PRESERVING IN-CONTEXT LEARNING ABILITY IN LARGE LANGUAGE MODEL FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained large language models (LLMs) are strong in-context learners that are able to perform few-shot learning without changing model parameters. However, as we show, fine-tuning an LLM on any specific task generally destroys its in-context ability. We discover an important cause of this loss, format specialization, where the model overfits to the format of the fine-tuned task and is unable to output anything beyond this format. We further show that format specialization happens at the beginning of fine-tuning. To solve this problem, we propose Prompt Tuning with MOdel Tuning (ProMoT), a simple yet effective two-stage fine-tuning framework that preserves in-context abilities of the pretrained model. ProMoT first trains a soft prompt for the fine-tuning target task, and then fine-tunes the model itself with this soft prompt attached. ProMoT offloads task-specific formats into the soft prompt that can be removed when doing other in-context tasks. We fine-tune mT5 XXL with ProMoT on natural language inference (NLI) and English-French translation and evaluate the in-context abilities of the resulting models on 8 different NLP tasks. ProMoT achieves similar performance on the fine-tuned tasks compared with vanilla fine-tuning, but with much less reduction of in-context learning performances across the board. More importantly, ProMoT shows remarkable generalization ability on tasks that have different formats, e.g. fine-tuning on a NLI binary classification task improves the model's in-context ability to do summarization (+0.53 Rouge-2 score compared to the pretrained model), making ProMoT a promising method to build general purpose capabilities such as grounding and reasoning into LLMs with small but high quality datasets. When extended to sequential or multi-task training, ProMoT can achieve even better out-of-domain generalization performance.

1 INTRODUCTION

Natural language processing (NLP) has recently been revolutionized by scaling up transformer based large language models (LLMs) together with large scale pretraining (Vaswani et al., 2017; Devlin et al., 2019; Raffel et al., 2020a; Brown et al., 2020; Rae et al., 2021; Chowdhery et al., 2022; Smith et al., 2022). In addition to improved downstream performances, these pretrained LLMs can perform a broad array of unforeseen tasks when provided with a few examples in-context. This in-context few-shot capability allows users to flexibly re-purpose LLMs for specific tasks with a minimum amount of supervised data, making it extremely convenient for fast prototyping and experimentation, especially in the low data regime.

However, even the largest and most advanced LLMs leave a lot to be desired. Grounding and eliminating hallucinations (Maynez et al., 2020), reasoning and logical clarity (Creswell & Shanahan, 2022), mathematics (Brown et al., 2020; Chowdhery et al., 2022; Noorbakhsh et al., 2021) are just a few examples where LLMs still lag behind the best human performances, or in some cases, the fine-tuned performances of the same model. While in many cases, scaling improve these qualities, it is possible that more than a few orders of magnitude in additional scale would be necessary to bridge these gaps, making a pure scaling solution impractical in the near term. In addition, it is possible that some of these properties may not improve indefinitely with larger scales. For example, Longpre et al. (2021) show that larger models have weaker context-based grounding properties as they tend to ignore the context more during reading comprehension question answering (QA).

How do we improve large language models beyond pretraining? The most common practice is to fine-tune it on a dataset specialized towards a downstream task - on most tasks, in-context learning does not achieve performances equivalent to supervised fine-tuning. As a result, the pretraining-finetuning paradigm lives on even in the era of LLMs. However, fine-tuning causes specialization. As we show in this paper, a fine-tuned model usually forgets its ability to perform unseen tasks with few-shot in-context prompts. In fact, the in-context performance could drop to none within one thousand steps of fine-tuning.

Ground-Truth Output	Mercedes' Lewis Hamilton took the outright championship lead for the first time this season with a dominant victory in the **Italian** Grand Prix.
Pretrained mT5	Hamilton won the **British** Grand Prix.
Traditionally fine-tuned mT5 on RTE	**True**
ProMoT fine-tuned mT5 on RTE	Lewis Hamilton won the **Italian** Grand Prix.

Table 1: Output comparison of pretrained and traditionally fine-tuned mT5 models vs. ProMoT fine-tuned on the RTE binary classification NLI dataset, performing an in-context 1-shot summarization task.

Loss of in-context abilities during fine-tuning is problematic when we try to improve general qualities of LLMs such as grounding, reasoning, mathematics, utilizing external resources etc. For example, grounding is desirable in a large number of tasks, including natural language inference (NLI), summarization, QA, conversation, data-to-text (Ji et al., 2022) and unforeseen tasks that could be performed in-context. There is no shortage of quality data on a few specific instantiations of these general capabilities, e.g. high quality NLI datasets that in principle could teach the model the general concept of grounding, if there was no forgetting or specialization. On the other hand, it is impossible to cover all tasks that use grounding in any fine-tuning set. Forgetting makes it difficult to utilize these high quality datasets to build general capabilities into the model.

In this work, we discover that the loss of in-context abilities is, to a large extent, caused by format specialization: overfitting to the specific task format during fine-tuning. For example, if we fine-tune the model on NLI binary classification, then it learns that the output can only be "True" or "False", making it lose its original ability to flexibly generate different output styles according to the in-context prompts of other tasks. In addition, we show that the format is usually learned first during fine-tuning, before the model fully learns the semantic content of the task.

We propose a simple solution to format specialization: PROmpt Tuning with MOdel Tuning (ProMoT). ProMoT off-loads format learning to a small amount of task-specific parameters that are external to the model. With ProMoT, we prefix a soft tunable prompt (Lester et al., 2021) to the input sequence during a 2-stage fine-tuning process: we first freeze the model and tune the prompt, then freeze the prompt and tune the model. Since format information is learned first, it mostly enters the soft prompt and is no longer required to be learned by the model. At inference time, we use the model with the tuned prompt if the inference task has the same format as the fine-tuning target task, otherwise we remove the tuned prompt and use the model directly.

Our experiments show that this simple method can significantly alleviate the forgetting on in-context abilities during fine-tuning, and also results in surprising positive transfer across tasks with totally different formats. Fine-tuning the model only on an NLI binary classification dataset, we are able to improve the general grounding properties of a mT5 XXL model, thus improving its in-context performance on summarization. See Table 1 for a concrete example.

