Continued Pretraining for Better Zero- and Few-Shot Promptability

Zhaofeng Wu[♋] **Robert L. Logan IV**[♈]
Pete Walsh[Ω] **Akshita Bhagia**[Ω] **Dirk Groeneveld**[Ω] **Sameer Singh**[Ω♑] **Iz Beltagy**[Ω]
[♋]MIT [♈]Dataminr Inc.
[Ω]Allen Institute for Artificial Intelligence [♑]University of California, Irvine
zfw@csail.mit.edu rlogan@dataminr.com
{petew,akshitab,dirkg,beltagy}@allenai.org sameer@uci.edu

Abstract

Recently introduced language model prompting methods can achieve high accuracy in zero- and few-shot settings while requiring few to no learned task-specific parameters. Nevertheless, these methods still often trail behind full model finetuning. In this work, we investigate if a dedicated continued pretraining stage could improve "promptability", i.e., zero-shot performance with natural language prompts or few-shot performance with prompt tuning. We reveal settings where existing continued pretraining methods lack promptability. We also identify current methodological gaps, which we fill with thorough large-scale experiments. We demonstrate that a simple recipe, continued pretraining that incorporates a trainable prompt during multi-task learning, leads to improved promptability in both zero- and few-shot settings compared to existing methods, up to 31% relative. On the other hand, we find that continued pretraining using MAML-style meta-learning, a method that directly optimizes few-shot promptability, yields subpar performance. We validate our findings with two prompt tuning methods, and, based on our results, we provide concrete recommendations to optimize promptability for different use cases.

1 Introduction

Conditioning language models (LMs) on manually-written or learned continuous prompts allows them to solve tasks with high accuracy and minimal parameter overhead (Brown et al., 2020; Li and Liang, 2021; Lester et al., 2021, *i.a.*). However, prompting performance often still lags behind traditional full finetuning. Natural language prompts usually underperform trained models even when manually curated (Brown et al., 2020; Sanh et al., 2022). Similarly, while learned prompts yield higher accuracy, they do not work as well when the training data is scarce (Gu et al., 2022), when the model is small or moderately sized (Lester et al., 2021), and when the tasks are difficult (He et al., 2022).

To reduce the gap between prompt and full model tuning, past work has shown that continued pretraining on data that resembles the downstream prompting setup induces better "promptability", i.e., zero-shot performance with natural language (NL) prompts and few-shot performance of prompt tuning (Sanh et al., 2022; Gu et al., 2022). However, in this paper, we identify several shortcomings of these methods. First, continued pretraining on NL prompts (Sanh et al., 2022) sometimes causes performance degradation with prompt tuning. Second, continued pretraining approaches that learn *only* a universal prompt initialization (Gu et al., 2022; Vu et al., 2022) bring only marginal improvement on the P3 datasets (Bach et al., 2022).

To further improve zero-shot and few-shot promptability, we investigate gaps in existing methods with different parameter configurations and training procedures. First, we explore the effect of incorporating a learned continuous prompt into multi-task learning (MTL), and find it to significantly improve zero- and few-shot promptability across the board. In addition, we explore MAML-style meta-learning (Finn et al., 2017; Nichol et al., 2018) as an alternative to the standard continued pretraining paradigm, but find that it underperforms simple MTL, despite its previous success on few-shot learning tasks (Li et al., 2017; Gu et al., 2018; Qian and Yu, 2019, *i.a.*). We perform an analysis of this phenomenon and present several explanations.

Through large-scale experiments, each involving continued pretraining on over 9B tokens (§A), we make several contributions: (1) we thoroughly evaluate continued pretraining methods, both existing and our proposed ones, in many setups; (2) we demonstrate that a simple continued pretraining recipe improves over existing methods by up to

This work was done when Zhaofeng Wu was at AI2, and Robert Logan was at UCI.

We release our code and models at https://github.com/allenai/better-promptability.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 4517–4531
December 7-11, 2022

31%; (3) we show that MAML-style meta-learning underperforms multi-task learning and provide explanations; (4) we provide concrete recommendations to improve promptability in various use cases.

2 Prompting

We review two types of prompting that we use: natural language (NL) prompting and prompt tuning.

