

Analyzing and Reducing the Performance Gap in Cross-Lingual Transfer with Fine-tuning Slow and Fast

Yiduo Guo^{1,*}, Yaobo Liang², Dongyan Zhao^{1,4,5†}, Bing Liu³, Duan Nan²

¹Wangxuan Institute of Computer Technology, Peking University

²Microsoft Research Asia

³Department of Computer Science, University of Illinois at Chicago

⁴National Key Laboratory of General Artificial Intelligence, ⁵BIGAI, Beijing, China

yiduo@stu.pku.edu.cn, yaobo.liang@microsoft.com, zhaody@pku.edu.cn

nanduan@microsoft.com, liub@uic.edu

Abstract

Existing research has shown that a multilingual pre-trained language model fine-tuned with one (source) language also performs well on downstream tasks for non-source languages, even though no fine-tuning is done on these languages. However, there is a clear gap between the performance of the source language and that of the non-source languages. This paper analyzes the fine-tuning process, discovers when the performance gap changes and identifies which network weights affect the overall performance most. Additionally, the paper seeks to answer to what extent the gap can be reduced by reducing forgetting. Based on the analysis results, a method named **Fine-tuning slow and fast** with four training policies is proposed to address these issues. Experimental results show the proposed method outperforms baselines by a clear margin.

1 Introduction

Multilingual pre-trained language models (LMs), such as mBERT (Devlin et al., 2018) and XLM-R (Conneau et al., 2019) have shown strong Zero-Shot Cross-Lingual transfer capabilities. Such a model F is usually pre-trained with unlabeled corpora D in multiple languages S to enable the model to learn cross-lingual knowledge H_{cross}^{pre} . To adapt to a downstream task, the pre-trained LM F is typically fine-tuned with a supervised dataset $D_{\hat{s}}$ of the downstream task T in one source language $\hat{s} \in S$ due to data scarcity in other languages. When the fine-tuned model \bar{F} is applied to the test set of the same task in the source language \hat{s} , it achieves strong performance $P_{\hat{s}}$. Interestingly, when \bar{F} is applied to non-source languages, it can also achieve good performance (Conneau et al., 2019). We denote the average performance on the test sets of other languages than \hat{s} as $P_{S/\hat{s}}$. However, the

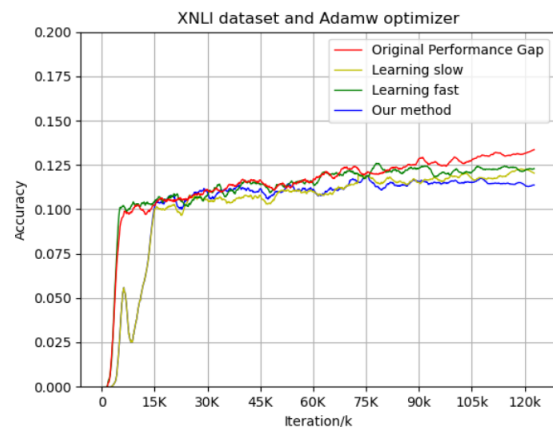


Figure 1: The performance gap $P_{\hat{s}} - P_{S/\hat{s}}$ every hundred updates on the XNLI dataset. 'Original performance gap' means that we directly fine-tune the model, and 'Fine-tuning slow'/'fine-tuning fast'/'our method' means that we use the fine-tuning slow algorithm/fine-tuning fast algorithm/the combination of both algorithms respectively to fine-tune the model.

gap $P_{\hat{s}} - P_{S/\hat{s}}$ is quite large (e.g., 13 percent for the XNLI dataset in Figure 1). One potential reason is that: during the fine-tuning of the model, the performance of non-source languages firstly increases with the performance of source language, then the arising of the performance of non-source languages becomes slower than that of the performance of source language as the forgetting of cross-lingual knowledge, resulting in a larger gap. Inspired by the study of *catastrophic forgetting* (CF) phenomenon in continual learning (CL), we introduce a classical concept in CL here to help solve our problem: the dilemma of *plasticity vs. stability*.

Plasticity vs Stability. In CL (Kirkpatrick et al., 2017), the learner needs to learn a sequence of different tasks incrementally. Plasticity means learning and performing well on the new task and stability means maintaining the learned knowledge of the previous tasks. The learner needs to find a balance between plasticity and stability because too much plasticity (e.g, changing the entire model drastically) causes serious CF of the learned knowledge, and too much stability (e.g. freezing the whole

*Work done during internship at Microsoft Research Asia

†Corresponding author

model) makes the model can not learn new things. Fine-tuning a multi-lingual LM F using only the corpus of one source language also meets this balance dilemma. Thus, Fine-tuning LM F needs to protect the cross-lingual knowledge H_{cross}^{pre} (stability) and also learn the new task knowledge H_{task}^{new} via fine-tuning to adapt to the specific downstream task (plasticity). However, further analysis of the performance gap and the dilemma of plasticity and stability in cross-lingual fine-tuning is needed.

This paper further investigates three research questions: 1) When does the performance gap arise during fine-tuning using a labeled source language corpus? 2) Where is the most important part of the pre-trained model for achieving strong zero-shot cross-lingual performances? 3) To what extent can we reduce the performance gap by reducing the forgetting of H_{cross}^{pre} ? Based on the experiments on three datasets of different downstream tasks, our analysis found that the performance gap arises significantly in the initial fine-tuning phase and increases slowly in the later phase (see Figure 2). Feed-forward weights in the bottom four layers are the key weights for the cross-lingual knowledge (See Figure 3 and Table 1) and should be updated slowly to avoid forgetting H_{cross}^{pre} . Attention weights in the top two layers have the pre-training task (e.g., Masked-Language Modeling) knowledge H_{task}^{pre} and H_{task}^{pre} is useless for the downstream task. So these weights should be updated fast to encourage forgetting H_{task}^{pre} . We also find that protecting the cross-lingual knowledge by freezing the weights related to it can reduce the performance gap (enough stability) but cannot eliminate the gap completely (See Figure 4). That means only reducing the forgetting of H_{cross}^{pre} is not enough for solving the performance gap.

Un-forgetting vs forgetting. Based on the above analysis, we propose a method called **Fine-tuning slow and Fast algorithm** to mitigate the forgetting of cross-lingual knowledge (stability) and also to selectively forget the knowledge related to the pre-training task (plasticity) to adapt F to the downstream task in fine-tuning F . Note that traditional techniques for solving the forgetting problem in continual learning are not applicable to our setting directly (see the reasons in Sec 5).

The proposed method consists of four learning rate policies. Policies I and II (stability policies) are respectively designed to avoid forgetting of H_{cross}^{pre} in the first fine-tuning stage and to avoid

the forgetting of H_{cross}^{pre} based on the tendency of the learning curve in the second fine-tuning stage. Policies III and IV (plasticity policies) are respectively designed to selectively forget the pre-training task knowledge in H_{task}^{pre} in the initial fine-tuning stage where the loss drops drastically and to further encourage forgetting of the pre-training task knowledge H_{task}^{pre} and the learning of H_{task}^{new} in the second fine-tuning stage.

This paper’s main contributions are as follows:

(1) We analyze the performance gap in cross-lingual fine-tuning and answer to what extent we can reduce the performance gap by avoiding forgetting cross-lingual knowledge.

