available in only a subset of languages, i.e. data available only for (qa-en, nli-en, nli-hi). Another scenario is a single language constraint where tasks are available in different languages, and neither task nor language is repeated, i.e. data is available only for

[2]Huggingface OdiaGenAI Hindi Dataset
[3]Kaggle Hindi Sentiment Analysis Dataset

(qa-en, nli-hi). For each of these scenarios, we consider three different task-language sequences. The exact sequences are available in appendix in Table C.6.

5.3 Evaluation

Following McCann et al. (2018), we employ exact match (EM) as an evaluation metric for cls and nli tasks across all languages. For qa, we use the F1 score, and Rouge (Lin, 2004) score is used for $summ$. More details on these metrics are available in Appendix C.

PercentLoss: We use the percentage loss metric instead of absolute performance to compare different fine-tuning strategies. For Task-Language pair t, l percentage loss is calculated as $PercentLoss_{t,l} = 100 \times \left(\frac{UB_{t,l} - R^{k'}_{t,l}}{UB_{t,l}}\right)$ where $UB_{t,l}$ is the performance of UpperBound methods for task t in language l[4], and $R^{k'}_{t,l}$ is the performance of respective methods on (t, l) task-language pair when trained till k' (final task-language pair). This percentage metric normalizes for the varying difficulties across tasks and evaluation metrics. A higher per cent loss indicates larger deviations from the multitask upper bound. This enables a fair comparison by penalizing methods more for degradation on easier tasks with high upper-bound performance.

Other than average PercentLoss, we use a task-wise performance average, forgetting, and forward transfer (Rodríguez et al., 2018) for all three different task-language sequences in all scenarios for evaluation. More details on evaluation metrics are available in Appendix D. We report all results averaged across three different task-language sequences. The order in which task-language data arrive for each sequence is available in appendix in Table C.6.

5.4 Baselines

Lower bound baseline sequential fine-tuning: This is native sequential fine-tuning (SFT). Where all model parameters are trained on the sequence of task-language data without any regularization or replay samples.

Non-continual upper bound baseline: These are strong upper bound models used as reference points for performance. Multilingual multitask finetuning (MTMLFT) trains a single model jointly across all task-language data. Individual finetuning (IFT) trains independent models on the dataset for each task-language pair.

EWC (Kirkpatrick et al., 2017): We focus on an elastic weight consolidation-based method, which mitigates forgetting by reducing the changes in parameters important for previously seen task-language pairs.

Experience replay (ER) (Chaudhry et al., 2019): This method alleviates forgetting by maintaining a size memory buffer balanced between each task-language pair and regular examples from memory for replay

Adapter finetuning: As mentioned in subsection B.2, adapters are small units added and fine-tuned on downstream tasks while keeping the base model frozen. These modules can be added after the attention block, or FFN in each transformer layer.

In this work, we explored two variants of adapter-based fine-tuning. **Adapter finetuning with parameter isolation** (AT) where we initialize and train a new set of parameters and save them separately for each task-language data in sequence. For a fair comparison of TLSA and replay-based method in terms of no. of trainable parameters, we have a second variant, **Adapter Finetuning with Experience replay** (AT_{ER}). In (AT_{ER}), the same adapters are finetuned while keeping the base model frozen across all task-language data sequentially with an experience replay.

Task specific adapters: Like TLSA, we explored the task-specific adapter (TSA) method, where instead of having separate adapters for each task and language, we only have separate adapters for each task added to all layers of transformer encoder and decoder. Just like task-specific adapters in TLSA, in this case, also task-specific adapter parameters θ_t is employed to learn a new task. If the same task t reappears, the previously learned θ_t is fine-tuned with regularization. The idea is to facilitate cross-lingual transfer for the same task across all languages.

Continual learning using MAD-X: We use MAD-X (Pfeiffer et al., 2020b) in CL setup (MAD-X_{CL}) by inserting invertible adapters and stacking task and language adapters in each layer of the mT5 model. Like TLSA, language-specific adapters are trained across tasks, and task-specific adapters across languages. Unlike TLSA, MAD-X uses frozen pre-trained language adapters, but for a fair comparison, we modify it to use randomly

[4]We consider multitask multilingual finetuning as a reference as it's considered as upper bound (UB) for CL framework

initialized adapters for both tasks and languages.

5.5 Implementation Details

For our experiments, we formulate text-to-text formulation for all experiments, where for cls and nli tasks, labels are mapped to words (e.g. 0/1/2 could be mapped to negative, positive, or neutral for cls task). We use the mT5 model (Xue et al., 2021), a multilingual encoder-decoder-based model pre-trained for masked language modelling objectives. In our experiments, we used pre-trained mT5-small checkpoint[5]. We train the model using cross-entropy loss, and additional regularization is added for approaches like TLSA, TSA and EWC.

Adapter placements: Based on an ablation study mentioned in Appendix E for placements of task and language-specific adapters, we placed two adapters in each encoder layer for all experiments, one after multi-head attention and one after FFN. We placed three adapters in each decoder block, one after self-attention, one after cross-attention, and one after FFN for each decoder block.

5.6 Results

This section compares different CL methods based on evaluation metrics mentioned in section 5.3. The results presented are averaged across three different task-language sequences. We found that TLSA$_{PI}$ outperforms all other methods, followed by ER and TLSA. Notably, ER performs better than all methods except TLSA$_{PI}$, even when using just 1% of the data for each task-language pair for replay. This superior performance is largely due to the replay samples acting as inherent regularization, combined with a higher number of trainable parameters (due to full fine-tuning of the backbone model), compared to parameter-efficient fine-tuning (PEFT) methods. Scalability analysis of Experience Replay with different no. of samples for replay is available in Appendix H

For a fair comparison in terms of trainable parameters, we also experimented with AT$_{ER}$. Despite showing strong forward transfer across most tasks, AT$_{ER}$ suffers from high forgetting, leading to poorer performance compared to TLSA. TLSA, without parameter isolation, outperforms all other PEFT methods and achieves nearly equivalent results to experience replay (less than 1% AvgPercentLoss difference) without accessing any historical data. Given that TLSA$_{PI}$ has polynomial memory growth but offers superior performance compared to TLSA, the choice between these variants ultimately depends on the tradeoff between memory consumption and performance needs.