Traditionally, NLP tasks are solved by task-specific models that predict label $y \in \mathcal{Y}$ from input $x \in \mathcal{X}$. We can consider LMs as functions that score any source and target text pair, $LM : \mathcal{V}^* \times \mathcal{V}^* \rightarrow \mathbb{R}$ with vocabulary $\mathcal{V}$.[1] Past work found that large LMs can be repurposed to solve many tasks by casting x, y into a text format using a template function $f : \mathcal{X} \cup \mathcal{Y} \rightarrow \mathcal{V}^*$ and taking as prediction $\arg\max_{y' \in \mathcal{Y}} LM(f(x), f(y'))$.

NL prompts, or instructions, are manually constructed $f(\cdot)$. Without task-specific training, they have been successfully used to elicit predictions from LMs to perform tasks with high accuracy (Brown et al., 2020; Logan IV et al., 2022).

Sharing the motivation, prompt tuning learns a continuous prompt to condition the model. It takes the source text embedded by the LM input embeddings, $\mathbf{s} \in \mathbb{R}^{N \times d}$ with length N and dimension d, and prepends learnable embeddings $E \in \mathbb{R}^{L \times d}$, where L is a hyperparameter, to obtain a new $(L + N)$-lengthed embedded sequence. We consider hybrid prompt tuning, where $\mathbf{s}$ is the embedding of the templatized $f(x)$, i.e., prompt tuning is always performed *in addition to* NL templates. This has been widely adopted due to demonstrated better performance (Gu et al., 2022; Min et al., 2022). We also study a variant of prompt tuning, sometimes called prefix tuning (Li and Liang, 2021), where the learnable vectors are added not only to the input but all transformer layers. See Lester et al. (2021) and Li and Liang (2021) for more details on these methods. Following the terminology of Liu et al. (2022b), we refer to the input-level method as **shallow** prompt tuning and the layer-specific method as **deep** prompt tuning.

3 Improving Promptability

In this section, we describe existing methods to improve promptability and a new paradigm that combines their advantages.

While prompt tuning sometimes performs close to full model finetuning (Lester et al., 2021; Liu et al., 2022b), there is often still a substantial gap, such as with limited training data (Gu et al., 2022), non-gigantic models (Lester et al., 2021), or challenging tasks (He et al., 2022). We therefore study ways to improve LMs' "promptability." We focus on a low resource setup and consider zero-shot NL prompts and few-shot learned prompts (which, again, are in conjunction with NL prompts; §2). For the former, better promptability increases the performance when LMs face textual prompts of new tasks. For the latter, it more effectively leverages limited training examples for higher accuracy.

We investigate if promptability can improve with a continued pretraining stage after LM pretraining (or LM adaptation for LM-adapted T5 (Lester et al., 2021)) and before task-specific finetuning. The model is trained on a collection of tasks that have NL prompts and evaluated on unseen tasks. The methods that we explore below differ in how the continued pretraining stage is performed. We use the notation **MTL-T_P_** to abbreviate those methods that are based on multi-task learning, where the blanks _ specify different configurations of the transformer (**T**) and the prompt (**P**) components during MTL. Architecturally, a method may continue to pretrain only the T5 model without prompt parameters, in which case we use P✗ to denote the lack of them; otherwise, both transformer and prompt parameters exist during MTL. We use 🔥 and 🧊 to denote if the corresponding component is trained or frozen in MTL, respectively. This notation describes the continued pretraining stage only: in the final finetuning stage, all methods include both the transformer and prompt components, but only the latter is updated.

Continued pretraining has been studied in limited settings. Sanh et al. (2022) proposed T0 by multi-task training a T5 model (Raffel et al., 2020) as continued pretraining. They updated T5 parameters through learning on continued pretraining tasks, not including a prompt component, and showed that this training improves zero-shot NL promptability. Following our nomenclature, we refer to this paradigm as **MTL-T🔥P✗**. Additionally, Gu et al. (2022) employed a similar stage, incorporating and multi-task training a *shallow* prompt as continued pretraining, while freezing the transformer parameters in this stage. They showed that

[1] We focus on encoder-decoder LMs based on T5 (Raffel et al., 2020). Past work considers them to work better than decoder-only LMs for prompting (Sanh et al., 2022).

this strategy helps few-shot promptability during finetuning. We refer to this paradigm as **MTL-T🧊P🔥**.