(2) We propose a method consisting of four learning rate policies to reduce forgetting of cross-lingual knowledge (stability) and to encourage forgetting of pre-training task-related knowledge (plasticity).

(3) We test our method in multiple datasets under zero and few-shot settings. Compared to the baseline, our method reduces the performance gap (Figure 1(XNLI) and Figure 5 in Appendix A (MLQA and NER)) and achieves better overall performance (Table 2) by protecting the cross-lingual knowledge and learning better task representation.

2 Analysis of Performance Gap and Forgetting of Cross-Lingual Knowledge in Fine-Tuning

This section studies three research questions, i.e., when $P_{\hat{s}} - P_{S/\hat{s}}$ happens and where the weights influence the overall performance mostly? It also answers to what extent we can reduce the performance gap $P_{\hat{s}} - P_{S/\hat{s}}$ by reducing the forgetting of cross-language knowledge in H_{cross}^{pre} .

2.1 Overall Setup

We directly use the multilingual pre-trained model XLM-R (Conneau et al., 2019) as the base LM due to its strong zero-shot cross-lingual transfer performance. We consider the Cross-lingual Natural Language Inference (XNLI) dataset (Conneau et al., 2018) which is a cross-lingual textual entailment dataset (classification task). Multilingual Question Answering (MLQA) dataset (Lewis et al., 2019) which is a multilingual machine reading comprehension task and NER dataset (named entity recognition task) in XTREME benchmark (Hu et al., 2020b). The metric for MLQA and NER is the F1 score and the metric for XNLI is accuracy. **All**

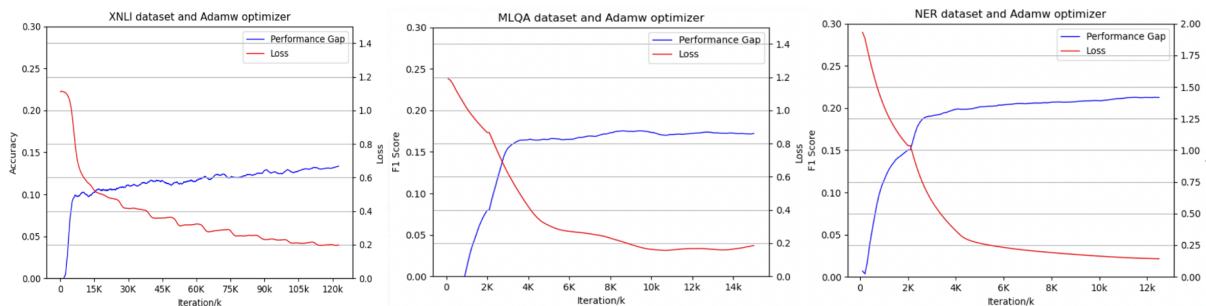


Figure 2: We record the loss and the performance gap between English and non-source languages every hundred updates over three different datasets and plot the curves in this figure.

results are the average of 4 random seeds. We use the zero-shot cross-lingual transfer setting with English as the source language for all experiments. More training details are in Appendix B.

2.2 When does the Performance Gap Arise?

We record the loss and calculate the performance gap on the validation set every hundred updates. Figure 2 shows that the occurrence of the performance gap can be divided into **two phases: (1) In the first phase P_1 (the first 20% of iterations), the performance gap occurs early and increases dramatically as the loss drops quickly in the initial training stage. (2) In the second phase P_2 (the last 80% iterations), the gap increases but is obviously slower than in the first phase and the loss drops slowly.**

2.3 Where is the Knowledge that Helps Cross-Lingual Transfer?

We use freezing and re-initializing functions to investigate the influence of the weights of each layer on overall performance. Note that we only choose one layer to do re-initializing/freezing operations in each experiment. Figure 3 shows that **the cross-lingual knowledge H_{cross}^{pre} widely exists in the first 10 layers** as re-initializing the weights in any of the ten layers causes performance drop **and is mainly located in the first four layers** as re-initializing the weights in one of the first four layers makes the performance drops obviously and freezing them boosts the performance. Also interestingly, the pre-trained knowledge in the last two layers has little influence on performance. Sometimes re-initializing one layer in the two layers even makes the performance better than the baseline performance (e.g., for the MLQA dataset). That is because the task of pre-training (e.g. Masked Language Model task) is different from the downstream task and mainly located in the last two layers. We call this kind of knowledge learned from

the pre-training task the **pre-training task knowledge H_{task}^{pre}** , which can have a negative transfer to the downstream task.

2.4 How much can We Reduce the Performance Gap by Reducing Forgetting?

To study this question, we fine-tune only the last one/two/three layers to provide strong stability for H_{cross}^{pre} . Figures 2 and 4 show that fine-tuning only the last few layers delays the first appearance of the performance gap and clearly decreases the performance gap. Also, the fewer layers are fine-tuned, the smaller the gap is. However, (1) a great gap still exists even if we only fine-tune the last layer (e.g., 9% difference on the XNLI dataset). That means **avoiding the forgetting of the pre-trained H_{cross}^{pre} can reduce the gap to some extent, but cannot solve the problem entirely.** (2) Fine-tuning fewer layers makes the overall performance drops significantly as the model does not have enough space to learn H_{task}^{new} (see Table 1). That means a smaller performance gap is not equal to better overall performance and we need to consider the plasticity too.

3 Method

This section proposes a method to avoid the forgetting of cross-lingual knowledge H_{cross}^{pre} (stability) and to encourage the forgetting of task knowledge for the pre-training task H_{task}^{pre} and learn new task’s knowledge H_{task}^{new} (plasticity). The core is to set different learning rate policies for the weights of the model based on both the layer’s location and the training phase.

3.1 Reducing Forgetting of Cross-Lingual Knowledge with Fine-tuning slow

We consider the protection of H_{cross}^{pre} first. The key challenge here is to strike a balance between

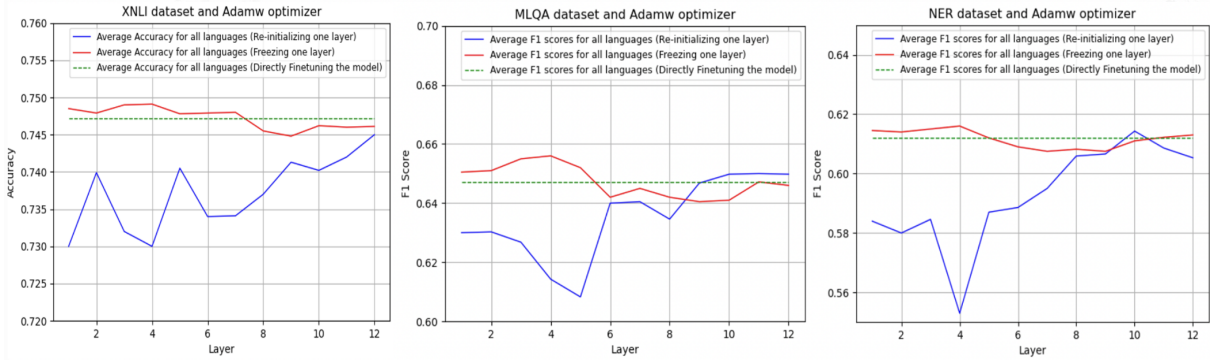


Figure 3: For all twelve layers in the pre-trained XLM-R model, we choose one layer and re-initialize its weight before training or freeze its weight during training. We record the final average performance and plot the curve. Y-axis is the metric and X-axis is the index of the layer we chose. The dotted line is the performance of directly fine-tuning model F .