To summarize, our contributions are 4-fold:

- We show empirically that the in-context capabilities are lost during fine-tuning, and that format specialization is one important cause for this loss. We also discover that format specialization happens at the very beginning of the fine-tuning.
- We propose a novel 2-stage fine-tuning framework: PROmpt Tuning with MOdel Tuning (ProMoT), where we utilize soft tunable prompts to absorb the task format during fine-tuning, thus alleviating in-context capabilities loss.

- We evaluate ProMoT on 10 different NLP tasks including classification, summarization, translation and question answering tasks, and compare it with prompt tuning and classical fine-tuning.
- Experimental results show the effectiveness of ProMoT: 1) ProMoT in general improves supervised performance beyond the limit of prompt tuning only. In certain tasks it matches or slightly outperforms the supervised performance of vanilla fine-tuning; 2) ProMoT significantly alleviates the forgetting of in-context capabilities on all tasks; 3) When finetuned on a single task, ProMoT results in positive task transfer across very dissimilar tasks when they share some semantic aspects; 4) When used in a sequential or parallel multi-task setting, ProMoT result in better models for most unseen in-context learning tasks. These properties makes ProMoT a promising method to build general purpose capabilities into LLMs with small fine-tuning datasets;

2 Related Work

Large language models are shown to have the abilities to do in-context learning, where they learn a new task during the inference stage by conditioning on a few training examples (Raffel et al., 2020b; Xue et al., 2020; Radford et al., 2018; Chowdhery et al., 2022). Chan et al. (2022); Gao et al. (2020) studies the effect of pretraining data distribution on in-context learning abilities on image recognition tasks, where the tension between in-context learning tasks and fine-tuning tasks is discussed. They propose changing the data distribution to ease such tension, which could be difficult for generative NLP tasks. ProMoT is an orthogonal method that does not require changes in data distribution.

Related is the notion of catastrophic forgetting, usually analyzed with a sequence of image classification tasks. Ramasesh et al. (2022) found that as model size increases, the model becomes less prone to catastrophic forgetting. However these works are mostly restricted to tasks of similar format, e.g. classification with the same number of classes. In this work we are interested in vastly different tasks, e.g. classification vs long form generation where the format itself is critical.

Instead of updating the model parameters, prompt-tuning (Lester et al., 2021; Zhang et al., 2021) appends continuous trainable embeddings (soft prompt) before the inputs and optimizes the prompt. Prompt-tuning underperforms fine-tuning in many cases, as shown in (Lester et al., 2021; Liu et al., 2021) and in our results. Prompt-tuning does not change the parameters of the pretrained model, thus cannot improve the pretrained model itself using the fine-tuning datasets. Similar as prompt-tuning, some other works propose to adapt large language models to a new task by only tuning a small set of additional parameters (Hu et al., 2021; He et al., 2021).

Another line of work uses fine-tuning to teach the model in-context few-shot or zero-shot abilities, providing a meta learning approach to in-context learning. Wei et al. (2021) fine-tunes the pretrained model on large-scale multitask datasets with diverse natural language prompts, improving the zero- and few-shot performance on unseen tasks. Min et al. (2021) incorporates the in-context learning objective into fine-tuning on multitask datasets with few-shot prompts. Such methods require fine-tuning on a large collection of different tasks to generalize, with each task being one datapoint of meta-learning. In contrast, ProMoT is not limited to this regime and shows positive task transfer with a single fine-tuning task. In addition, such approaches often require human engineered instructions or prompts for each task, a volatile process involving lots of trial and error. ProMoT does not require extensive prompt engineering as it optimizes the soft prompts with data. Finally, ProMoT is largely orthogonal to meta-learning style methods. They can be combined by applying ProMoT to a massively multitask setting where further improved generalization is expected. One could also apply ProMoT directly to the meta task of learning to solve in-context tasks. This, however, is beyond the scope of our current paper and will be left for future exploration.

3 Why are in-context abilities lost during fine-tuning?

In this section, we show empirically that 1) in-context abilities are lost during fine-tuning; 2) format specialization is an important cause for such loss; 3) format specialization happens at the very beginning of supervised fine-tuning.

3.1 LOSS OF IN-CONTEXT CAPABILITIES DURING FINE-TUNING

Pretrained LLMs achieve good performance with in-context learning. However, in many applications one may want to fine-tune LLMs. In this section, we show that the in-context learning performance usually drop significantly after vanilla fine-tuning.

To demonstrate this effect, we fine-tune a pretrained mT5 XXL model(13B parameters) (Xue et al., 2020) on the Recognizing Textual Entailment (RTE) dataset (Wang et al., 2019), which is a binary classification dataset predicting whether two given sentences are entailed. We fine-tune the model with default hyper-parameters for mT5 and task format used in PaLM (Chowdhery et al., 2022) where the output labels are True and False.

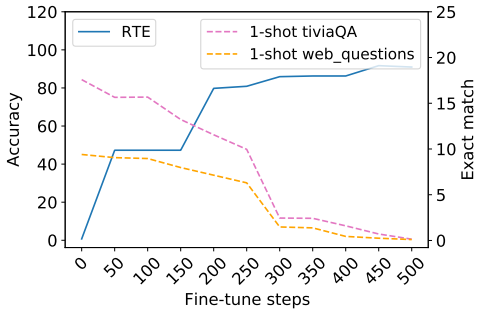

Figure 1: Learning curve of a model fine-tuned on RTE dataset for 500 steps. Left axis: Accuracy on RTE dataset. Right axis: Exact match rate on two 1-shot QA tasks.