In this work, we study the gains of the previous two continued pretraining approaches, as well as a model that synthesizes them, **MTL-T🔥P🔥**, which we are the first to propose. For few-shot downstream tuning, the learned prompt can act as a good initialization compared to **MTL-T🔥P✗**. In the zero-shot setup, prior work has discovered that including certain text in a prompt, such as "Let's think step by step," can adjust the reasoning of LMs to yield substantially improved performance across tasks (Kojima et al., 2022; Askell et al., 2021). The learned prompt here could function analogously. Compared to **MTL-T🧊P🔥**, on the other hand, the additional capacity brought by more updatable parameters could further boost model performance.

MAML-style meta-learning (Finn et al., 2017) directly optimizes for the downstream updates and can outperform MTL for full model finetuning (Dou et al., 2019; Bansal et al., 2020a). Yet, it similarly remains unexplored for prompting. We examine first-order MAML (**FOMAML**; Finn et al., 2017), performing T steps of prompt tuning in the inner loop and updating all parameters in the outer loop. We also evaluate a version of **Reptile** (Nichol et al., 2018) adapted for our setting that performs T steps of prompt tuning followed by one step of full model tuning, and use the resulting Reptile gradient for model updates. They have the same architecture as **MTL-T🔥P🔥** and all parameters are trainable too. We provide a detailed description and theoretical discussion of these processes in §B. See the original papers for more details.

4 Experimental Setup

We use P3, a collection of NL-templatized examples for a variety of datasets, for training and evaluation using the standard splits in Sanh et al. (2022). Not only is there no dataset overlap between training and evaluation, but no *task* overlap either (e.g., sentiment vs. QA), making it challenging. We report dataset statistics in §A. We perform continued pretraining for one epoch over all training datasets. Each dataset has multiple templates, each evaluated with accuracy. As different datasets have different numbers of answer choices and hence different baseline accuracy, we report Average Relative Gain (ARG; Ye et al., 2021) as a single summary metric by averaging across all templates the relative accuracy improvement over a random baseline. We perform significance testing using bootstrap with 1,000 iterations, in each iteration randomly sampling evaluation examples and comparing the two models in question. §D reports per-dataset results.

Following Sanh et al. (2022), we initialize the continued pretraining stage from T5 finetuned with an LM objective (Lester et al., 2021), making it more amenable to prompting. We experiment with two sizes: T5-Large with 770M parameters and T5-XL with 3B parameters. We retrain T0 (Sanh et al., 2022), i.e. **MTL-T🔥P✗**, to eliminate confounding factors in the training procedure. We also reproduce Gu et al. (2022)'s experiment in our setup, i.e. **MTL-T🧊P🔥**, pretraining a *shallow* prompt with other parameters frozen. During few-shot finetuning, we train on the same 16 examples for 100 epochs. §C reports additional hyperparameters.

5 Results

Table 1 reports our results. From **No Cont. Pretraining**, we find that continued pretraining is crucial for prompt tuning with low resources—without it, only few-shot deep prompt tuning yields slightly above-random performance. These results contradict previous findings that few-shot prompt tuning works well without this stage (Min et al., 2022). We believe this is due to the challenging nature of the P3 evaluation datasets, compared to the simple sentence classification tasks previously investigated. This is consistent with what He et al. (2022) observed in the full-data setting where deep prompt tuning performs sub-optimally on difficult tasks.

Existing methods for continued pretraining have their drawbacks. In contrast to Gu et al. (2022), we found that **MTL-T🧊P🔥** with a shallow prompt does not substantially perform above random. We attribute this to (1) their simpler evaluation tasks which, unlike ours, have decent prompt-tuned performance without continued pretraining; and (2) their hand-designed pretraining tasks that match their evaluation tasks, while P3 conversely avoids training-evaluation task overlap, requiring generalizability. Vu et al. (2022) also found **MTL-T🧊P🔥** to be effective, though with high resources. We also compare with T0, i.e. **MTL-T🔥P✗**, where both the official model and our reproduction suffer from degraded performance when few-shot shallow prompt tuned (compared to 0-shot), likely because the prompt added during finetuning is intrusive, and the limited gradient updates are not sufficient to re-