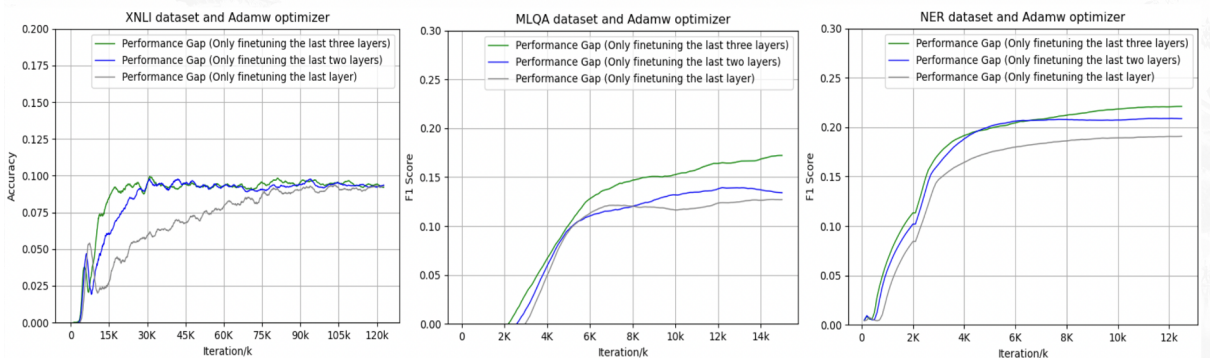


Figure 4: We fine-tune only the last one/two/three layers and record the performance gap on the validation set every hundred updates. We then plot those curves in this figure.

Dataset	Baseline	Last one	Last two	Last three	Freeze four	Freeze attention	Freeze feed-forward	Enlarge two	Enlarge attention	Enlarge feed-forward
XNLI	74.7±0.2	60.5±0.2	67.5±0.3	71.5±0.4	74.8±0.1	75.0±0.1	75.2±0.3	74.5±0.2	75.1±0.4	75.0±0.3
MLQA	64.7±0.3	33.2±0.7	48.9±0.3	43.4±0.1	52.5±0.5	64.8±0.2	66.3±0.3	66.4±0.3	66.1±0.5	65.8±0.6
NER	61.2±0.1	40.1±0.2	48.9±0.3	52.8±0.5	60.4±0.6	61.5±0.3	61.8±0.1	59.7±0.1	61.3±0.2	60.5±0.2

Table 1: Performance on XNLI, MLQA, and NER datasets in the zero-shot setting. All values are the averages of four different seeds. For the baseline, we directly fine-tune the pre-trained model. In the 'Last one/two/three' experiments, we only fine-tune the last one/two/three layers respectively. In the 'Freeze four'/'Freeze attention'/'Freeze feed-forward' experiments, we freeze the weights/attention weights/feed-forward weights in the first four layers. In the 'Enlarge two'/'Enlarge attention'/'Enlarge feed-forward' experiments, we enlarge the learning rate of the weights/attention weights/feed-forward weights in the last two layers by multiplying it with 10.

maintaining H_{cross}^{pre} and learning H_{task}^{new} . Based on the above analysis, we propose a fine-tuning slow algorithm consisting of the following two training policies and apply them to different sets.

Policy I: Avoiding drastic update of weights related to cross-lingual knowledge in the first fine-tuning phase P_1 . The performance gap increases quickly in P_1 and H_{cross}^{pre} in weights are forgetting quickly. That is because the loss in this phase drops drastically and gives a big gradient update for each weight to update, and the stability for H_{cross}^{pre} is not enough. So our goal here is to reduce the update of weights related to H_{cross}^{pre} in this stage by multiplying their learning rate with a

learning rate multiplier $K = c_1$ ($c_1 < 1$).

Policy II: Adjusting the learning rate of the key weights for cross-lingual transfer dynamically in the second fine-tuning phase P_2 . After the first phase, the gap increases slowly and our goal is to make the weights adapt to the downstream task and to avoid forgetting cross-lingual knowledge. So we set $K = 1$ for weights related to H_{cross}^{pre} to provide more plasticity and dynamically adjust the learning rate of the key weights related to H_{cross}^{pre} additionally to avoid the forgetting. Our idea for the key weights is to provide more plasticity for them when the loss drops quickly to learn a new task and to provide more stability (avoid-

ing unnecessary forgetting) when the loss drops slowly. We propose to set a dynamic multiplier K based on the learning curve for the key weights of H_{cross}^{pre} . Assume that \mathcal{L}_t is the training loss at the t th iteration and $\phi(\mathcal{L}_t) \in [0, 1]$ is a function reflecting the tendency of the learning curve. The bigger the $\phi(\mathcal{L}_t)$ is, the faster the loss drops. Then we have $K = R(\phi(\mathcal{L}_t))$, where R is a monotonic function. In this way, when the loss of the model drops quickly to adapt to the new task, K is also bigger to provide more plasticity. Note that policy II in P_2 has no conflict with policy I as the drastic loss drop in P_1 is not desirable for the weights related to H_{cross}^{pre} to adapt to the task.

Layers to apply policies I and II. If re-initializing the weights in one layer obviously drops the performance across three datasets, we denote the weights in this layer belong to S_θ^I . If freezing the weights in one layer improves the performance, we denote the weights in this layer belong to S_θ^{II} ($S_\theta^{II} \in S_\theta^I$). The latter is usually more important for cross-lingual transfer. Based on the re-initializing/freezing experiment (see Figure 3), we know that weights in the first 10 layers belong to S_θ^I and weights in the first 4 layers belong to S_θ^{II} . For policy I, we apply it to S_θ^I as we do not want to forget cross-lingual knowledge H_{cross}^{pre} due to the big gradient updates.

Attention vs Feed-forward To further investigate the best choice of protecting weights in the first four layers, we conduct experiments to freeze all weights/all weights of the multi-head layer/all weights of the feed-forward layer in the first four layers. The results in the second part of Table 1 show that freezing all weights of the feed-forward layer in the first four layers achieves the best performance over three datasets. With the additional re-initialization experiments (see Table 6 in Appendix C), we find that is because the weights of the feed-forward layer are the most important weights for cross-lingual transfer in the first four layers. So we apply policy II to the weights of the feed-forward layer in S_θ^{II} as we want to protect H_{cross}^{pre} and to provide more plasticity to learn H_{new}^{task} .

3.2 Encouraging Forgetting of Pre-training Task Knowledge with Learning Fast

As shown earlier, the **pre-training task** knowledge H_{pre}^{task} is usually useless or even harmful to the downstream task. Here we design an algorithm to utilize big gradient updates (naturally happen in the

first phase or are created by enlarging the learning rate) to encourage the model to forget H_{pre}^{task} and to learn better downstream task’s knowledge H_{new}^{task} . We refer to this as the fine-tuning fast algorithm consisting of two training policies and apply them to different sets:

Policy III: Do not slow down the update of the weights related to H_{pre}^{task} in the first fine-tuning phase P_1 . In P_1 , the model is actively looking for a point that can reduce the loss drastically and has enough energy to break the limitation of the pre-trained knowledge H_{pre}^{task} . So we allow the model to update the weights related to H_{pre}^{task} in this phase without lessening their learning rate.