Table 1 contains the results of complete and partial task-language sequences, and Table 2 contains the results of task-language sequence with a single language constraint. Evaluation based on forgetting and forward transfer is available in Appendix F. In a single language constraint-based sequence, where no tasks or languages are repeated, AT, TSA, and TLSA yield the same performance. Therefore, we report only the results for AT. Since AT is a parameter isolation-based task-language order invariant technique, it will perform the same across all task-language sequences.

No. of layers for task and language-specific adapters?: As the mT5-small is already pre-trained on the multilingual corpus, it already contains language-related knowledge; hence, we decided to keep fewer layers containing language-specific adapters than layers containing task-specific adapters. We did the ablation study on different layers containing language-specific and task-specific adapters on task-language Seq1. We observed that out of 8 encoders and 8 decoders of mT5-small, 2 encoder and 2 decoder layers with language-specific adapters and the rest with task-specific adapters in TLSA method perform optimally, compared to more or less no of layers with language-specific adapters. Performance for different language-specific adapters is available in Figure 3(b). Task-wise details are available in Table H.16

Regularization strength for EWC in TLSA: To investigate the influence of regularization strength on the performance of TLSA, we conducted experiments on Complete Task-Language Sequence on order 1 (refer Table C.6). Figure 3 (a) shows that TLSA with EWC regularization performs better than TLSA without EWC regularization. However, it shows that setting regularization strength to high values hurts the model's performance. Taskwise Avg. PercentLoss is available in Table H.17.

Cross lingual and cross task transfer in TLSA: As outlined in subsection 4.1, since task-specific adapters are shared across all the languages for a given task, they were hypothesized to facilitate the cross-lingual transfer. Conversely, language-specific adapters are shared across all tasks for a particular language and enable cross-task transfer.

[5]Huggingface google/mt5-small checkpoint

	Performance on Complete Task Language Sequences				Performance on Partial Task-Language Sequences			
Method	**Seq1**	**Seq2**	**Seq3**	**Average**	**Seq4**	**Seq5**	**Seq6**	**Average**
Continual learning methods with 100% trainable parameters								
SFT	35.10	63.26	34.64	44.33 ± 16.39	55.57	56.08	50.41	54.02 ± 3.14
EWC	38.59	53.03	40.49	44.04 ± 7.85	46.66	48.05	46.25	46.99 ± 0.94
ER	9.42	12.77	12.26	11.48 ± 1.81	**6.41**	6.00	7.06	6.49 ± 0.53
Parameter efficient CL methods with less then 0.5% trainable parameters								
AT	13.79	13.79	13.79	13.79 ± 0.00	11.42	11.42	11.42	11.42 ± 0.00
AT_{ER}	11.50	14.15	13.47	13.04 ± 1.38	15.72	14.46	13.79	14.66 ± 0.98
MAD-X_{CL}	22.09	25.33	19.23	22.22 ± 3.05	34.61	26.98	13.24	24.94 ± 0.00
TLSA	10.32	12.82	11.74	11.63 ± 1.25	7.85	5.09	8.81	7.25 ± 1.93
$TLSA_{PI}$	**8.82**	**8.05**	**6.19**	**7.69 ± 1.35**	6.92	**5.67**	**5.93**	**6.17 ± 0.66**
TSA	20.20	23.70	24.26	22.72 ± 2.20	14.62	14.93	12.77	14.11 ± 1.17
IFT	0.35	0.35	0.35	0.35 ± 0.00	-4.85	-4.85	-4.85	-4.85 ± 0.00
MTMLFT	0.00	0.00	0.00	0.00 ± 0.00	0.00	0.00	0.00	0.00 ± 0.00

Table 1: Average PercentLoss for task-language sequences and average across three task-language sequences for complete and partial task-language sequences. Note: PercentLoss of MTMLFT is 0.00, as we refer to it as the upper bound to calculate PercentLoss. The highest-performing method is highlighted in bold.

Method	**Seq7**	**Seq8**	**Seq9**	**Average**
Continual learning methods with 100% trainable parameters				
SFT	40.42	85.45	52.90	59.59 ± 23.25
EWC	38.30	82.83	51.79	57.64 ± 22.83
ER	**1.08**	13.27	9.61	**7.99 ± 6.26**
Parameter Efficient CL methods with less then 0.5% trainable parameters				
AT	9.60	**9.60**	**9.60**	9.60 ± 0.00
AT_{ER}	17.03	46.16	21.85	28.35 ± 15.61
MAD-X_{CL}	12.24	12.24	12.24	12.24 ± 0.00
IFT	-7.98	-7.98	-7.98	-7.98 ± 0.00
MTMLFT	0.00	0.00	0.00	0.00 ± 0.00

Table 2: Average PercentLoss for task-language sequences and average across three task-language sequences single language constraint based task-language sequences. Note: PercentLoss of MTMLFT is 0.00, as we refer to it as the upper bound to calculate PercentLoss.

For the empirical evaluation of these hypotheses, we conducted experiments where, after initially training task-specific adapters in one language, we froze these adapters and sequentially trained the model on the same task in other languages, updating only language-specific adapters. Similarly, after initial training on language-specific adapters in one task, we froze those adapters and trained the model on different tasks within the same language.

The observations revealed that updating task-specific and language-specific adapters significantly enhanced cross-lingual and cross-task transfer effectiveness. Figure 3(d) illustrates the performance comparison between the TLSA model with frozen language-specific adapters, as discussed earlier, and the standard TLSA approach across all tasks within the same language. Additionally, Figure 3(c) provides a comparative analysis of cross-lingual transfer across all languages for each task, using the previously described model with frozen task-specific adapters and the original TLSA configuration.