	T5-Large (770M)				T5-XL (3B)			
	Shallow		Deep		Shallow		Deep	
	0-shot	16-shot	0-shot	16-shot	0-shot	16-shot	0-shot	16-shot
No Cont. Pretraining (LM-adapted T5)	–1.9*	–1.5	–1.9*	2.3	–1.5*	–1.5	–1.5*	3.7
Previous methods								
MTL-T🧊P🔥 (Ours à la Gu et al. (2022))	2.8	2.9	—	—	–1.7	–1.6	—	—
MTL-T🔥P✗ (Sanh et al., 2022)	—	—	—	—	25.2*	19.1	25.2*	33.8
MTL-T🔥P✗ (Our reproduction)	26.7*	23.1	26.7*	32.6	32.4*	30.0	32.4*	42.9
Our methods								
MTL-T🔥P🔥	**30.2**	**30.2**	**28.7**	**37.4**	**33.5**	**33.5**	**33.3**	**43.2**
FOMAML	21.8	21.8	20.8	32.8	27.7	27.7	24.5	40.0
Reptile	18.4	18.4	24.9	33.0	23.1	23.2	26.3	39.7

Table 1: Average Relative Gain (ARG) on the P3 evaluation datasets. Random performance is 0.0 ARG. Each row in *Our methods* represents four pretrained models, for Large/XL × Shallow/Deep, while **MTL-T🔥P✗** shares the pretrained model for Shallow and Deep as it has no pretrained prompt. We **bold** the highest number in each column. *: These models have no prompt and hence no Shallow/Deep distinction in 0-shot experiments.

cover from it. We note that the official T0 model is not well-optimized: even without hyperparameter tuning, our implementation is significantly better ($p < 0.001$ for all).

MTL-T🔥P🔥 significantly outperforms **MTL-T🔥P✗**, the strongest existing method we examine, across all settings ($p < 0.005$ for all) except for few-shot deep prompt tuning on T5-XL ($p = 0.21$). For zero-shot NL promptability, the improvement could be due to the extra model capacity, or the multi-task trained prompt adjusting the reasoning of the LM, analogous to the text-based "Let's think step by step" effect (Kojima et al., 2022). For few-shot shallow prompt tuning, unlike **MTL-T🔥P✗**, **MTL-T🔥P🔥** does not degrade in performance, resulting in 31% higher ARG than **MTL-T🔥P✗** on T5-Large. This is likely because of the model's familiarity with the prompt, though the limited capacity of shallow prompt tuning does not yield benefits either. Nevertheless, with deep prompt tuning, which gives the model sufficient conditioning capacity, few-shot tuning does lead to performance increase, again outperforming **MTL-T🔥P✗**. Here, **MTL-T🔥P🔥** provides a good prompt initialization and alleviates its intrusiveness. These results emphasize the importance of continued pretraining being aware of the downstream finetuning process. Interestingly, however, the gap between these two models shrinks as the model size increases, no longer significant at T5-XL ($p = 0.21$). Also, notably, pretraining with a shallow prompt has better 0-shot performance than a deep prompt. This highlights that higher pretraining capacity is not always beneficial, and matches our motivation from text-based conditioning which also happens at the input level.

FOMAML and Reptile surprisingly underperform **MTL-T🔥P🔥** in few-shot prompt tuning, even though they specifically optimize for this procedure and have demonstrated success in NLP for full model finetuning (Dou et al., 2019; Bansal et al., 2020b, 2021, *i.a.*) and few-shot learning (Gu et al., 2018; Qian and Yu, 2019; Mi et al., 2019, *i.a.*). While Ye et al. (2021) also found FOMAML to underperform MTL, they sub-optimally only performed one inner loop update. Here, we show that this comparison holds for more appropriate hyperparameters. This could be due to the fewer number of gradient updates: to perform one gradient update, MTL uses one training batch, while FOMAML with T inner loop steps or Reptile with T prompt tuning steps use $T + 1$ batches. Not only might this be an inefficient use of training examples, but compute FLOPs too, since each inner loop/prompt tuning step involves a full forward-backward pass. We attempt using a $T + 1$ times smaller meta batch size (see §B for more detail) to pretrain a deep T5-Large-sized Reptile. When prompt-tuned, it achieves 22.8 ARG, which is even lower, possibly due to higher gradient estimation noise. Alternatively, other factors could affect the performance of meta-learning. It is, for example, well-known that MAML-style meta-learning can be unstable and

sensitive to architectures and hyperparameters (Antoniou et al., 2019). This instability is likely amplified by our large heterogeneous multi-task setup and our inability to afford hyperparameter search. Furthermore, its theoretical foundation has mostly only been examined through simple optimizers, predominantly SGD (Finn et al., 2017; Nichol et al., 2018). How it interacts with optimizers more common in modern NLP, such as Adafactor (which we use), remains to be explored.