Policy IV: Increasing the learning rate of the key weights related to H_{pre}^{task} in the second fine-tuning phase P_2 . In P_2 , the loss drops gradually and the model finally needs to converge to local minima. But the model may not stop learning the new task’s knowledge H_{new}^{task} . To verify this, we use the representation similarity metric CKA (Kornblith et al., 2019) to measure the similarity of the representation of the current training data batch to the pre-trained model and to the current training model. Figure 6 in Appendix D shows that the similarity of the hidden representation from the last two layers is still dropping in the second phase (except the NER dataset) and the model is striving to learn a better task representation that is different from the pre-trained one. But the loss becomes small and drops slowly in P_2 and the model doesn’t have enough energy (Pezeshki et al., 2021) to forget the pre-training task knowledge and to learn H_{new}^{task} . So if the representation similarity of the last two layers is still dropping in P_2 , we encourage the model to update the key weights relevant to the task knowledge by multiplying their learning rate with a learning rate multiplier $K = c_2$ ($c_2 > 1$).

Layers to apply policies III and IV. Based on the re-initializing experiment (Figure 2), we know that re-initializing the weights in the last two layers improves the performance or drops the performance slightly. That means that the weights in the two layers have little cross-lingual knowledge and have H_{pre}^{task} which has a negative effect on the learning of the downstream task. We denote the set of weights that has this property as V_θ^I and apply policy III to it.

Attention vs Feed-forward In the second phase, the model is trying to learn a better task representation and needs to converge to a stable point. So

enlarging the learning rate of all weights in V_θ^I may not be the best choice (e.g., disturbing the convergence). To investigate the best choice of weights in the last two layers, we conduct experiments with an increased learning rate of different weight sets. Based on the results of the third part of Table 1, we find that increasing the learning rate of all weights in the attention layer of the last two layers achieves the best performance. That implies the weights of the attention layer are the key weight in the learning of the downstream task and that not changing the learning rate of other weights in the last two layers provides much stability. So we denote the weights of the attention layer in V_θ^I as V_θ^{II} and apply policy IV to it.

3.3 Fine-tuning slow and Fast Algorithm

Formally, a typical multi-lingual pre-trained model F comprises a stack of L transformer layers with each layer containing an attention head layer l_θ^a followed by a feed-forward network l_θ^f . At the t -th training iteration of the fine-tuning process, the updating rule of the weight θ_t of model F based on our fine-tuning slow and fast algorithm is:

$$\theta_t = \begin{cases} \theta_{t-1} - K \cdot r \nabla \theta_{t-1} & \text{if } t \in P_1 \wedge \theta \in S_\theta^I \parallel \\ & \text{tin } P_2 \wedge \theta \in V_\theta^{II} \cup S_\theta^{II} \\ \theta_{t-1} - r \nabla \theta_{t-1} & \text{otherwise} \end{cases} \quad (1)$$

where r is the learning rate and $\nabla \theta_{t-1}$ is the weight modification calculated by the back-propagation algorithm. S_θ^I , S_θ^{II} , V_θ^I and V_θ^{II} are the weight sets for the application of policy I, II, III, and IV respectively. We use $t \in P_1$ to identify if the t -th iteration belongs to the first phase P_1 . The learning rate multiplier K is determined by:

$$K = \begin{cases} c_1 & \text{if } t \in P_1 \wedge \theta \in S_\theta^I \\ c_2 & \text{if } t \notin P_1 \wedge \theta \in V_\theta^{II} \\ R(\phi(\mathcal{L}_t)) & \text{if } t \notin P_1 \wedge \theta \in S_\theta^{II} \end{cases} \quad (2)$$

where c_1 and c_2 are constant and $R(\phi(\mathcal{L}_t))$ is a monotonic function based on the function $\phi(\mathcal{L}_t)$ that can reflect the tendency of the learning curve. Our method maintains the stability for cross-lingual knowledge and increases the plasticity to adapt to the new task. We verify it in the following section.

4 Experiment

We now use three downstream tasks: Named Entity Recognition (NER), Question Answering (QA), and Natural Language Inference (NLI), to experimentally evaluate the performance of our proposed

Fine-tuning slow and fast algorithm under the zero-shot and few-shot settings.

4.1 Experiment Setup

Datasets: We adopt the NER (Hu et al., 2020b), MLQA (Lewis et al., 2019), and XNLI (Conneau et al., 2018) datasets from the XTREME benchmark (Hu et al., 2020b) for NER, QA, and NLI respectively. The details of the datasets and training details are listed in Section 2.

Zero-shot and Few-shot settings. We define the zero-shot setting as fine-tuning a pre-trained model for a downstream task using its labeled data in one source language (e.g. English). Then we apply the fine-tuned model to all target languages. We define the few-shot setting as fine-tuning a pre-trained model for a downstream task using its labeled data in one source language (e.g., English) and a few labeled data from other languages. All labeled data are mixed to form a training dataset and then we use it to fine-tune the pre-trained model. For the source of the few-shot data, we split the original validation set into the few-shot data group and the new validation set. Note that the number of data points in the validation set is usually larger than 5000. So extracting the few-shot data from the validation set does not influence its validation function.

Baselines: (1) Directly Fine-tuning (DF) the model with the English training corpus; (2) Noisy-Tune (Wu et al., 2022), which prevents LMs from overfitting the data in pre-training and reducing the gap between pre-training and downstream tasks by adding a small amount of noise to perturb the LM parameters before fine-tuning. (3) Fine-tuning slow algorithm (FS), which fine-tunes the model with the fine-tuning slow algorithm. (4) Learning Fast algorithm (FF), which fine-tunes the model with the fine-tuning fast algorithm.

Choice of adaptive multiplier $R(\phi(\mathcal{L}_t))$ and $t \in P_1$, and hyper-parameters: For $R(\phi(\mathcal{L}_t))$ in Eq. 2, we first calculate the average value of the losses in the recent 100 iterations as $\frac{\mathcal{L}_{t-100:t}}{100}$ and the losses in the 100 iterations prior to the recent 100 iterations as $\frac{\mathcal{L}_{t-200:t-100}}{100}$. Then we define $\phi(\mathcal{L}_t) = \frac{\mathcal{L}_{t-100:t}}{\mathcal{L}_{t-200:t-100}}$. When the loss drops quickly ($\mathcal{L}_{t-200:t-100} \gg \mathcal{L}_{t-100:t}$), $\phi(\mathcal{L}_t)$ is close to 0. And when the loss drops slowly, $\phi(\mathcal{L}_t)$ is close to 1. We do not use $\frac{\mathcal{L}_t}{\mathcal{L}_{t-1}}$ to represent $\phi(\mathcal{L}_t)$ as the losses in adjacent iterations usually do not have a big difference and so it cannot accurately describe the tendency. Then $R(\phi(\mathcal{L}_t))$ is defined