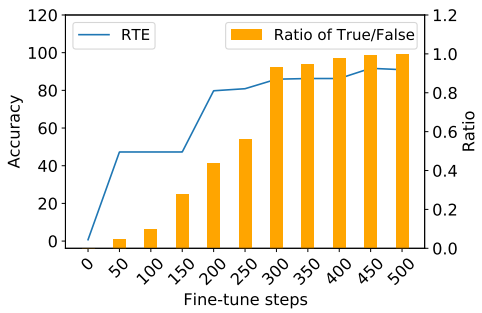

Figure 2: Change in statistics of True/False in outputs when evaluated on 1-shot in-context TriviaQA dataset. Left axis: Accuracy on RTE. Right axis: Ratio of True/False.

To probe the model's in-context capabilities during fine-tuning, we evaluate with two openbook QA tasks, trivia-qa (Joshi et al., 2017) and web_questions (Berant et al., 2013) with 1-shot in-context prompts in a closed book setting. Figure 1 shows that when the accuracy on RTE dataset increases with fine-tuning, the in-context abilities drop drastically. We show that this phenomenon is generic and not a result of specific fine-tuning or evaluation tasks with more results in Section 5.2.

3.2 FORMAT SPECIALIZATION AS A CAUSE OF FORGETTING

LLMs such as mT5 XXL have the capacity to handle a large number of tasks. So why are the in-context few-shot abilities easily lost after a few hundred steps of fine-tuning? A natural hypothesis is that due to the homogeneity of output formats in fine-tuning datasets, the model quickly learns to follow this output format no matter what the input sequence is. This leads to loss of in-context learning abilities on other tasks that do not share the format of the fine-tuned task. To be more specific, by "format" we refer to the characteristics of the fine-tuning labels as a subset of all possible output sequences. For example, the format of RTE is a set of two labels, "True" or "False", among all possible sequences of tokens of various lengths. More generally, format might include factors such as the language used, typical output lengths and styles, special tokens or punctuation, upper/lower case styles etc. that are common among most data points of a fine-tuning task. Since, by definition, most data points share the same format, the model receives a strong gradient signal that the output should follow this format independent of the input signal, thus its in-context performance on tasks with any other format will drop drastically, even if they share important semantic similarities with the fine-tuned task.

To verify this hypothesis, we fine-tune the mT5 XXL model on RTE task (classification) and then evaluate on TriviaQA task (question answering task) with 1-shot prompt. In evaluation, we count the percentage of outputs which are 'True' or 'False' in the test set. Figure 2 shows that as the fine-tuning proceeds, the model outputs more 'True' or 'False' even with a 1-shot prompted input from TriviaQA. In particular, after 300 fine-tuning steps, 90% of the output becomes 'True' or 'False'. The same phenomenon happens on other in-context tasks. For 1-shot WMT16 En-De translation, after 500 steps, more than 99% of the output becomes 'True' or 'False'. This indicates that format specialization is an important reason for loss of in-context capabilities during fine-tuning.

3.3 FORMAT LEARNING HAPPENS FIRST DURING STANDARD FINE-TUNING

Next, we show experimental evidence that format learning happens first during standard fine-tuning. This is not surprising as the overwhelming majority of fine-tuning data points has very similar formats, causing a gradient signal that dominates over other, more nuanced elements such as the semantic content of the task.

More concretely, for the RTE dataset, the format refers to the fact that output $\in \{\text{True}, \text{False}\}$, while the semantic content refers to the correlation between the input sequence and the output label. We isolate format learning from semantic learning by creating a randomized RTE dataset where the output labels are randomly shuffled, thus are no longer correlated with the input sequences. The gradients of format learning, g_{format}, are then given by the gradients on the randomized RTE dataset. By comparing with the gradient g on the original RTE we can probe as to when format learning happens during fine-tuning. As we can see from Figure 3, at the very beginning of fine-tuning (step 0), The full RTE gradient g is highly aligned with the format only gradient g_{format}, signified by $\cos(\langle g_0, g_{\text{format},0}\rangle) \approx 1$. Since randomized RTE and original RTE share the format information only and contain totally different semantic content, this alignment implies that the model is mostly learning the format. After 400 fine-tuning steps, this alignment disappears where the cosine similarity drops to around 0.2.[1] Comparison between more steps can be found in Appendix Figure 8.

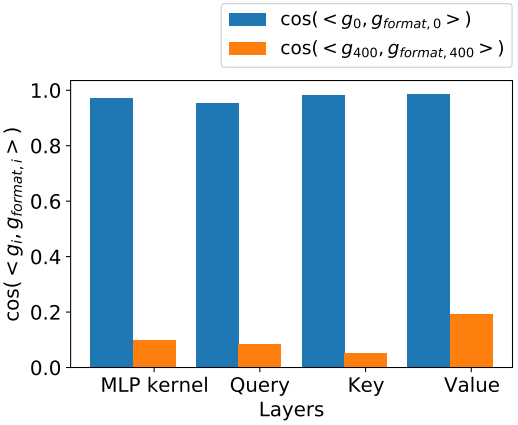

Figure 3: Cosine similarity between the full gradient g and the format gradient g_{format} on different parts of the last decoder layer. We collect and show the cosine value for gradients on MLP kernel, Query, Key and Value on the attention module. The g and g_{format} are much better aligned at the start of training, compared to at 400 steps.

4 PROPOSED METHOD: PROMPT TUNING WITH MODEL TUNING (PROMOT)

The observations from Section 3 inspire us to decompose format learning from fine-tuning, which can alleviate severe forgetting of in-context abilities during fine-tuning. The key idea is to offload format learning in a separate set of parameters external to the model during the earliest training stage, and then train the model which will now focus more on the semantic skills. We propose a two-stage fine-tune strategy called ProMoT, illustrated in Figure 4. At the first stage, ProMoT uses prompt-tuning to capture the format in a trainable soft prompt while the model itself is frozen. At the second stage, ProMoT freezes the learned soft prompt and fine-tunes the model itself to focus on semantic skills that might be more transferable.