How does continual learning methods generalize to unseen task-language pairs?: To analyze the zero-shot transfer to unseen task-language pairs during training, we made the inference on remaining task-language pairs in the case of partial task-language sequences. We infer that most CL methods have a positive correlation between their performance for each task in terms of average performance for each task in seen and unseen languages. Table 3 contains a task-wise performance of different CL methods for zero-shot generalization averaged across three sequences. We observed that while most methods perform well on classification (cls) and natural language inference (nli) tasks, their performance significantly declines on more difficult tasks such as question-answering (qa) and summarization ($summ$). This is primarily due to models training on specific tasks that in-

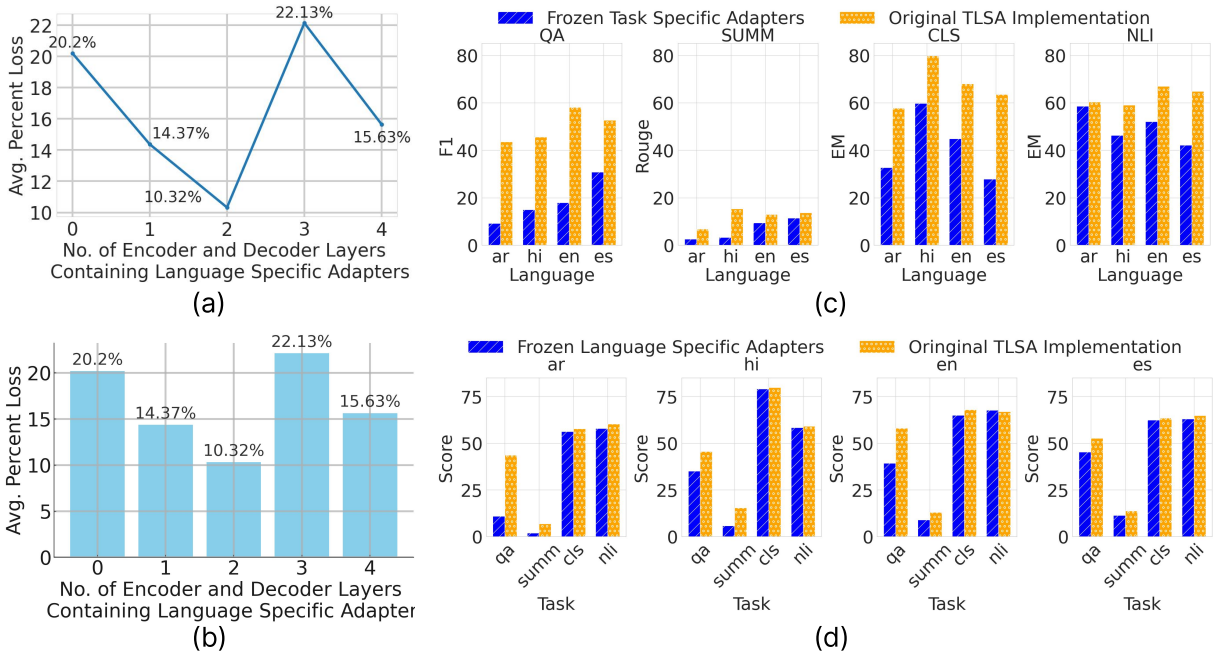

Figure 3: Figure (a) shows the Average percentage loss for different regularization strengths for task-language Seq1. (b) shows the Average percentage loss for task-language Seq1 for different no. of layers containing language-specific adapters. (c) Language-wise comparison for each task for TLSA with frozen task-specific adapters and original TLSA implementation. (d) Task-wise competition for each language for TLSA with frozen language-specific adapters and original TLSA implementation.

Method	CLS	NLI	QA	SUMM
SFT	45.47	37.05	0.60	0.01
EWC	43.35	39.59	1.99	0.03
ER	46.80	51.50	8.87	0.14
TSA	**54.85**	**58.48**	**40.59**	**2.84**
TLSA	46.35	57.45	6.07	0.48
AT$_{ER}$	42.67	48.81	11.27	0.05
MAD-X$_{CL}$	34.62	39.24	2.07	0.03
MTMLFT	52.37	53.87	13.14	0.49

Table 3: Taskwise average performance on three task sequences for zero-shot generalization using remaining task-language pairs in partial task-language sequence.

troduce strong token biases, negatively impacting their zero-shot performance.

For example, when training models on classification and NLI tasks in English and Spanish, the language-specific adapters develop biases towards tokens associated with these tasks and labels. As a result, this specialization leads to poor performance when the model is applied in zero-shot settings to question-answering and summarization in the same languages. Among the methods evaluated, TSA consistently outperforms others, including TLSA, in zero-shot scenarios. TSA facilitates better cross-lingual transfer, while all other methods except AT are capable of cross-lingual transfer, suffer in zero-shot settings due to inter-task interference, resulting in poorer performance than TSA.

6 Conclusion And Future Work

We introduce and study the problem of task and language incremental continual learning. We provide the first benchmark to compare the effectiveness of different CL methods for the TLCL case. We further introduce TLSA, a parameter and memory-efficient fine-tuning method for TLCL, and its variant TLSA$_{PI}$, eliminating forgetting by parameter isolation. TLSA$_{PI}$ outperforms all other CL methods for complete and partial task-language sequence, while ER performs better in single-language constraint-based scenarios. TLSA outperforms all other PEFT methods and performs on par with ER without having access to historical data. The selection of TLSA vs TLSA$_{PI}$ can be made based on a tradeoff between memory and performance. We also explore their capabilities for zero-shot generalization to unseen task-language pairs. However, we find them to struggle with zero-shot generalization. Therefore, a key future direction is to improve zero-shot transfer capabilities within CL to unseen task-language pairs.