Dataset	XNLI				NER				MLQA				
	M	0	5	10	20	0	5	10	20	0	5	10	20
DF		74.7±0.2	75.1±0.2	75.4±0.3	75.5±0.3	61.3±0.2	68.1±0.1	70.6±0.1	72.5±0.1	64.7±0.2	64.8±0.3	64.8±0.2	64.9±0.2
NoisyTune		74.9±0.2	75.1±0.1	75.5±0.2	75.6±0.1	61.3±0.1	67.8±0.3	70.7±0.2	72.7±0.2	64.8±0.2	64.8±0.2	64.9±0.3	65.0±0.2
FS		75.2±0.3	75.5±0.2	75.5±0.3	76.0±0.1	62.3±0.2	68.5±0.2	71.2±0.3	72.7±0.2	66.1±0.5	66.3±0.3	66.4±0.5	66.8±0.2
FF		75.0±0.2	75.4±0.2	75.6±0.4	75.9±0.2	62.1±0.2	68.3±0.3	70.7±0.1	72.5±0.2	66.4±0.2	66.5±0.3	66.5±0.2	66.7±0.3
Our method		75.6±0.1	75.7±0.3	76.1±0.2	76.5±0.2	62.5±0.1	69.1±0.2	71.7±0.1	72.9±0.1	66.6±0.2	66.8±0.3	66.8±0.2	67.0±0.3

Table 2: Performance on XNLI, MLQA, and NER datasets in the zero-shot and few-shot settings. All values are the averages of four different seeds. M is the number of few-shot training data for each non-source language. DF (directly fine-tuning the model), NoisyTune (Wu et al., 2022), FS (fine-tuning slow algorithm), and FF (fine-tuning fast algorithm) are the baselines.

Language	en	fr	de	avg
DF	70.2±0.2	69.1±0.5	70.0±0.1	69.7±0.2
Our method	70.5±0.2	70.4±0.1	70.7±0.2	70.5±0.2

Table 3: Accuracy performance on Large QAM dataset in the zero-shot setting. All values are the averages of four different seeds. 'avg' is the average performance over all target languages.

Method	Baseline		Our method	
	source	non-source	source	non-source
XNLI	84.8±0.3	74.0±0.3	85.6±0.2 (+0.8)	74.9±0.2 (+0.9)
NER	82.3±0.2	60.7±0.3	82.3±0.2 (+0.0)	62.0±0.1 (+1.3)
MLQA	79.4±0.3	62.2±0.2	80.5±0.2 (+1.1)	64.3±0.2 (+2.1)

Table 4: Source and non-source languages' performance in the zero-shot setting. For the baseline, we directly fine-tune the model. All values are the averages of four different seeds.

as:

$$R(\phi(\mathcal{L}_t)) = \max(1 - \phi(\mathcal{L}_t)^r, 0) \quad (3)$$

where r is a hyper-parameter. When the loss drops quickly, $R(\phi(\mathcal{L}_t))$ is close to 1 and gives the parameters more plasticity to adapt to the new task and vice versa. We choose $\frac{\mathcal{L}_{t-200:t-100} - \mathcal{L}_{t-100:t}}{100} > 0.1$ to represent $t \in P_1$ in Eq. 1 as the loss drops very quickly in this case and the model needs policies I and III to protect H_{cross}^{pre} and learn H_{task}^{new} . We set r as 3. We set c_1 and c_2 (Eq. 2) as 0.01 and 10 respectively. The ablation study is in Section 4.6.

4.2 Results of Zero-Shot Fine-Tuning

To evaluate the zero-shot performance of our method and baselines, we record the average F1 score performance (mean and standard deviation) of all target languages for the MLQA and NER datasets and the average accuracy for the XNLI dataset. The results are reported in Table 2, which shows that our method achieves highly superior results to the baseline methods.

4.3 Results of Few-Shot Fine-Tuning

The results of few-shot fine-tuning performance are reported in Table 2, which shows that:(1) with

the increasing number of few-shot data per language, the performance improves as the model can learn better cross-lingual task representation from the multi-lingual training corpus. (2) Our method still outperforms baselines obviously as it protects the pre-trained cross-lingual knowledge H_{cross}^{pre} and forgets the pre-training task's knowledge H_{task}^{pre} .

4.4 Large Training Corpus Fine-Tuning

We collect and construct a QAM task dataset (Liang et al., 2020) with 12 million English training data points by a business search engine. QAM classification task aims to predict whether a <question, passage> pair is a QA pair. Zero-shot fine-tuning model on a corpus like this is more challenging as the model is easier to forget H_{cross}^{pre} . From Table 3, we observe that our method outperforms the baseline obviously, which shows that our method also works well with a large dataset. From Figure 7 in Appendix E, we find that (1) the performance of non-source languages firstly increases and then drops (due to forgetting) (2) the performance of source language increases during the whole training process and the gap becomes larger in the later phase. (3) our method reduces the gap by protecting cross-lingual knowledge and improves the performance of non-source languages. More analyses and training details are in Appendix E.

4.5 Analysis of the Influence on Source Language and Non-Source Language

Our method improves both source and non-source languages' performance in Table 4. Additionally, it helps non-source languages more, as it protects the cross-lingual knowledge in H_{cross}^{pre} , leading to a significant improvement in non-source languages' performance. We report each language's performance on the XNLI dataset in Appendix F.

Hyper-parameter	c_1					c_2					r			
Value	0.5	0.1	0.01	0.001	0	5	10	15	20	100	1	2	3	4
XNLI	75.3±0.1	75.1±0.2	75.6±0.1	75.5±0.2	75.5±0.1	75.1±0.2	75.6±0.1	75.0±0.1	74.9±0.4	71.8±0.3	75.2±0.1	75.4±0.1	75.6±0.1	75.4±0.3
NER	62.1±0.2	62.2±0.2	62.5±0.1	62.3±0.1	62.0±0.2	-					62.1±0.1	62.3±0.1	62.5±0.1	62.2±0.2
MLQA	66.1±0.	66.5±0.1	66.6±0.2	66.3±0.3	66.3±0.3	66.1±0.2	66.6±0.2	66.4±0.1	66.0±0.2	27.3±0.7	66.4±0.3	66.4±0.2	66.6±0.2	66.2±0.2

Table 5: Performance on XNLI, NER, and MLQA datasets in the zero-shot with different hyper-parameter values of c_1, c_2 , and r . All values are the averages of four different seeds. We don’t record the performance on the NER dataset with different c_2 here. The reason is: as a low-level task, the NER task is similar to the pre-training task and its task representation does not continue to be far from the pre-trained one in the second phase (see Figure 6) and so we don’t apply policy IV on this dataset.

4.6 Analysis of the Influence of Learning Rate Multiplier Hyper-Parameters c_1, c_2 and r

We ablate on the learning rate multiplier’s hyper-parameters c_1, c_2 (Eq. 2) and r (Eq. 3). We analyze their influence by setting different values for them and recording the final overall performance. From Table 5, we observe that reducing the value of c_1 from 0.5 to 0 improves performance initially (by protecting H_{cross}^{pre}) but then decreases performance (due to lack of enough plasticity). We set c_1 as 0.01 as it strikes a balance between stability and plasticity. Increasing the value of c_2 from 5 to 100 improves performance initially (by learning better H_{task}^{new}) but then decreases performance (due to lack of stability to converge). So we set c_2 to 10. Increasing r from 1 to 4 improves performance initially (as the adaptive multiplier is closer to 1 and provides more stability) but then decreases performance (due to lack of enough plasticity). So we set r as 3. Our choice is consistent across the three datasets, showing our method’s robustness.