Stage 1: Prompt-Tuning. Instead of updating the weight parameters, Prompt-tuning (Lester et al., 2021) prepends a continuous trainable prompt (soft prompt) before the embedded inputs and then updates this soft prompt during training. The soft prompt for a given fine-tuned task $P_e \in \mathbb{R}^{p \times e}$ is a small set of free parameters taking the form of a few trainable embeddings, where p is the prompt length and e is the embedding size. Given an input sequence from the task $(x_1, ..., x_n)$, prompt-tuning first embeds the input sequence with the embedding layer in the pretrained model, and then concatenates the soft prompt before the embedded input. The soft prompt is then optimized to reduce the loss while the pretrained model is frozen. As indicated in Section 3.3, fine-tuning first learns the format. We expect that by prompt tuning first, the soft prompt will learn the format. Although we cannot guarantee that the soft prompts only learns the format, the small capacity can prevent the

[1] We compute the format gradient at 400 steps, $g_{\text{format},400}$, by first fine-tuning the model on RTE for 400 steps, then computing the gradient on the randomized RTE dataset with the same batch of input sequences.

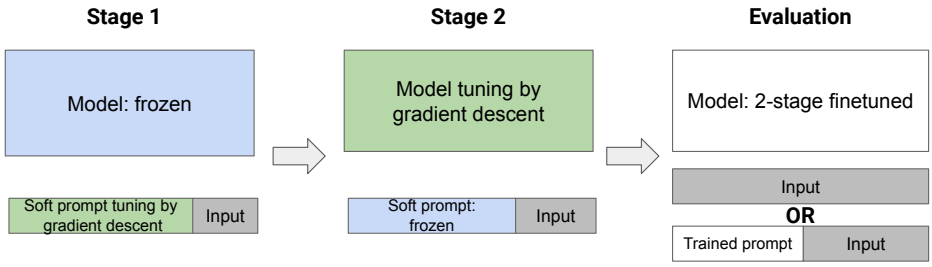

Figure 4: Overview of our two-stage fine-tuning strategy. We run prompt-tuning at Stage 1 and model fine-tuning with trained prompt at Stage 2.

soft prompt from learning all semantic skills in most realistic NLP tasks, as demonstrated by the performance gap between prompt tuning and standard fine-tuning.

Stage 2: Fine-tuning with trained prompt. After prompt-tuning, we expect the trained prompt to already store the format information. We then freeze the soft prompt and fine-tune the pretrained model. Importantly, as shown in Figure 4, the soft prompt is still prepended before the input during this stage, forcing the model to learn things not captured already by the soft prompt.

Evaluation. After the two-stage fine-tuning, we obtain a fine-tuned model checkpoint and a trained soft prompt for this specific fine-tuning target task. We expect the soft prompt stores most of the format information, and we only use this prompt during inference when the inference task has the same format as the fine-tuned target task. Otherwise, we remove the learned prompt and simply feed the original input into the fine-tuned model.

ProMoT is a general framework that can be combined with different prompt-tuning and fine-tuning techniques. For example, we can use few-shot fine-tuning at the second stage where the inputs are appended with few-shot prompts, which can further improve the few-shot performance as proposed by Min et al. (2021).

5 Experiments

5.1 Settings

Datasets. We use RTE (Wang et al., 2019) and WMT14 En-Fr as our two fine-tuning target tasks. RTE is a binary classification task and WMT14 En-Fr is a translation task. They are selected as examples of classification and generative tasks.

To evaluate in-context abilities, we use four types of tasks including natural language inference from superGLUE (Wang et al., 2019), translation from WMT, QA including triviaQA (Joshi et al., 2017) and web_questions (Berant et al., 2013) that are evaluated in a closed book setting, and summarization with XSum (Narayan et al., 2018) and WikiLingua (Ladhak et al., 2020). For each task, we use 1-shot and 4-shots in-context prompts where the prompt template is the same with PaLM (Chowdhery et al., 2022). Input templates and output post-processing can be found in the Appendix.

For classification tasks, we report the accuracies. For QA tasks, we report the exact match ratio. For translation tasks, we report the BLEU score (Papineni et al., 2002). And for summarization tasks, we report the Rouge-2 score (Lin, 2004) . We report the results on development set for classification tasks (RTE, CB and WiC) and results on test set for all other tasks.

Models. We use the mT5 (Xue et al., 2020) checkpoint for T5x XXL model (Raffel et al., 2020b) in all of our experiments, which contains 13B parameters. mT5 is pretrained on a large-scale multilingual dataset, which makes it a good choice for the translation tasks used in our experiments.

Comparing methods. In our experiments, we have several different training configurations and baselines, including

- Traditional fine-tuning: Fine-tune the pretrained model without trainable prompt.
- Prompt-tuning: Tune the trainable prompt with pretrained model frozen. The in-context learning performance of prompt-tuning is the same as the pretrained model.
- ProMoT: Our proposed two-stage fine-tuning strategy.
- Fine-tuning/Prompt-tuning/ProMoT with 1-shot: For each input/output pair during training and evaluation, we sample and prepend a 1-shot example from the training split.

Hyper-parameters. For all mT5 models, we fine-tune with learning rate 0.001, drop rate 0.1 and label smoothing 0.1. For all prompt-tuning experiments, we use learning rate 0.2. For all tasks except summarization tasks, we choose the model input sequence length larger than the input length in datasets. For summarization, we cut each input to 1024 tokens. We use Adafactor optimizer and batch size 64 without data-packing across all experiments. In inference, we use beam search to decode the outputs with width 4. More experimental settings are provided in the appendix.

5.2 SINGLE TASK FINE-TUNING

Table 2: In-context performance of model finetuned on RTE and evaluated on 8 few-shot tasks. The accuracy on fine-tuned task (RTE) is in the first row. Prompt-tuning doesn't modify pretrained model parameters and has the same in-context performance as pretrained model. We mark the lowest and highest performance for each task with red and bold numbers, respectively.