Recommendations. Based on our findings, we recommend practitioners to always incorporate a prompt during continued pretraining and to train the entire model. Without downstream task-specific tuning, such as when there is no training data or sufficient compute, a shallow prompt yields better accuracy. When few-shot task-specific prompt tuning is affordable, continued pretraining with a deep prompt enables the best performance.

6 Conclusion

We demonstrated that the simple recipe of continued pretraining with a prompt significantly improves zero-shot NL promptability and few-shot learned promptability. MAML-based meta-learning, on the other hand, obtains worse performance, for which we provided several explanations. Nonetheless, we believe future efforts to leverage their conceptual advantage could be fruitful, perhaps aided by our observations. We also hope to study the effect of continued pretraining with other parameter injection methods (Houlsby et al., 2019; Hu et al., 2022; Liu et al., 2022a).

Limitations

Due to the expensive nature of our experiments, each involving continued pretraining on over 9B tokens (§A), we could not afford to perform hyperparameter tuning, and instead took hyperparameters from prior work. It is, nevertheless, possible that careful hyperparameter tuning might yield slightly different trends from what we observed. Furthermore, because of computational constraints, we were unable to perform experiments on the largest released T5 model with 11B parameters. Though we validated our findings on two model sizes, it has been found that larger models sometimes demonstrate qualitatively different results (Srivastava et al., 2022; Lampinen et al., 2022; Wei et al., 2022). We would be excited to see if our experiments could be reproduced at a larger model scale.

Acknowledgments

We appreciate Victor Sanh, Albert Webson, Colin Raffel, Zaid Alyafeai, and other members of the Bigscience project who answered many of our questions when we reimplemented T0, and Yuxian Gu when we reimplemented Gu et al. (2022). We also thank Sébastien M. R. Arnold whose help was crucial for our meta-learning implementation. We are also grateful for the members at AI2 and UC Irvine, as well as the anonymous reviewers for their valuable feedback.

References

Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2019. How to train your MAML. In *International Conference on Learning Representations*.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment.

Stephen Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-david, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Fries, Maged Alshaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-jian Jiang, and Alexander Rush. 2022. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104, Dublin, Ireland. Association for Computational Linguistics. Accessed from `https://huggingface.co/datasets/bigscience/P3`.

Trapit Bansal, Karthick Prasad Gunasekaran, Tong Wang, Tsendsuren Munkhdalai, and Andrew McCallum. 2021. Diverse distributions of self-supervised tasks for meta-learning in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5812–5824, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020a. Learning to few-shot learn across diverse natural language classification tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5108–5123, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020b. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534, Online. Association for Computational Linguistics.

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. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1192–1197, Hong Kong, China. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners.

Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. 2022. Can language models learn from explanations in context?

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-sgd: Learning to learn quickly for few-shot learning.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Robert Logan IV, Ivana Balazevic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2022. Cutting down on prompts and parameters: Simple few-shot learning with language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2824–2835, Dublin, Ireland. Association for Computational Linguistics.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3151–3157. International Joint Conferences on Artificial Intelligence Organization.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330, Dublin, Ireland. Association for Computational Linguistics.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2639–2649, Florence, Italy. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal 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 Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Jones, Joshua B. Tenen-

baum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Madotto Andrea, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramón Risco Delgado, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Debnath Shyamolima, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Timothy Telleen-Lawton, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhao Xinran, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. 2022. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7163–7189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Input: Number of inner loop steps T, meta batch size B
Initialize LM-adapted T5 parameters ϕ^{orig}
global_grad = **0**
for $b \leftarrow 1, \cdots, B$ **do**
  ϕ = clone(ϕ^{orig})
  for $t \leftarrow 1, \cdots, T$ **do**
    data = next_batch() `// support`
    grad = forward_backward(ϕ, data)
    update(ϕ, grad, prompt_only=True)
  data = next_batch() `// query`
  grad = forward_backward(ϕ, data)
  global_grad += grad
global_grad = global_grad / B
update(ϕ^{orig}, global_grad, prompt_only=False)

Algorithm 1: The FOMAML algorithm for prompt tuning.

Input: Number of inner loop steps T, meta batch size B, inner loop learning rate α
Initialize LM-adapted T5 parameters ϕ^{orig}
global_grad = **0**
for $b \leftarrow 1, \cdots, B$ **do**
  ϕ = clone(ϕ^{orig})
  for $t \leftarrow 1, \cdots, T$ **do**
    data = next_batch() `// support`
    grad = forward_backward(ϕ, data)
    update(ϕ, grad, prompt_only=True)
  data = next_batch() `// query`
  grad = forward_backward(ϕ, data)
  update(ϕ, grad, prompt_only=False)
  global_grad –= ϕ
global_grad = global_grad / (αB) + ϕ^{orig}
update(ϕ^{orig}, global_grad, prompt_only=False)

Algorithm 2: The Reptile algorithm for prompt tuning.