4.7 Analysis of the Influence of the Usage Order of Weight Sets $S_{\theta}^I, S_{\theta}^{II}, V_{\theta}^I$ and V_{θ}^{II} .

To further verify the effectiveness of our method in applying different weight sets for each policy, we conduct experiments that apply both Policies I and II to only $S_{\theta}^I/S_{\theta}^{II}$ respectively, as well as experiments that apply Policy I to S_{θ}^{II} and Policy II to S_{θ}^I . We also perform similar experiments for sets V_{θ}^I and V_{θ}^{II} . From Table 8 in Appendix G, we find that our method achieves the best performance by making a good balance between plasticity and stability. Further analyses are in Appendix G.

5 Related Work

Fine-tuning multilingual language models (MLLMs). Recent MLLM systems (e.g., mBERT (Devlin et al., 2018), mT5 (Xue et al., 2020), and XLM-R (Conneau et al., 2019)) have shown strong zero-shot transfer ability to non-source languages when fine-tuning with only a source language

corpus. However, the performance gap between the source language and non-source languages is still large. Most previous works focus on learning robust task representation (Fang et al., 2021; Zheng et al., 2021; Jiang et al., 2022) and strong pre-trained cross-lingual representation (Chi et al., 2020; Wang et al., 2020; Ouyang et al., 2020; Hu et al., 2020a). But they haven’t analyzed the performance gap in fine-tuning and the relation between forgetting and the gap. We fill this research gap.

Forgetting in continual learning. Continual learning aims to design algorithms to learn tasks incrementally. Its main challenge is forgetting. Many methods have been proposed to reduce forgetting. (1) Regularization methods (Kirkpatrick et al., 2017; Chen et al., 2020; Li et al., 2022; Lee et al., 2021) penalize the changes on important weights for previous tasks, (2) Replay methods (Buzzega et al., 2020; Rolnick et al., 2019; Wang et al., 2022) re-train a few previous samples with new task’s data, and (3) Parameter-fixed methods (Vidoni et al., 2020; He et al., 2021; Xu et al., 2021) protect parameters learned for previous tasks. The regularization method needs to estimate the important weight during pre-training and the replay method needs to store data from the pre-training corpus. But we usually don’t have those during the fine-tuning phase. Moreover, all of those methods focus on avoiding forgetting but we find that pre-training task knowledge is not suitable for the adaptation of downstream tasks and those methods cannot get rid of it. We propose a novel method that controls the forgetting effect to avoid the forgetting of cross-lingual knowledge and encourage the forgetting of pre-training task knowledge to learn better new task knowledge. Our method is orthogonal to previous works in cross-lingual fine-tuning.

6 Conclusion

This paper first analyzed when the performance gap arises and where the important cross-lingual knowl-

edge is and reduced the lower bound of the performance gap by avoiding forgetting cross-lingual knowledge. Based on our analysis, a novel method is proposed to control the forgetting effect in fine-tuning a multi-lingual pre-trained model. We verify its effectiveness over multiple datasets and settings.

7 Limitations

Although we believe that controlling the forgetting in the fine-tuning phase to avoid forgetting cross-lingual/general knowledge and to reduce the negative interference from misaligned pre-training tasks and downstream tasks can benefit other fine-tuning settings (e.g. Multi-task setting), we have not yet investigated these settings. In the future, we will try to propose a more general method for fine-tuning a large pre-trained model across various settings.

References

- Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930.
- Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651*.
- Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2020. In-foxlm: An information-theoretic framework for cross-lingual language model pre-training. *arXiv preprint arXiv:2007.07834*.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.
- Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Yuwei Fang, Shuohang Wang, Zhe Gan, Siqi Sun, and Jingjing Liu. 2021. Filter: An enhanced fusion method for cross-lingual language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12776–12784.
- Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.
- Junjie Hu, Melvin Johnson, Orhan Firat, Aditya Siddhant, and Graham Neubig. 2020a. Explicit alignment objectives for multilingual bidirectional encoders. *arXiv preprint arXiv:2010.07972*.
- Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020b. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR.
- Lan Jiang, Hao Zhou, Yankai Lin, Peng Li, Jie Zhou, and Rui Jiang. 2022. Rose: Robust selective fine-tuning for pre-trained language models. *arXiv preprint arXiv:2210.09658*.
- 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.
- Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3519–3529. PMLR.
- Seanie Lee, Hae Beom Lee, Juho Lee, and Sung Ju Hwang. 2021. Sequential reptile: Inter-task gradient alignment for multilingual learning. *arXiv preprint arXiv:2110.02600*.
- Patrick Lewis, Barlas Oğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2019. Mlqa: Evaluating cross-lingual extractive question answering. *arXiv preprint arXiv:1910.07475*.
- Dingcheng Li, Zheng Chen, Eunah Cho, Jie Hao, Xiaohu Liu, Fan Xing, Chenlei Guo, and Yang Liu. 2022. Overcoming catastrophic forgetting during domain adaptation of seq2seq language generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5454.
- Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, et al. 2020. Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018.

- Xuan Ouyang, Shuohuan Wang, Chao Pang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-m: Enhanced multilingual representation by aligning cross-lingual semantics with monolingual corpora. *arXiv preprint arXiv:2012.15674*.
- Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958.
- Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. 2021. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 34:1256–1272.
- Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for ner. *arXiv preprint arXiv:1902.00193*.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32.
- Marko Vidoni, Ivan Vulić, and Goran Glavaš. 2020. Orthogonal language and task adapters in zero-shot cross-lingual transfer. *arXiv preprint arXiv:2012.06460*.
- Liyuan Wang, Xingxing Zhang, Kuo Yang, Longhui Yu, Chongxuan Li, Lanqing Hong, Shifeng Zhang, Zhenguo Li, Yi Zhong, and Jun Zhu. 2022. Memory replay with data compression for continual learning. *arXiv preprint arXiv:2202.06592*.
- Zirui Wang, Zachary C Lipton, and Yulia Tsvetkov. 2020. On negative interference in multilingual models: Findings and a meta-learning treatment. *arXiv preprint arXiv:2010.03017*.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.
- Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. NoisyTune: A little noise can help you finetune pretrained language models better. *arXiv preprint arXiv:2202.12024*.
- Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. *arXiv preprint arXiv:2109.05687*.
- Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.
- Bo Zheng, Li Dong, Shaohan Huang, Wenhui Wang, Zewen Chi, Saksham Singhal, Wanxiang Che, Ting Liu, Xia Song, and Furu Wei. 2021. Consistency regularization for cross-lingual fine-tuning. *arXiv preprint arXiv:2106.08226*.

A Performance gap of the MLQA dataset and NER dataset

From Figure 5, we observe that (1) our method delays the first appearance of the performance gap. (2) our method has a lower performance gap compared to the baseline.

B Details of the datasets and model.

XNLI is a cross-lingual textual entailment dataset. In this dataset, we use the MultiNLI (Williams et al., 2017) training data (English) to fine-tune the pre-trained model and then test the fine-tuned model with all 15 languages.

MLQA is a multilingual machine reading comprehension task for question answering. The test performance gap is the gap between the F1 score of the source (English) and the average F1 score of the other six target languages (*Arabic, German, Spanish, Hindi, Vietnamese* and *Chinese*).