Datasets	Pretrained		Prompt-tuning		Standard Fine-tuning		ProMoT (Ours)		ProMoT + 1-shot (Ours)	
RTE	-		91.34		92.06		92.78		**93.86**	
	1-shot	4-shots	1-shot	4-shots	1-shot	4-shots	1-shot	4-shots	1-shot	4-shots
CB	46.43	51.790	46.43	51.790	73.214	82.143	66.070	67.860	**83.929**	82.143
WiC	49.687	49.687	49.687	49.687	50.000	50.157	51.411	**53.605**	51.254	50.627
triviaQA	17.582	19.022	17.582	19.022	0.150	0.115	17.643	18.660	17.820	**19.623**
web_questions	9.695	13.041	9.695	13.041	0.049	0.049	11.073	**13.189**	10.138	12.106
WMT16_ende	3.974	**8.830**	3.974	**8.830**	0.000	0.000	2.018	3.694	2.256	4.894
WMT16_enro	1.818	**3.918**	1.818	**3.918**	0.000	0.000	0.704	0.959	0.870	1.868
XSum	6.410	2.353	6.410	2.353	0.000	0.000	**7.020**	7.006	6.941	3.935
WikiLingua	4.585	1.329	4.585	1.329	0.000	0.000	4.841	**4.903**	4.874	3.434

Table 3: In-context performance of model finetuned on WMT14 En-Fr and evaluated on 8 few-shot tasks. BLEU on the fine-tuned task is in the first row. Prompt-tuning doesn't modify pretrained model parameters and has the same in-context performance as pretrained model. We mark the lowest and highest performance for each task with red and bold numbers, respectively.

Datasets	Pretrained		Prompt-tuning		Standard Fine-tuning		ProMoT (Ours)		ProMoT + 1-shot (Ours)	
WMT14 En-Fr	-		39.28		**41.796**		41.3		41.19	
	1-shot	4-shots	1-shot	4-shots	1-shot	4-shots	1-shot	4-shots	1-shot	4-shots
CB	46.430	51.790	46.430	51.790	16.071	32.143	41.071	**57.143**	41.071	53.571
WiC	49.687	49.687	49.687	49.687	**50.627**	49.060	50.157	50.313	49.843	**50.627**
triviaQA	17.582	**19.022**	17.582	**19.022**	3.200	3.147	13.630	15.204	16.927	18.191
web_questions	9.695	**13.041**	9.695	**13.041**	0.886	6.152	9.400	7.923	10.138	12.008
WMT16_ende	3.974	8.830	3.974	8.830	0.808	0.175	15.517	15.548	**16.139**	15.626
WMT16_enro	1.818	3.918	1.818	3.918	1.526	0.424	**18.544**	17.799	17.568	16.808
XSum	**6.410**	2.353	**6.410**	2.353	0.045	1.857	1.493	0.650	3.407	4.362
WikiLingua	4.585	1.329	4.585	1.329	0.030	0.429	1.142	0.524	4.215	**4.727**

We fine-tune the pretrained model with ProMoT on two target tasks and evaluate on several few-shot evaluation tasks. At ProMoT stage 1, we run prompt-tuning for 5000 steps and save a checkpoint every 1000 steps, then select the prompt checkpoint with the best performance on target task. At ProMoT stage 2, we freeze the trained prompt and fine-tune the model for 1000 steps, checkpointing every 100 steps. We then pick the model checkpoint with highest performance on the finetuned task as our final checkpoint. For comparison, we run prompt-tuning and traditional fine-tuning for 5000

and 1000 training steps respectively and report the performance of the best checkpoint. We explore finetuning with more steps in Appendix A.2.

We list the results for RTE in Table 9 and WMT14 En-Fr translation in Table 3. We see that after traditional fine-tuning on RTE, the performance on binary classification tasks (CB and WiC) with the same format may increase. However, the performance on other tasks drops to nearly zero. This further validates that fine-tuned models suffer from format specialization as discussed in Section 3. With ProMoT, the model achieves similar accuracy on RTE while forgetting much less on all few-shot evaluation tasks compared to standard fine-tuned models. More importantly, ProMoT models trained with only RTE outperform the pretrained model on summarization tasks like XSum and WikiLingua. Since ProMoT is a general framework, we can combine it with natural language few-shot prompts during fine-tuning stage, which is shown in Min et al. (2021) to improve in-context learning abilities. More concretely, training input of ProMoT + 1-shot is the concatenation of the soft prompt, the natural language 1-shot prompt, and then the original input sequence. Table 9 shows that ProMoT + 1-shot further boosts the performance over the original ProMoT on some tasks.

When we apply ProMoT finetuning on English to French translation, we find similar trend (Table 3): Compared to standard fine-tuning, we observe significantly less forgetting on in-context learning tasks across the board. Compared to the pretrained model, we observe positive transfer on translation tasks for other language pairs as well as natural language inference binary classifications.

5.3 Sequential Training and Multitask Training

As we can see from Section 5.2, ProMoT alleviates forgetting of in-context abilities across the board and has positive transfer on some tasks. It is however not surprising that ProMoT on different tasks have different influences on in-context evaluation tasks. For example, ProMoT on RTE causes some forgetting on translation while improving classification and summarization. On the contrary, ProMoT on En-Fr has some forgetting on summarization while boosting other translation tasks. This inspires us to apply ProMoT on multiple datasets for a generally better model.

In this section, we explore injecting knowledge into a single model with ProMoT by training on multiple tasks with sequential training and multitask training. For sequential training, we first run ProMoT on the 1-shot RTE dataset, and then on 1-shot En-Fr translation dataset. For multitask training, we mix the data from different tasks in one batch and train a soft prompt for each task. All the training configurations are the same in Section 5.2. We illustrate the results in Figure 5 and Table 7.

In Figure 5, we first report the accuracy on two fine-tuning target datasets RTE and WMT14 En-Fr. For sequential two-stage training, we evaluate the RTE accuracy using the trained prompt for RTE together with the final model checkpoint after sequential training. After the second ProMoT on En-Fr translation, the accuracy on RTE does not drop much, revealing the potential of our method in the continual-learning space. More importantly, after sequential ProMoT, we obtain a model with significantly improved in-context performances compared to the pretrained model on both unseen classification tasks (CB, WiC) and translation tasks (XSum, WikiLingua). The exact numbers in Figure 5 and more results on 4-shots evaluation can be found in Table 6 in the Appendix. Results for multi-task training and related analysis can be found in Appendix A.2.