Limitations

Since this is one of the first studies on task and language incremental continual learning, the paper focuses on laying the groundwork by establishing the experimental setting on a set of NLP tasks and languages. Possible limitations of our experimental setup are the restrictions on evaluating the mT5-small checkpoint and fixing the number of samples for each task-language data. However, in real-life scenarios, one may have access to different amounts of data for each task-language pair. In our setup for evaluation, the language chosen (en, es, hi, ar) is diverse but still a relatively high resource; extending the TICL setup on underrepresented (or even absent) during the pretraining of the model may provide interesting insights.

Acknowledgement

We acknowledge the funding support from the Science and Engineering Research Board, India (SERB - CRG/2021/006436). We also acknowledge the support from Japan International Cooperation Agency (JICA) and Sony Research India (SRI). We express our sincere gratitude to Mr. Vishnuprasadh Kumaravelu and Mr. Sayanta Adhikari, members of the Bayesian Reasoning And Inference (BRAIN) Lab at IIT Hyderabad, for their continuous feedback throughout the development of this work.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Lukasz Augustyniak, Szymon Woźniak, Marcin Gruza, Piotr Gramacki, Krzysztof Rajda, Mikoł aj Morzy, and Tomasz Kajdanowicz. 2023. Massively multilingual corpus of sentiment datasets and multi-faceted sentiment classification benchmark. In *Advances in Neural Information Processing Systems*, volume 36, pages 38586–38610. Curran Associates, Inc.

Kartikeya Badola, Shachi Dave, and Partha Talukdar. 2023. Parameter-efficient finetuning for robust continual multilingual learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9763–9780, Toronto, Canada. Association for Computational Linguistics.

Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. *CoRR*, abs/1909.08478.

Alexandre Berard. 2021. Continual learning in multilingual NMT via language-specific embeddings. In *Proceedings of the Sixth Conference on Machine Translation*, pages 542–565, Online. Association for Computational Linguistics.

Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6523–6541, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. 2019. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

Yung-Sung Chuang, Shang-Yu Su, and Yun-Nung Chen. 2020. Lifelong language knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2914–2924, Online. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

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.

Carlos Escolano, Marta R. Costa-jussà, and José A. R. Fonollosa. 2019. From bilingual to multilingual neural machine translation by incremental training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 236–242, Florence, Italy. Association for Computational Linguistics.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. 2020. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR.

Robert French. 1993. Catastrophic interference in connectionist networks: Can it be predicted, can it be

prevented? *Advances in Neural Information Processing Systems*, 6.

Jinlan Fu, See-Kiong Ng, and Pengfei Liu. 2022. Polyglot prompt: Multilingual multitask prompt training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9919–9935, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shuhao Gu, Bojie Hu, and Yang Feng. 2022. Continual learning of neural machine translation within low forgetting risk regions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1707–1718, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6440.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019a. 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.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019b. 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.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, volume 34, pages 1022–1035. Curran Associates, Inc.

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

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.

Shadab Khan, Surbhi Agarwal, and PK Srijith. 2022. Lifelong language learning with adapter based transformers. In *Continual Lifelong Learning Workshop at ACML 2022*.

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.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of

prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730.

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

Meryem M'hamdi, Xiang Ren, and Jonathan May. 2023. Cross-lingual continual learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3908–3943, Toronto, Canada. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. 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.

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

Natalia Díaz Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. 2018. Don't forget, there is more than forgetting: new metrics for continual learning. *CoRR*, abs/1810.13166.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. LAMAL: language modeling is all you need for lifelong language learning. *CoRR*, abs/1909.03329.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning.

Weizhi Wang, Zhirui Zhang, Junliang Guo, Yinpei Dai, Boxing Chen, and Weihua Luo. 2022. Task-oriented dialogue system as natural language generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2698–2703, New York, NY, USA. Association for Computing Machinery.

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

Xiao Wang, Yuan Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023b. Trace: A comprehensive benchmark for continual learning in large language models. *ArXiv*, abs/2310.06762.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and Wenqiu Zeng. 2023c. 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.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, et al. 2023d. 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Mengjie Zhao and Hinrich Schütze. 2021. Discrete and soft prompting for multilingual models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8547–8555, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Task and Language Specific Adapters Algorithm

Algorithm 1 contents end-to-end training process for TLSA where Θ_{tasks} and Θ_{langs} denotes a Dictionary to store parameters of task-specific and language-specific adapters.

Algorithm 1 Incremental Continual Learning with Task and Language-Specific Adapters and EWC

$S_{\text{tasks}} \leftarrow \phi, S_{\text{langs}} \leftarrow \phi$
Initilize empty dictionaries Θ_{tasks} and Θ_{langs}
for all $(t, l) \in K$ **do**
  if $t \in S_{\text{tasks}}$ **then**
    $\theta_t \leftarrow \Theta_{\text{tasks}}[t]$
  else
    $\theta_t \leftarrow$ RandomInitialization()
  end if
  if $l \in S_{\text{langs}}$ **then**
    $\theta_l \leftarrow \Theta_{\text{langs}}[l]$
  else
    $\theta_l \leftarrow$ RandomInitialization()
  end if
  for all $(x_i, y_i) \in \{X_i^{t,l}, Y_i^{t,l}\}_{i=1}^{N}$ **do**
    Optimize based on Equation 1
  end for
  $\Theta_{\text{tasks}}[t] \leftarrow \theta_t, \Theta_{\text{langs}}[l] \leftarrow \theta_l$
  $S_{\text{tasks}} \leftarrow S_{\text{tasks}} \cup \{t\}, S_{\text{langs}} \leftarrow S_{\text{langs}} \cup \{l\}$
end for

B Background

B.1 Elastic Weight Consolidation

For a model that has learned task A and is now learning task B, The loss function for task B, modified with EWC, is given by