NER is a named entity recognition task and we use the Wikiann (Pan et al., 2017) dataset. The metric is the F1 score. We use the balanced train, dev, and test splits in (Rahimi et al., 2019).

Following (Hu et al., 2020b), the fine-tuning batch size is 32. We use the Adam optimizer with warm-up and learning rate $5e-6$. For XNLI, we fine-tune the model with the English corpus for 10 epochs and evaluate it on the English dev set every 3k steps to select the best model. For NER, we fine-tune 20 epochs. For MLQA, we follow BERT (Devlin et al., 2018) for SQuAD (Rajpurkar et al., 2016) and set the learning rate to $3e-5$, batch size to 12, and we train the model for 2 epochs.

We select the model with the best of the average result on the dev sets of all languages.

XLM-R has 550 million parameters and we run the experiments with GPU A100.

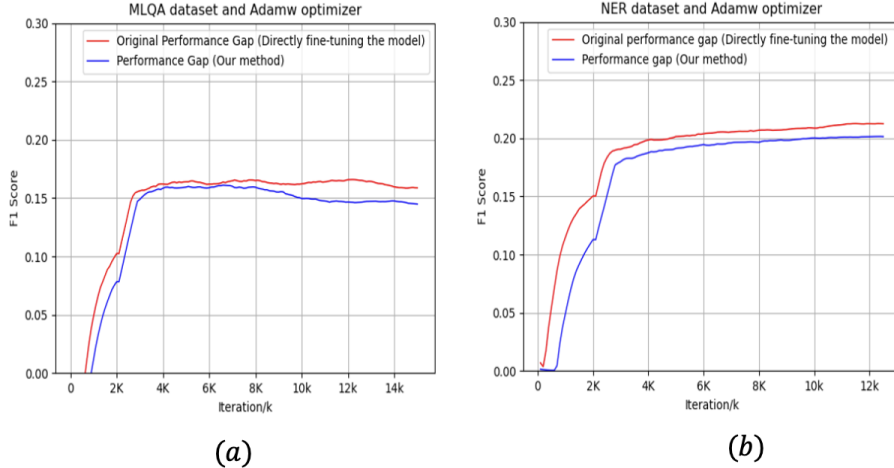


Figure 5: The performance gap $P_s - P_{S/s}$ every hundred updates on the MLQA and NER datasets. 'Original performance gap' means that we directly fine-tune the model, and 'our method' means that we use the Fine-tuning slow and fast algorithm to fine-tune the model.

Dataset	Baseline	Re-initialize four	Re-initialize attention	Re-initialize feed-forward
XNLI	74.7±0.2	64.0±0.1	68.1±0.1	67.6±0.2
MLQA	64.7±0.3	28.7±0.3	50.1±0.1	46.0±0.1
NER	61.2±0.1	45.1±0.2	52.5±0.2	45.5±0.2

Table 6: Performance on XNLI, MLQA, and NER datasets in the zero-shot setting. All values are the averages of four different seeds. For the baseline, we directly fine-tune the pre-trained model. In the 'Re-initialize four', 'Re-initialize attention' and 'Re-initialize feed-forward' experiments, we re-initialize the weights/attention weights/feed-forward weights in the first four layers and then we fine-tune the model.

C Re-initialization experiments for the weights in the first four layers

From Table 6, we find that (1) re-initializing weights in the first four layers reduces the overall performance as the cross-lingual knowledge is lost. (2) Re-initializing the feed-forward weights in the first four layers has a worse effect on the overall performance than re-initializing the attention weights in the first four layers. That means the feed-forward weights in the first four layers are more important than the attention weights in the first four layers for cross-lingual transfer.

D Figure for the CKA representation similarity

Figure 6 describes the similarity of the hidden representations from the pre-trained model and the working model. We observe that the bottom layers usually have a higher similarity than the top layers, indicating that the top layers need larger adjustments to adapt to the downstream task. Also, we find that the similarity of the last two layers continues to decrease in the second phase over the MLQA and XNLI datasets, indicating that the model is still trying to learn a better task representation by modifying the weights in the last two layers during the second phase.

E The performance analysis of the QAM dataset

We use the Adam optimizer with warm-up and learning rate $5e-6$ to fine-tune the model with the English corpus for 1 epoch as the training corpus is big enough for the model to achieve the best performance by running one epoch. The batch size is 32. We select the model with the best of the average result on the dev sets of all languages (every 3k updates).

We record the performance gap and each language's performance on the dev set every thousand updates and report the curve in the first 120k iterations (Figure 7) as the best overall performance model is selected in the first 120k iterations. In the later iterations, the gap rises and the overall performance drops. As shown in Figure 7, the tendency of the gap curve is monotonically increasing during the whole training process, and the main reason for this is the decline in the performance of the non-source languages. Our method improves overall performance by reducing the forgetting of cross-lingual knowledge.

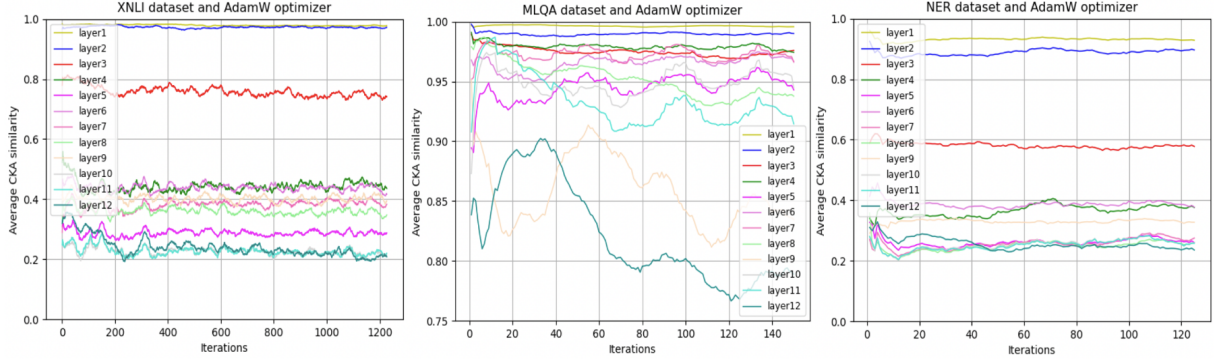


Figure 6: For the pre-trained model and the current working model, we first calculate each layer’s hidden representation on the current data batch, and then calculate and record the representation similarity of the hidden representations from the same layer in the pre-trained model and the working model every hundred updates over three different datasets. We plot the curves in this figure. A lower value (similarity) indicates a larger distance.