A Dataset Details

We use P3 as our training and evaluation datasets (Bach et al., 2022). It contains 35 datasets grouped into 8 tasks: Multiple-Choice QA, Extractive QA, Closed-Book QA, Sentiment, Topic Classification, Structure-To-Text, Summarization, and Paraphrase Identification. Examples in each dataset are templatized using multiple human-written templates. Across the 35 datasets, there are a total of 313 templates. For continued pretraining, we follow Sanh et al. (2022) and only use the training split of each dataset. Four tasks are held out for evaluation in P3: Sentence Completion, Natural Language Inference, Coreference Resolution, and Word Sense Disambiguation. They consist of 11 evaluation datasets (considering the three splits of ANLI as separate datasets) and 116 templates in total. We use the training split of each dataset for few-shot experiments, and, following Sanh et al. (2022), evaluate on the validation splits. The only exception is StoryCloze which does not have a training split, so we use its validation split for training and evaluate on its test split. Unlike T0, we do not evaluate on the BIG-Bench datasets (Srivastava et al., 2022) as they had not stabilized as a collection of datasets at the time of this work. All the prompts in P3 are collected in English.

To make training more efficient, we right-truncate all source sequences to 768 tokens and target sequences to 192 tokens. For the continued pretraining stage, this affects 2% of all training examples, and among the 313 templates, 24 have more than 1% examples truncated. Also, following Sanh et al. (2022), we cap all datasets to have a maximum of 500k examples. This results in 31.6M training examples across all datasets and templates, totaling 5.7B tokens on the source side and 3.5B tokens on the target side.

B Meta-Learning Details

In this section, we elaborate on our meta-learning training procedures. Algorithm 1 contains pseudocode for our first-order MAML (FOMAML) procedure. In the inner loop, we perform T steps of prompt tuning on a cloned model using support data. In the outer loop, we use query data to evaluate the prompt-tuned model and compute gradients. We use the first-order approximation where the gradient is not taken with respect to the entire prompt tuning process but only the forward pass with query data because it is computationally more tractable, and past work has shown that this first-order approximation does not hurt performance much, if at all (Finn et al., 2017; Dou et al., 2019). Theoretically, to perfectly simulate the downstream prompt tuning procedure, we should use the same batch of support data for the T steps of update. Nevertheless, this would traverse the training data much more slowly, so we use different support batches. Our theoretical analysis through the perspective of Reptile below also justifies this.

In preliminary experiments, we found a naïve

adoption of Reptile (Nichol et al., 2018) to yield subpar performance. As there is no inner- and outer-loop distinction in Reptile, doing prompt tuning leads to only the prompt parameter being updated throughout the entire continued pretraining stage, likely causing the performance degradation. This effect is also seen in our multi-task learning setup with the **MTL-T🧊P🔥** model. Thus, we propose to adapt Reptile to better suit prompt tuning, which we illustrate in Algorithm 2. It is similar to FOMAML, but instead of considering the outer loop's gradient as the meta-learning gradient, it uses $\frac{1}{\alpha B}\sum_{b=1}^{B}(\phi^{\text{orig}} - \phi^b)$ where b is the cloned model's final parameter for meta-batch b.

Now we theoretically justify our proposed Reptile version, mostly following the original proof structure in Nichol et al. (2018), in the context of prompt tuning. We can think of the downstream finetuning stage as starting from the initial model parameters ϕ_0 and performing T steps of prompt tuning on the same batch of training data which produces a loss function $L_{\text{train}}(\phi)$ for some model parameters ϕ. Then this model is evaluated on some test data that similarly produces a loss function $L_{\text{test}}(\phi_T)$ for the final trained model ϕ_T. Let us first abbreviate some gradients and Hessians:

$$\begin{aligned} g_i &= L'_{\text{train}}(\phi_i) = \frac{\partial}{\partial \phi_i} L_{\text{train}}(\phi_i) \\ H_i &= L''_{\text{train}}(\phi_i) \\ \overline{g} &= L'_{\text{test}}(\phi_0) \\ \overline{H} &= L''_{\text{test}}(\phi_0) \end{aligned}$$

Then we can write each step of the prompt tuning process as an update function:

$$\begin{aligned} U(\phi) &= \phi - \alpha \mathbf{m} \circ L'_{\text{train}}(\phi) \\ U'(\phi) &= I - \alpha \mathbf{M} \circ L''_{\text{train}}(\phi) \end{aligned}$$

where α is the learning rate and $\mathbf{m}$ and $\mathbf{M}$ are boolean masks that contain 1 for the prompt parameters. $\circ$ indicates element-wise multiplication, which we prescribe to take the highest precedence in the equations below.