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \frac{\lambda}{2} R(\theta, \theta_A^*)$$
$$R(\theta, \theta_A^*) = \sum_i F_i(\theta_i - \theta_{A,i}^*)^2 \qquad (2)$$

Where $\mathcal{L}_B(\theta)$ is the loss function for Task B, θ represents model parameters, θ_A^* represents model parameters after training on task A, which are treated as optimal values for the previous task. F denotes the diagonal fisher information matrix corresponding to parameters θ, indicating the importance of each weight. The summation over i in the equation signifies that this regularization is applied to each parameter individually. This means that each parameter θ_i from the model's parameters θ is compared against its optimal value $\theta_{A,i}^*$ from Task A, weighted by the corresponding value F_i from the fisher information matrix, F_i denotes the following

$$F_i = \mathbb{E}_{x \sim p(x)} \left[\left(\frac{\partial \log p(y|x;\theta)}{\partial \theta_i} \right)^2 \right] \qquad (3)$$

B.2 Adapters

Adapters (Houlsby et al., 2019b) are lightweight alternatives to full model fine-tuning, consisting of only a tiny set of newly introduced parameters at every transformer layer. These adapter layers comprise a down projection matrix D, which projects input into lower dimensional space h_{down}, a nonlinearity f and an Up Projection U. The associated transformation can be described as $A(h) = h + U^T f(D^T h)$, where h is hidden representation input to the adapter block. These adapters can be placed after the transformer layer's attention block or feed-forward network (FFN).

C Dataset

C.1 Task Description

- Classification (Augustyniak et al., 2023): We consider sentiment analysis, where the goal is to classify the sentiment expressed in the input sentences into one of the following classes: positive, negative or neutral.

- Natural Language Inference (NLI) (Conneau et al., 2018): The NLI task is to find if a given premise sentence contradicts, entails or is neutral towards the hypothesis sentence.

- Context-Based Question Answering (Artetxe et al., 2020): Given a paragraph, the task is to identify the answer to a question as a span of the corresponding paragraph.

- Text Summarization (Hasan et al., 2021): Generate a concise and meaningful text summary given a text document

Task	Dataset	Evaluation Metrics	Language	Abbrivation	Train	Val	Test
Natural Language Inference (nli)	XNLI	EM	English	nli-en	8000	1000	2500
			Hindi	nli-hi	8000	1000	2500
			Spanish	nli-es	8000	1000	2500
			Arabic	nli-ar	8000	1000	2500
Text Summarization (summ)	XSUMM	ROUGE	English	summ-en	8000	1000	2500
			Hindi	summ-hi	8000	1000	2015
			Spanish	summ-es	8000	578	563
			Arabic	summ-ar	8000	1000	1262
Context Based Question Answering (qa)	SQUAD + XSQUAD	F1 Score	English	qa-en	8000	1000	1168
			Hindi	qa-hi	8000	1000	1094
			Spanish	qa-es	8000	1000	1150
			Arabic	qa-ar	8000	1000	1136
Text Classification (cls)	MMS	EM	English	cls-en	8000	1000	2500
			Spanish	cls-es	8000	1000	2500
			Arabic	cls-ar	8000	1000	2500
	OdiaGenAI + Kaggle		Hindi	cls-hi	8000	1000	2500

Table C.4: Dataset source, statistics, and evaluation metrics for all 16 task-language pairs used in our CL experiments.

D Evaluation Metric

We use the following evaluation metrics to measure the performance of different tasks that we considered in this experimental setup suggested by McCann et al. (2018). **Exact Match** (EM): We use exact match to evaluate cls and nli tasks, where we compare the normalized generated label (strip out articles and punctuation) and actual text label of a particular class. **F1 Score** (F1): We calculate normalized F1 score for question answering task. **Rouge**: For text summarization, we employ the rouge score, as suggested in McCann et al. (2018), we take an average of ROUGE-1, ROUGE-2 and ROUGE-L scores.

To measure transfer and forgetting in CL setup, we calculate forward transfer and forgetting in the following way.

Forgetting: Assume $k' = (t', l'), t' \in T, l' \in L$ as the final task-language pair observed by the model, then we define forgetting for a particular task $t \in T$ as

$$\mathbf{F}_t = \frac{\sum_{l \in L} \left[R_{k',(t,l)} - R_{(t,l),(t,l)} \right]}{\mathbf{1}_{t=t'}(|L|-1) + \mathbf{1}_{t \neq t'}|L|} \quad (4)$$

where $R_{k,\hat{k}}$ implies to the performance of our model on $\hat{k}$ task-language pair when trained till k task-language pair.

Forward Transfer: Assume $\bar{k} = (\bar{t}, \bar{l}), \bar{t} \in T, \bar{l} \in L$ as the first task-language pair encountered by the model, then forward transfer corresponding to a particular task $t \in T$,

$$\mathbf{FWT}_t = \frac{\sum_{l \in L} \left[R_{(t,l),(t,l)} - R_{(t,l)} \right]}{\mathbf{1}_{t=\bar{t}}(|L|-1) + \mathbf{1}_{t \neq \bar{t}}|L|} \quad (5)$$

where $R_{(t,l)}$ implies the performance of the model when independently trained on (t, l) task-language pair.

E Experimental Details

All of our experiments were performed on 3 NVIDIA V100-32GB GPUs on a DGX-2 Server. Our Implementation uses PyTorch (Paszke et al., 2019), the transformers (Wolf et al., 2020), the adapter-hub (Pfeiffer et al., 2020a), accelerate (Gugger et al., 2022), and the datasets (Lhoest et al., 2021) library. For the evaluation metric, we referred to implementation from McCann et al. (2018). We use pre-trained mt5-small checkpoint to initialize our models.

We conducted all of our experiments with per device batch size 8, gradient accumulation 2, and constant learning rate 1e-4 for 25 epochs using the huggingface transformers Seq2SeqTrainer class. We use default reduction factor 16 and Bert initialization (Pfeiffer et al., 2020a) for adapters across all experiments. For TLSA and TSA, the regularization strength used is 1e3, which we found via ablation study over different regularization strengths.