5.4 Ablation Study

In this section, we conduct several ablation study in Table 4 to show the effectiveness of our method. First, we jointly fine-tune both the soft prompt and the model at the same time. This results in severe forgetting of in-context learning abilities similar to that of standard fine-tuning. We thus show that the benefit of less forgetting and positive transfer comes from the two-stage nature of ProMoT instead of merely having the soft prompt. During joint training, soft prompt and the main model are trained together for 1000 steps and then we evaluate the in-context 1-shot performance on evaluation tasks. In addition, we also fine-tune the models while attaching a random soft prompt that is fixed during training. As we expected, a random prompt contains no format information and does not help the model offload format learning, resulting in severe forgetting. Another natural baseline is to fine-tune the model with natural language prompts instead of soft prompts in place - after all, the natural language prompts contain few-shot examples that also capture the task format to some extend. We compared this approach in a 1-shot scenario and did not observe significantly reduced forgetting

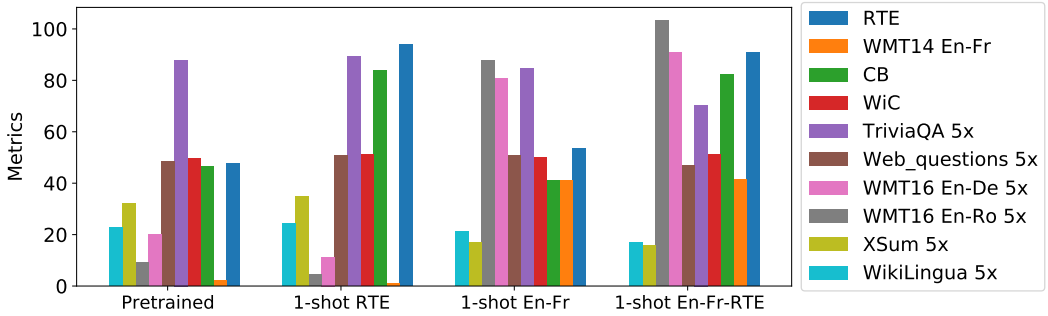

Figure 5: Results of sequential ProMoT fine-tuning compared with pretrained model and ProMoT fine-tuning on single tasks. Exact numbers and more results can be found in the Appendix. We scale up the Rouge-2 score for summarization tasks, the exact match rate for QA tasks and BLEU score for translation tasks by 5x to show all the tasks on the same figure.

Table 4: Here we show the ablation study results. Joint fine-tuning means fine-tuning the soft prompt and the main model together. Fine-tuning with 1-shot runs traditional fine-tuning on with a 1-shot natural language prompt attached to every input sequence. Fine-tuning with random prompt fine-tunes the main model with a soft prompt randomly initialized with uniform distribution and kept fixed during finetuning. We compare these abalations with ProMoT + 1-shot where ProMoT is applied on input sequences with an attached 1-shot natural language prompt.

Datasets	Joint Fine-tuning	Fine-tuning with 1-shot	Fine-tuning with random prompt	ProMoT + 1-shot (Ours)
RTE	90.97	90.97	92.06	93.86
CB	**83.929**	78.571	**83.929**	**83.929**
WiC	50.47	51.411	**51.724**	51.254
triviaQA	0.751	0.027	0.831	**17.82**
web_questions	0.64	0.000	0.295	**10.138**
WMT16_ende	0.000	0.000	0.000	**2.256**
WMT16_enro	0.000	0.000	0.000	**0.87**
XSum	0.000	0.000	0.000	**6.941**
WikiLingua	0.000	0.000	0.000	**4.874**

compared to standard fine-tuning. Most of in-context tasks still drop to zero performances. This demonstrates that soft prompts in ProMoT works better than natural language prompts in capturing format information and alleviating the forgetting of in-context learning abilities during fine-tuning.

6 Conclusions

In this paper, we found one important cause of the lose of in-context abilities during LLM fine-tuning: format specialization. Our analysis shows that format specialization happens at the very beginning of fine-tuning. This motivates us to develop ProMoT, a simple yet effective two-stage fine-tuning framework that utilizes soft trainable prompts to absorb the task format before fine-tuning the model. Our experimental results on a diverse set of NLP tasks show the effectiveness of ProMoT for preserving in-context capabilities during fine-tuning. ProMoT also shows surprising positive transfer across very different tasks, making it a promising method that could build general purpose capabilities into LLMs with small finetunig datasets. We also explored sequential training and multi-task training on two training tasks which can achieve even better generalization.

REFERENCES

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In Proceedings of the 2013 conference on empirical methods in natural language processing, pp. 1533–1544, 2013.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in NeurIPS, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Stephanie C. Y. Chan, Adam Santoro, Andrew K. Lampinen, Jane X. Wang, Aaditya Singh, Pierre H. Richemond, Jay McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers, 2022. URL https://arxiv.org/abs/2205.05055.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Antonia Creswell and Murray Shanahan. Faithful reasoning using large language models, 2022. URL https://arxiv.org/abs/2208.14271.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL, pp. 4171–4186, 2019. URL https://aclanthology.org/N19-1423.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. arXiv preprint arXiv:2012.15723, 2020.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. arXiv preprint arXiv:2110.04366, 2021.

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

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. CoRR, abs/2202.03629, 2022. URL https://arxiv.org/abs/2202.03629.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. arXiv preprint arXiv:1705.03551, 2017.

Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 4034–4048, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.360. URL https://aclanthology.org/2020.findings-emnlp.360.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pp. 74–81, 2004.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. arXiv preprint arXiv:2110.07602, 2021.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. CoRR, abs/2109.05052, 2021. URL https://arxiv.org/abs/2109.05052.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 1906–1919, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.173. URL https://aclanthology.org/2020.acl-main.173.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. arXiv preprint arXiv:2110.15943, 2021.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. arXiv preprint arXiv:1808.08745, 2018.