With T iterations of U, we have:

$$\phi_T = \phi_0 - \alpha \mathbf{m} \circ \sum_{j=0}^{T-1} g_j \tag{1}$$

Plugging in Equation 1, by Taylor's theorem:

$$\begin{aligned} g_i &= L'_{\text{train}}(\phi_i) \\ &= L'_{\text{train}}(\phi_0) + L''_{\text{train}}(\phi_0)(\phi_i - \phi_0) + \mathcal{O}(\alpha^2) \\ &= g_0 + H_0(\phi_i - \phi_0) + \mathcal{O}(\alpha^2) \\ &= g_0 - \alpha H_0 \mathbf{m} \circ \sum_{j=0}^{i-1} g_j + \mathcal{O}(\alpha^2) \\ &= g_0 - \alpha i H_0 \mathbf{m} \circ g_0 + \mathcal{O}(\alpha^2) \end{aligned} \tag{2}$$

where the last step can be seen by induction, iteratively applying the second-to-last line.

With a similar process, we can derive:

$$H_i = H_0 + \mathcal{O}(\alpha) \tag{3}$$

The FOMAML gradient is the same as $L'_{\text{test}}(\phi_T)$. Plugging in Equation 2 but sweeping its non-leading terms into $\mathcal{O}(\alpha^2)$:

$$\begin{aligned} g_{\text{FOMAML}} &= L'_{\text{test}}(\phi_T) \\ &= L'_{\text{test}}(\phi_0) + L''_{\text{test}}(\phi_0)(\phi_T - \phi_0) \\ &\quad + \mathcal{O}(\alpha^2) \\ &= \overline{g} + \overline{H}(\phi_T - \phi_0) + \mathcal{O}(\alpha^2) \\ &= \overline{g} - \alpha \overline{H} \mathbf{m} \circ \sum_{j=0}^{T-1} g_j + \mathcal{O}(\alpha^2) \\ &= \overline{g} - \alpha T \overline{H} \mathbf{m} \circ g_0 + \mathcal{O}(\alpha^2) \end{aligned}$$

The full MAML gradient takes the derivative throughout the entire prompt tuning process. Plugging in Equation 3 and $g_{\text{FOMAML}} = L'_{\text{test}}(\phi_T)$ and sweeping terms into $\mathcal{O}(\alpha^2)$ when possible:

$$
\begin{aligned}
g_{\text{MAML}} &= \frac{\partial}{\partial \phi_0} L_{\text{test}}(\phi_T) \\
&= \frac{\partial}{\partial \phi_0} L_{\text{test}}(U(U(\cdots U(\phi_0)))) \\
&= U'(\phi_0)U'(\phi_1)\cdots U'(\phi_{T-1})L'_{\text{test}}(\phi_T) \\
&= \left(\prod_{j=0}^{T-1}(I - \alpha \mathbf{M} \circ L''_{\text{train}}(\phi_j))\right) L'_{\text{test}}(\phi_T) \\
&= \left(\prod_{j=0}^{T-1}(I - \alpha \mathbf{M} \circ H_j)\right) L'_{\text{test}}(\phi_T) \\
&= \left(I - \alpha \mathbf{M} \circ \sum_{j=0}^{T-1} H_j\right) L'_{\text{test}}(\phi_T) \\
&\quad + \mathcal{O}(\alpha^2) \\
&= \left(I - \alpha \mathbf{M} \circ \sum_{j=0}^{T-1} H_j\right) (\overline{g} - \alpha T \overline{H} \mathbf{m} \circ g_0) \\
&\quad + \mathcal{O}(\alpha^2) \\
&= (I - \alpha T \mathbf{M} \circ H_0)\,(\overline{g} - \alpha T \overline{H} \mathbf{m} \circ g_0) \\
&\quad + \mathcal{O}(\alpha^2) \\
&= \overline{g} - \alpha T \mathbf{M} \circ H_0 \overline{g} - \alpha T \overline{H} \mathbf{m} \circ g_0 \\
&\quad + \mathcal{O}(\alpha^2)
\end{aligned}
$$