Task	Question	Context	Answer
Text Classification	Is the given sentence positive, negative or neutral?	{Input Sentence }	positive/ negative/ neutral
Natural Language Inference	Hypothesis: {hypothesis_sentence} Entailment, neutral, or contradiction?	Premise: {premise_sentence}	entailment/ neutral/ contradiction
Context Based Question Answering	{Question}	{ Context }	{ Expected Answer }
Text Summarization	What is the summary?	{ Document }	{ Expected Summary }

Table C.5: Prompts used for different tasks, to unify them as context-based question answering tasks. Here, there is no explicit prompt for the question answering task as it's already a context-based question answering task.

We use 1% samples per task-language data for experience replay. For standard EWC-based finetuning, we use regularization strength to be 10. In adapters-based approaches like AT, AT$_{ER}$, TLSA and TSA, only 0.44% parameters are trainable. While other approaches fine-tune all model parameters. In TLSA, we added language-specific adapters to the first and third encoder layers and the sixth and eighth decoder layers. The decision of adapter positioning is based on ablations study mentioned in subsection H.1

Table E.7 contains the total and trainable no of parameters in each approach.

F Taskwise Forgetting and Forward transfer

We evaluated different CL methods using metrics like Forgetting and Forward transfer as defined in Appendix D. Table F.8, Table F.10 and Table F.13 contain task-wise average forward transfer for complete task-language sequence, partial task-language sequence and task-language sequence with single language constraints, respectively.

Table F.9, Table F.11 and Table F.12 contain task-wise average Forgetting for complete task-language sequence, partial task-language sequence and task-language sequence with single language constraints, respectively. We observed that TLSA has less forgetting than other methods.

Since AT is a parameter isolation-based approach, forgetting and forward transfer is zero; hence, we have not added those values to the respective tables. In single-language constraint-based sequences, since there is no overlap between tasks or languages methods like AT, TLSA, TSA, MAD-X$_{CL}$ has zero forgetting and forward transfer since all the parameters are isolated for each task and language; hence, we skipped those values in respective tables.

G Abilation Study

H Scalability Analysis of Experience Replay

As experience replay is outperforming TLSA, we conducted scalability analysis with different no. of data points for replay to see at which point ER outperforms TLSA.

H.1 Abilation Study for TLSA

Along with finding the best regularization strength and no. of layers to add language-specific adapters, after deciding the regularization strength and no. of layers to add language-specific adapters, we conducted an ablation study on placing task and language-specific adapters in different encoder and decoder layers. We observed that placing language-specific adapters in the early layers of the encoder and later layers of the decoder performs better than adding language-specific adapters in the initial layers of the encoder and the initial layer of the decoder. We hypothesise that adding language-specific adapters in initial layers allows it to map to universal representation space across all languages, while adding language-specific adapters at later layers of the decoder helps by adding language-specific bias during generation. We also observe

	Seq	Task-Language Sequence
Complete Task-Language Sequence	1	qa-ar → summ-hi → cls-en → nli-es → qa-hi → nli-en → summ-ar → cls-hi → cls-ar → qa-en → nli-ar → summ-es → summ-en → cls-es → nli-hi → qa-es
	2	nli-hi → cls-ar → summ-en → qa-hi → cls-es → qa-ar → summ-ar → nli-es → summ-hi → qa-en → cls-en → nli-ar → qa-es → cls-hi → nli-en → summ-es
	3	nli-en → qa-es → cls-ar → summ-ar → nli-hi → qa-en → cls-en → summ-hi → qa-ar → cls-es → summ-en → nli-ar → nli-es → cls-hi → summ-es → qa-hi
Partial Task-Language Sequence	4	qa-hi → cls-en → summ-hi → qa-ar → nli-es → summ-ar → nli-hi → cls-es
	5	nli-es → qa-hi → cls-en → summ-hi → cls-es → summ-ar → qa-ar → nli-hi
	6	summ-ar → cls-es → qa-hi → qa-ar → nli-hi → summ-hi → nli-es → cls-en
Single Language Constraint	7	summ-hi → qa-ar → nli-es → cls-en
	8	cls-en → nli-es → qa-ar → summ-hi
	9	qa_ar → cls_en → summ_hi → nli_es

Table C.6: 9 different orders of task-language sequences for continual learning experiments. Seq 1-3 corresponds to a continual learning scenario where all tasks' data is available in all the languages. Seq 4-6 are for partial task-language sequences, where each task can appear in one or more languages. Seq 7-9 denotes experimental scenario for experiments with single language constraint-based sequences.

Method	**Total no. of parameters**	**No. of trainable Parameters**
Methods that requires full model finetuning		
SFT	300,176,768	300,176,768
EWC	300,176,768	300,176,768
ER	300,176,768	300,176,768
IFT	300,176,768	300,176,768
MTMLFT	300,176,768	300,176,768
PEFT based CL Methods		
AT	301,509,248	1,332,480
AT_{ER}	301,509,248	1,332,480
TLSA	301,509,248	1,332,480
TSA	301,509,248	1,332,480
MAD-X_{CL}	301,374,592	1,197,824

Table E.7: No of total and trainable parameters for each method.

that adding LSA in alternate layers performs better than adding them in consecutive layers. Table H.15 contains Avg. PercentLoss for Seq1 for different positioning of TSA and LSA.