Kimia Noorbakhsh, Modar Sulaiman, Mahdi Sharifi, Kallol Roy, and Pooyan Jamshidi. Pretrained language models are symbolic mathematics solvers too! arXiv preprint arXiv:2110.03501, 2021.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher, 2021. URL https://arxiv.org/abs/2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR, 21(140):1–67, 2020a. URL http://jmlr.org/papers/v21/20-074.html.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21(140):1–67, 2020b.

Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=GhVS8_yPeEa.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. CoRR, abs/2110.08207, 2021. URL https://arxiv.org/abs/2110.08207.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model, 2022. URL https://arxiv.org/abs/2201.11990.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), Advances in NeurIPS, 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. Advances in neural information processing systems, 32, 2019.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. CoRR, abs/2109.01652, 2021. URL https://arxiv.org/abs/2109.01652.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. arXiv preprint arXiv:2010.11934, 2020.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. Differentiable prompt makes pre-trained language models better few-shot learners. arXiv preprint arXiv:2108.13161, 2021.

A APPENDIX

A.1 EXPERIMENT DETAILS

Input template used in experiments In Table 5, we list the natural language input template used in our experiments for each task

Task	Template
RTE	[premise] question: [hypothesis] Is it true or false? answer: {True, False}
CB	[premise] question: [hypothesis] Is it true or false or neither? answer: {True, False, Neither}
WiC	[sentence1] [sentence2] question: The word [word] is used in the same way in the two sentences. Is it true or False? answer: {True, False}
QA	Q: [question] A:
Translation	Translate [source language] to [target language]: [sentence 1]
Summarization	Article: [article] One sentence summary:

Table 5: Input template for each task

Output post-processing For each task, we first extract the text after <extra_id_0> and before <extra_id_1>, then trim the text by locating and remove the text after the second prefix token (Q:, Translate, Article:). For classification tasks including RTE, CB and WiC, we check whether the first output token is True or False.

A.2 ADDITIONAL EXPERIMENT RESULTS

More results of sequential ProMoT training In Table 6 we show the exact numbers in Figure 5 and additional evaluation results on 4-shots datasets.

Table 6: Results for sequentially training first on RTE and then on En-Fr with ProMoT + 1-shot.

Datasets	Pretrained Model		ProMoT 1-shot RTE		ProMoT 1-shot En-Fr		ProMoT on 1-shot En-Fr after RTE	
WMT14 En-Fr	1.982		0.92		41.19		41.33	
RTE	47.653		93.86		53.43		90.975	
	1-shot	4-shots	1-shot	4-shots	1-shot	4-shots	1-shot	4-shots
CB	46.43	51.79	83.929	82.143	41.071	53.571	82.143	80.357
WiC	49.687	49.687	51.254	50.627	49.843	50.627	51.097	57.367
triviaQA	17.582	19.022	17.82	19.623	16.927	18.191	14.081	16.98
web_questions	9.695	13.041	10.138	12.106	10.138	12.008	9.4	11.713
WMT16_ende	3.974	8.83	2.256	4.894	16.139	15.626	18.132	17.9
WMT16_enro	1.818	3.918	0.87	1.868	17.568	16.808	20.642	21.512
XSum	6.410	2.353	6.941	3.935	3.407	4.362	3.185	3.959
WikiLingua	4.585	1.329	4.874	3.434	4.215	4.727	3.422	3.143

Training more steps: trade-off between fine-tuning target task and in-context learning abilities In Section 5.2, we report the results of the best checkpoints within 1000 steps of fine-tuning. With a longer training period, we can see a more clear trade-off between the performance on fine-tuning target task and the performance on in-context learning abilities. Here we show the long-term trade-off between fine-tuning target task and in-context evaluation tasks by scattering the performance of different checkpoints within 20000 steps fine-tuning. In figure 6, and 7, we plot the trade-off on classification and translation tasks, respectively.

As we can see from the figures, datapoints for ProMoT is higher than traditional fine-tuning on the figures, which implies that with the same performance on fine-tuning target task, forgetting is alleviated with ProMoT fine-tuning.

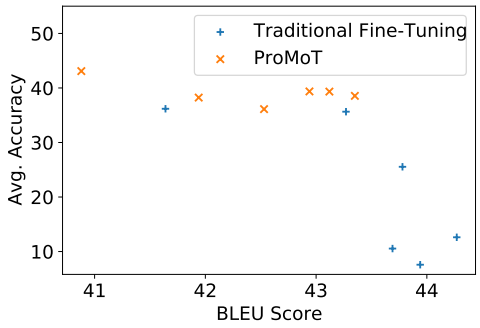

Figure 6: Trade-off between BLEU score of En-Fr (horizontal axis) and average accuracy on classification tasks (vertical axis) when fine-tuning the model on En-Fr translation.

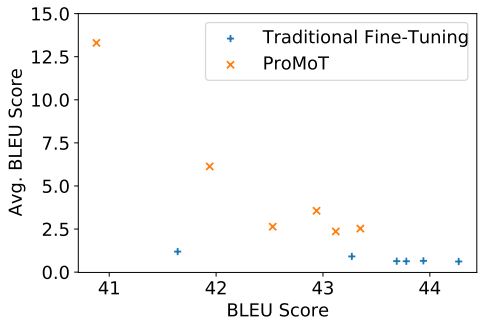

Figure 7: Trade-off between BLEU score of En-Fr (horizontal axis) and average BLEU score on other language pairs (vertical axis) when fine-tuning the model on En-Fr translation.

Multi-task training Another commonly-adopted method to train a model on multiple tasks is parallel multi-task training, especially when we known the training tasks beforehand. During multi-task training, data points from different tasks are sampled according to corresponding ratios to construct the training batches. Multi-task training has been shown to improve the generalization ability of large language models during finetuning, often with natural language prompts attached to distinguish the tasks (Sanh et al., 2021; Wei et al., 2021; Min et al., 2021). In this section, we apply ProMoT on RTE and WMT14 En-Fr in a multi-task finetuning setting. We show that ProMoT generalize substantially better on a subset of in-context tasks than traditional multi-task finetuning. At the first stage, we train a different soft prompt for each task. At the second stage, we prepend the corresponding trained and frozen prompt on each training example and tune the pretrained model on a mixture of data from different tasks. We use the same training and evaluation tasks and keep most of hyper-parameters the same except doubling training steps in the first stage. We show the results in Table 7, both on fine-tuned tasks and 1-shot out-of-domain evaluation tasks. With multi-task ProMoT, forgetting on QA and summarization are alleviated compared to sequential ProMoT. Besides, although traditional multi-task fine-tuning can improve model generalization and thus alleviate the forgetting problem on some tasks, ProMoT multi-task training has much better transferred performance on translation tasks on unseen language pairs.