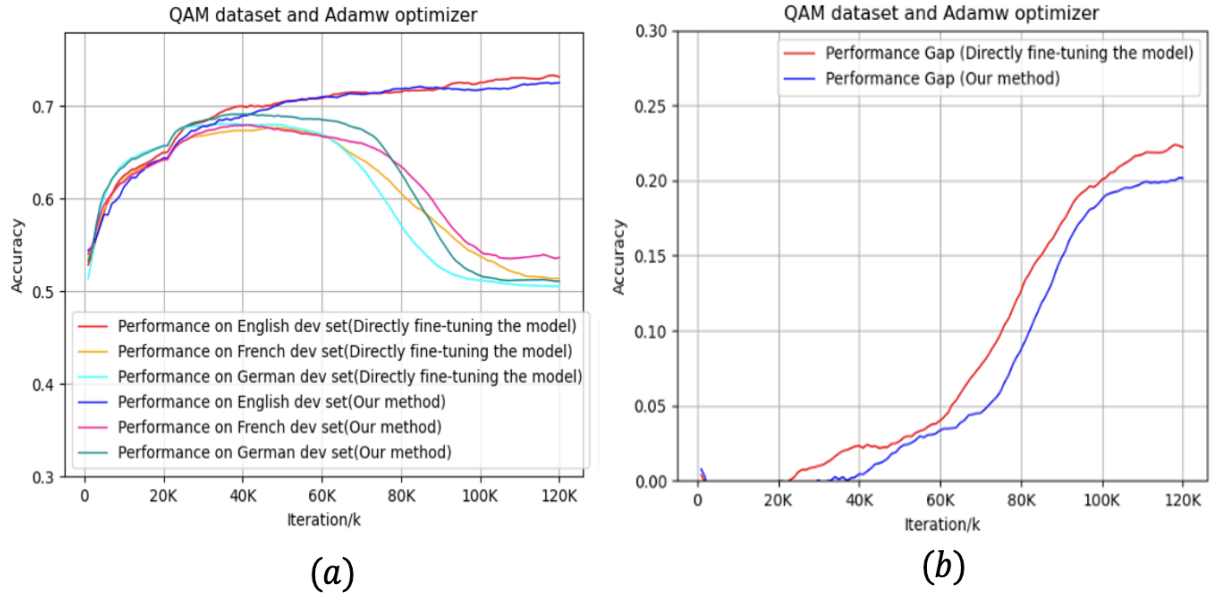


Figure 7: We directly fine-tune the model on the QAM dataset as the baseline. English is the source language. German and French are the non-source language. We calculate and record their performance on the validation set every thousand updates.

F Analysis of the influence of our method for each language

From Table 7, we find that the fine-tuning slow algorithm improves almost all languages’ performance as it protects the cross-lingual knowledge. The fine-tuning fast algorithm (learning the new task knowledge H_{task}^{new}) also improves the performance of the source language and some non-source languages as it provides a better task representation. Our method achieves the best performance over all languages, especially for some low-resource languages (e.g., sw and ur).

G Analysis of the influence of $S_{\theta}^I, S_{\theta}^{II}, V_{\theta}^I$, and V_{θ}^{II}

From Table 8, we observe that the ‘Only $S_{\theta}^I/S_{\theta}^{II}$ ’ experiments achieve worse performance than our method as they lack enough plasticity/stability for the cross-lingual knowledge, respectively. The ‘ S_{θ}^{II} to S_{θ}^I ’ experiment achieves the worst performance as it lacks enough stability in the first phase and lacks enough plasticity in the second phase. The ‘Only $V_{\theta}^I/V_{\theta}^{II}$ ’ experiments do not provide enough space for learning new tasks in the first phase/stability to converge in the second phase, respectively, so their performance is worse. The ‘ V_{θ}^{II} to V_{θ}^I ’ experiment has both disadvantages of the ‘Only $V_{\theta}^I/V_{\theta}^{II}$ ’ experiments, so its performance is

Language	ar	bg	de	el	en	es	fr	hi	ru	sw	th	tr	ur	vi	zh
DF	72.2±0.3	78.0±0.3	77.3±0.2	76.0±0.1	84.8±0.3	79.5±0.2	78.7±0.3	70.7±0.2	76.1±0.5	65.1±0.4	72.7±0.2	73.1±0.1	66.5±0.5	75.1±0.1	74.5±0.4
NosiyTune	72.2±0.3	78.0±0.3	77.3±0.2	75.6±0.1	84.8±0.3	79.5±0.2	78.7±0.3	70.7±0.2	76.1±0.5	65.1±0.4	72.7±0.2	73.1±0.1	66.5±0.5	75.1±0.1	74.5±0.4
FS	73.0±0.2	78.4±0.5	77.6±0.2	76.6±0.3	85.2±0.2	80.0±0.1	79.3±0.1	71.1±0.1	76.8±0.4	65.4±0.5	72.9±0.1	74.0±0.3	66.5±0.3	75.7±0.1	75.0±0.3
FF	72.6±0.4	78.0±0.2	77.4±0.1	76.1±0.2	85.0±0.0	79.1±0.4	78.5±0.4	70.7±0.2	76.1±0.4	65.4±0.4	73.6±0.1	73.8±0.4	67.2±0.6	75.4±0.3	74.9±0.1
Our method	73.6±0.2	79.0±0.2	78.1±0.3	76.6±0.2	85.6±0.2	80.2±0.5	79.5±0.1	71.5±0.2	77.1±0.3	65.9±0.3	73.3±0.1	74.2±0.3	67.1±0.1	76.2±0.1	75.9±0.2

Table 7: Each target language’s performance on the XNLI dataset in the zero-shot setting. All values are the averages of four different seeds.

also poor. We did not apply policy IV on the NER dataset, so we did not record the result of the V_{θ}^{II} to V_{θ}^I experiment on the NER dataset.

Dataset	Our method	Only S_{θ}^I	Only S_{θ}^{II}	S_{θ}^{II} to S_{θ}^I	Only V_{θ}^I	Only V_{θ}^{II}	V_{θ}^{II} to V_{θ}^I
XNLI	75.6±0.1	75.2±0.1	74.5±0.1	74.2±0.2	74.6±0.3	74.7±0.2	74.5±0.2
MLQA	66.6±0.2	66.2±0.2	65.9±0.3	65.1±0.1	66.3±0.3	66.2±0.2	66.3±0.1
NER	62.5±0.1	62.4±0.3	62.3±0.2	62.3±0.2	62.5±0.1	62.3±0.1	-

Table 8: Performance on XNLI, MLQA, and NER datasets in the zero-shot setting. All values are the averages of four different seeds. In the 'Only $S_{\theta}^I/S_{\theta}^I$ ' experiments, we apply both policies I and II to only S_{θ}^I /the weights of the feed-forward layer in S_{θ}^{II} respectively. In the ' S_{θ}^{II} to S_{θ}^I ' experiment, we apply policy I to the weights of the feed-forward layer in S_{θ}^{II} and apply policy II to S_{θ}^I . In the 'Only $V_{\theta}^I/V_{\theta}^{II}$ ', experiments, we apply both policy III and IV to only $V_{\theta}^I/V_{\theta}^{II}$ respectively. In the ' V_{θ}^{II} to V_{θ}^I ', experiment, we apply policy III to V_{θ}^{II} and apply policy IV to V_{θ}^I .

ACL 2023 Responsible NLP Checklist

A For every submission:

- A1. Did you describe the limitations of your work?
Section 7
- A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.
- A3. Do the abstract and introduction summarize the paper’s main claims?
Section 1
- A4. Have you used AI writing assistants when working on this paper?
Not applicable. Left blank.

B Did you use or create scientific artifacts?

Not applicable. Left blank.

- B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.
- B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.
- B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.
- B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.
- B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.
- B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C Did you run computational experiments?

Section 4

- C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix 2

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

- C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?

Appendix 2

- C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

Section 4

- C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

Not applicable. Left blank.

D Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

- D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

Not applicable. Left blank.

- D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?

No response.

- D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?

No response.

- D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?

No response.

- D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

No response.