Method	CLS	NLI	QA	SUMM
SFT	0.02 ± 0.41	-0.67 ± 1.04	2.97 ± 1.38	0.19 ± 0.05
EWC	-0.44 ± 0.21	-0.02 ± 0.45	2.12 ± 0.43	0.29 ± 0.03
ER	-0.76 ± 0.23	-0.40 ± 1.30	2.44 ± 0.82	0.15 ± 0.28
AT_{ER}	**1.15 ± 0.52**	**10.57 ± 3.80**	**11.52 ± 1.00**	**0.69 ± 0.13**
TLSA	0.92 ± 0.17	9.67 ± 1.58	4.80 ± 2.17	-0.36 ± 0.23
TSA	1.02 ± 0.57	9.15 ± 1.26	3.89 ± 1.88	-0.24 ± 0.16
MAD-X_{CL}	0.65 ± 0.23	9.43 ± 2.26	3.92 ± 2.21	-0.23 ± 0.92

Table F.8: Taskwise average forward transfer average across three task-language sequences for complete task-language sequencer scenario.

Method	CLS	NLI	QA	SUMM
SFT	34.10 ± 10.97	12.88 ± 6.17	23.81 ± 27.64	12.16 ± 1.14
EWC	34.23 ± 8.80	10.64 ± 11.15	23.93 ± 27.68	12.47 ± 1.28
ER	7.29 ± 0.29	3.60 ± 3.34	6.07 ± 5.15	3.44 ± 0.52
AT_{ER}	10.16 ± 0.38	3.65 ± 3.21	1.41 ± 4.82	2.33 ± 0.16
TLSA	**6.20 ± 1.62**	-0.92 ± 1.45	**-0.61 ± 4.94**	**1.41 ± 1.43**
TSA	8.21 ± 0.99	**-2.22 ± 1.59**	1.07 ± 3.07	8.08 ± 0.70
MAD-X_{CL}	13.42 ± 2.12	1.46 ± 3.43	2.09 ± 1.23	6.83 ± 3.28

Table F.9: Taskwise Forgetting average across three different task-language sequences for complete task-language sequencer scenario.

Method	CLS	NLI	QA	SUMM
SFT	0.36 ± 0.43	-0.54 ± 1.61	1.03 ± 0.87	0.12 ± 0.09
EWC	-0.14 ± 1.09	-0.35 ± 0.96	1.02 ± 0.73	-0.02 ± 0.07
ER	-1.05 ± 0.26	-0.25 ± 1.45	0.56 ± 0.68	-0.09 ± 0.09
AT_{ER}	**1.11 ± 0.65**	3.83 ± 3.40	**8.77 ± 1.75**	**0.47 ± 0.22**
TLSA	-0.30 ± 0.44	5.97 ± 1.28	5.86 ± 2.67	-1.02 ± 0.88
TSA	-1.25 ± 0.23	**6.45 ± 1.01**	4.47 ± 0.15	-5.13 ± 2.24
MAD-X_{CL}	-2.34 ± 1.94	2.85 ± 2.94	6.36 ± 1.93	-3.85 ± 0.35

Table F.10: Taskwise average forward transfer average across three task-language sequences for partial task-language sequencer scenario.

Method	CLS	NLI	QA	SUMM
SFT	17.82 ± 22.59	18.96 ± 15.05	44.03 ± 6.00	13.22 ± 3.89
EWC	17.82 ± 23.50	16.91 ± 13.91	38.09 ± 12.68	11.54 ± 4.32
ER	3.97 ± 1.65	3.64 ± 1.92	7.58 ± 3.24	2.36 ± 0.30
AT_{ER}	6.09 ± 3.89	3.42 ± 6.06	6.16 ± 3.58	2.01 ± 0.72
TLSA	**1.55 ± 0.51**	-1.42 ± 0.83	**1.26 ± 2.49**	**0.39 ± 0.81**
TSA	2.46 ± 0.14	**-3.31 ± 1.12**	2.38 ± 0.47	4.55 ± 2.08
MAD-X_{CL}	4.64 ± 1.86	-1.86 ± 0.46	1.76 ± 0.46	3.75 ± 2.57

Table F.11: Taskwise average forgetting averaged across three task-language sequences for partial task-language sequencer scenario.

Method	CLS	NLI	QA	SUMM
SFT	35.37 ± 30.64	31.07 ± 30.64	44.45 ± 4.84	13.47 ± 11.67
EWC	36.62 ± 31.77	29.77 ± 30.23	42.95 ± 7.15	13.20 ± 11.45
ER	**4.62 ± 4.18**	**3.72 ± 3.99**	**13.36 ± 5.47**	**1.48 ± 1.65**
AT_{ER}	11.63 ± 10.22	13.07 ± 12.50	18.26 ± 11.15	0.89 ± 0.79

Table F.12: Taskwise Forgetting averaged across three task-language sequences for single language constraint-based task-language sequencer scenario.

Method	CLS	NLI	QA	SUMM
SFT	-0.84 ± 1.16	-1.19 ± 0.54	0.28 ± 0.52	**0.02 ± 0.09**
EWC	-0.64 ± 1.24	-1.27 ± 0.61	**1.08 ± 0.71**	-0.17 ± 0.04
ER	-1.01 ± 0.22	-1.73 ± 0.49	-0.61 ± 0.51	-0.03 ± 0.04
AT_{ER}	**26.82 ± 30.25**	**26.64 ± 20.86**	-18.69 ± 23.15	-25.21 ± 31.49

Table F.13: Taskwise forward transfer average across three different task-language sequences for single language constraint-based task-language sequencer scenario.

#Examples For Replay	Complete Task-Language Sequence	Partial Task-Language Sequence	Single Language Constraint
10	14.94	23.79	25.58
20	13.61	15.83	11.11
40	**8.34**	**6.17**	**5.28**
80	6.85	6.41	3.54
100	6.32	2.88	2.10

Table H.14: AvgPercentLoss for experience replay using different no. of examples for replay using Seq1, Seq4 and Seq7 for Complete Task-Language Sequence, Partial Task-Language Seq and Single language constraint

Encoder Layers with LSA	Encoder Layers with TSA	Decoder Layers with LSA	Decoder Layers with TSA	Avg. PercentLoss
0,2	1,3,4,5,6,7	5,7	0,1,2,3,4,6	**10.32**
0,1	2,3,4,5,6,7	6,7	0,1,2,3,4,5	10.7
0,1	2,3,4,5,6,7	0,1	2,3,4,5,6,7	25.13
0,2	1,3,4,5,6,7	0,2	1,3,4,5,6,7	25.31

Table H.15: Performence for adding task-specific adapters(TSA) and Language language-specific adapters (LSA) in different layers in encoder and decoder layers.