Table 7: Comparison of multi-task training on RTE and WMT14 En-Fr translation. We compare the evaluation results of pretrained mT5 model, traditional multi-task fine-tuning (FT), Sequential (Seq) ProMoT training and multitask (Multi) ProMoT training. 1-shot means we add a 1-shot natural language prompt before each training input (in addition to the soft prompts if we are using ProMoT). We report performance on two fine-tuning target tasks (RTE and WMT14 En-Fr) and eight 1-shot out-of-domain evaluation tasks.

	Pretrained	multi-task FT	Seq ProMoT	Multi ProMoT	1-shot multi-task FT	1-shot Seq ProMoT	1-shot Multi ProMoT
RTE		90.250	92.419	91.340	91.700	90.975	93.140
WMT14 En-Fr		41.338	40.880	40.726	40.869	41.33	40.553
CB	46.429	80.357	82.143	83.929	87.500	82.143	85.714
WiC	49.687	51.097	51.724	51.411	53.292	50.627	52.038
TriviaQA	17.582	15.761	12.906	16.989	16.53	14.081	17.184
Web_questions	9.695	9.695	8.661	10.039	9.400	9.400	10.384
WMT16 En-De	3.974	0.882	17.028	18.833	2.503	18.132	17.572
WMT16 En-Ro	1.818	1.524	19.657	18.410	5.625	20.642	18.574
XSum	6.41	0.436	2.933	4.496	1.816	3.185	4.321
WikiLingua	4.585	0.717	2.256	2.892	4.325	3.422	3.564

Additional experiments on T5 and FLAN-T5 To show the performance of our method on English-based pretrained model, we did two additional experiments on FLAN-T5 and T5 XXL with fine-tuning target task RTE.

Table 8: In-context performance of FLAN-T5 XXL finetuned on RTE and evaluated on 8 few-shot tasks. The accuracy on fine-tuned task (RTE) is in the first row. Prompt-tuning doesn't modify pretrained model parameters and has the same in-context performance as pretrained model.

Datasets	Pretrained	Prompt-tuning	Standard Fine-tuning	ProMoT (Ours)
RTE	-	93.86	94.22	94.22
CB	83.929	83.929	85.714	83.929
WiC	61.755	61.755	67.085	62.069
triviaQA	33.926	33.926	36.241	33.926
web_questions	34.744	34.744	33.612	34.695
WMT16_ende	11.591	11.591	9.987	11.719
WMT16_enro	16.806	16.806	14.816	16.725
XSum	21.825	21.825	21.732	21.788
WikiLingua	22.858	22.858	22.778	22.848

Table 9: In-context performance of T5 XXL finetuned on RTE and evaluated on 8 few-shot tasks. The accuracy on fine-tuned task (RTE) is in the first row. Prompt-tuning doesn't modify pretrained model parameters and has the same in-context performance as pretrained model.

Datasets	Pretrained	Prompt-tuning	Standard Fine-tuning	ProMoT (Ours)
RTE	-	91.7	93.5	93.14
CB	55.357	55.357	62.500	73.214
WiC	49.843	49.843	50.000	50.784
triviaQA	34.147	34.147	0.018	33.855
web_questions	16.043	16.043	0.000	15.945
WMT16_ende	0.134	0.134	0.000	0.018
WMT16_enro	0.055	0.055	0.000	0.008
XSum	1.261	1.261	0.000	1.791
WikiLingua	1.118	1.118	0	2.251

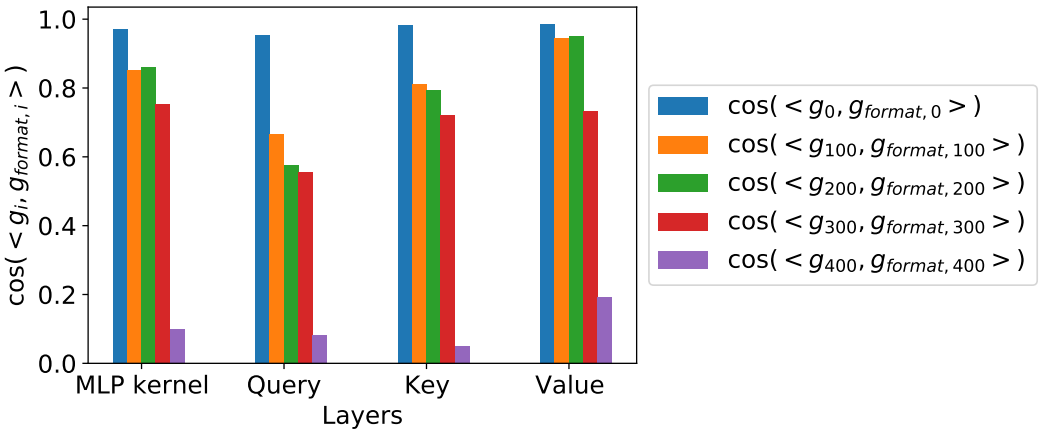

Figure 8: Cosine similarity between the full gradient g and the format gradient g_{format} on different parts of the last decoder layer. We collect and show the cosine value for gradients on MLP kernel, Query, Key and Value on the attention module.

Plotting more steps for Figure 3 To further strengthen our conclusion in Figure 3, here we plot the gradient alignment from step 0 to step 400. As we can see from the figure, gradient alignment drops significantly after 300 steps which is matched with Figure 2 where the true and false ratio increases before 300 steps and then remains stable.