#LSA in Encoder	#TSA in Encoder	#LSA in Decoder	#TSA in Decoder	CLS	NLI	QA	SUMM	AvgPercentLoss
0	8	0	8	67.03	**64.81**	48.25	6.41	-20
1	7	1	7	67.19	63.37	**51.57**	9.67	-14.31
2	6	2	6	68.8	63.13	51.11	**12.36**	**-9.98**
3	5	3	5	**70.15**	61.7	34.61	8.98	-21.91
4	4	4	4	68.89	59.46	44.47	11.67	-15.25

Table H.16: Taskwise average performance and Avg. PercentLoss for employing different no of task-specific and language-specific adapters in TLSA method.

Regularization Strength	CLS	NLI	QA	SUMM	Avg. PercentLoss
0	8.85	-5.64	27.5	52.74	20.86
1.00E+03	5.21	**-8.38**	**17.53**	**26.91**	**10.32**
1.00E+04	**4.98**	-5.27	40.24	37.45	19.35
1.00E+05	8.31	1.74	47.36	28.06	21.37
1.00E+06	12.43	8.93	53.02	29.92	26.08
1.00E+07	19.01	11.86	56.51	28.82	29.05

Table H.17: Taskwise Average PercentLoss for different regularization strengths for TLSA method.

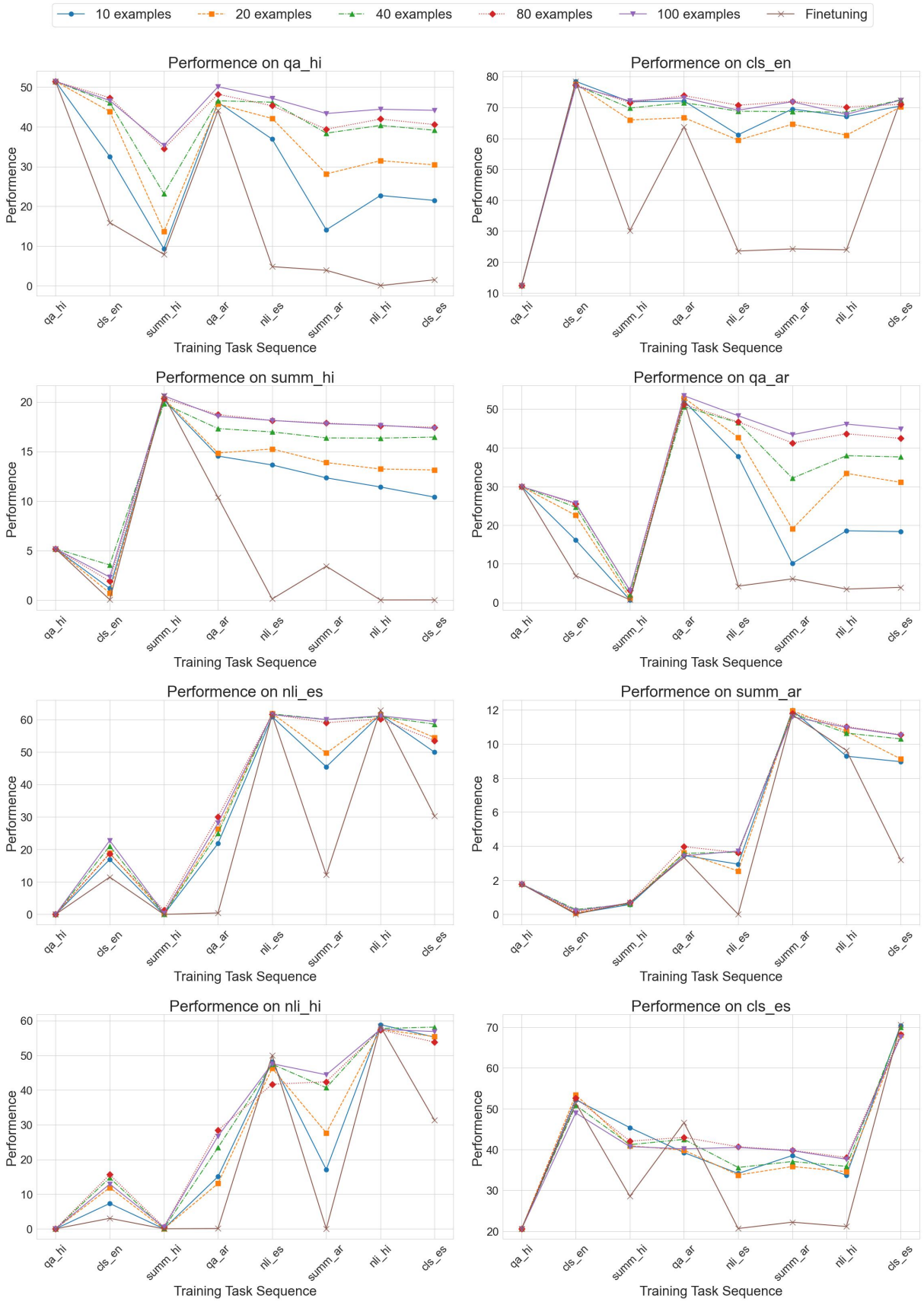

Figure H.4: Performance on different tasks for different numbers of samples when using experience replay. seq4 is shown. The title of each plot denotes the task-language being evaluated. Labels on the X-axis indicate, from left to right, the order in which each task-language training data subset is exposed to the model.