The Reptile gradient, in our adaptation, takes the prompt's gradient during the T steps and the entire model's gradient for one step. Taking the Reptile gradient from Algorithm 2 and using ϕ_{T+1} to represent the parameters after the outer loop full-finetuning update:

$$
\begin{aligned}
g_{\text{Reptile}} &= \frac{1}{\alpha}(\phi_0 - \phi_{T+1}) \\
&= \mathbf{m} \circ \sum_{j=0}^{T-1} g_j + g_{\text{FOMAML}} \\
&= \mathbf{m} \circ \sum_{j=0}^{T-1} (g_0 - \alpha j H_0 \mathbf{m} \circ g_0) \\
&\quad + g_{\text{FOMAML}} + \mathcal{O}(\alpha^2) \\
&= T\mathbf{m} \circ g_0 - \alpha \frac{T(T-1)}{2} \mathbf{m} \circ (H_0 \mathbf{m} \circ g_0) \\
&\quad + g_{\text{FOMAML}} + \mathcal{O}(\alpha^2) \\
&= \overline{g} + T\mathbf{m} \circ g_0 \\
&\quad - \alpha \frac{T(T-1)}{2} \mathbf{m} \circ (H_0 \mathbf{m} \circ g_0) \\
&\quad - \alpha T \overline{H} \mathbf{m} \circ g_0 + \mathcal{O}(\alpha^2)
\end{aligned}
$$

We can see that all three meta-learning gradients have a similar effect: they only contain a mixture of lone gradients terms ($\overline{g}, g_0$), which act as a pure multi-task learning objective, and terms that are Hessian times gradient, which Nichol et al. (2018) termed "AvgGradInner" and showed to encourage the expected similarity between different data batches, improving generalization.

Back to our use of different data batches in FOMAML's inner loop and Reptile's prompt tuning steps. If the inner loop uses the same support (i.e., training) data, as in the derivation above, the "AvgGradInner" terms become somewhat degenerate, with the same term scaled T or $\frac{T(T-1)}{2}$ times. With different inner loop batches, on the other hand, there would be more diverse Hessian-gradient interactions between different batches of data and hence encouraging generalization between more tasks.

C Training Details

Due to the expensiveness of our experiments, we did not perform any hyperparameter tuning. For all continued pretraining runs, we follow Raffel et al. (2020) and Sanh et al. (2022) and use Adafactor (Shazeer and Stern, 2018) with a 0.001 learning rate. We use a batch size of 4,096 which we calculated to be close to what Sanh et al. (2022) used.[2] We clip gradients to unit norm. For shallow prompt tuning, we follow Min et al. (2022) and use $L = 20$ prompt tokens, each with the same dimension as the word embedding size, on the source side only. For deep prompt tuning, we similarly use 20 hidden vectors that are prepended in every transformer layer, on both the source and target side for added capacity. For meta-learning, we use a batch size of 16, simulating our 16-shot evaluation (see below), and a meta batch size of 128. We perform 7 steps of inner loop updates (FOMAML) / prompt tuning (Reptile), following Bansal et al. (2020b) and Bansal et al. (2021), and similarly using Adafactor with learning rate 0.001. All continued pretraining experiments run for one epoch over the training datasets with no checkpoint selection. In few-shot finetuning, we train on one batch of 16 randomly selected examples for 100 epochs (the same batch throughout training), following Min et al. (2022). Like Min et al. (2022), we do not manually balance the label distribution in these examples, unlike in prior work (Gao et al., 2021; Logan IV et al., 2022).

[2] They used example packing by putting multiple examples in one sequence to better suit TPUs, which we didn't use, so the batch sizes are not perfectly comparable.

We perform all experiments on 80GB A100 GPUs. Each continued pretraining run takes four (sometimes eight) of them. The largest MTL model takes 10 days to pretrain with four GPUs, while the largest meta-learning model takes 14 days.

D Per-Dataset Results

In Figures 1 to 3, we compare the per-dataset *accuracy* of **MTL-T🔥P✗** (our reproduction), **MTL-T🔥P🔥**, **FOMAML**, and **Reptile**. We omit **MTL-T🧊P🔥** due to its near-random performance.

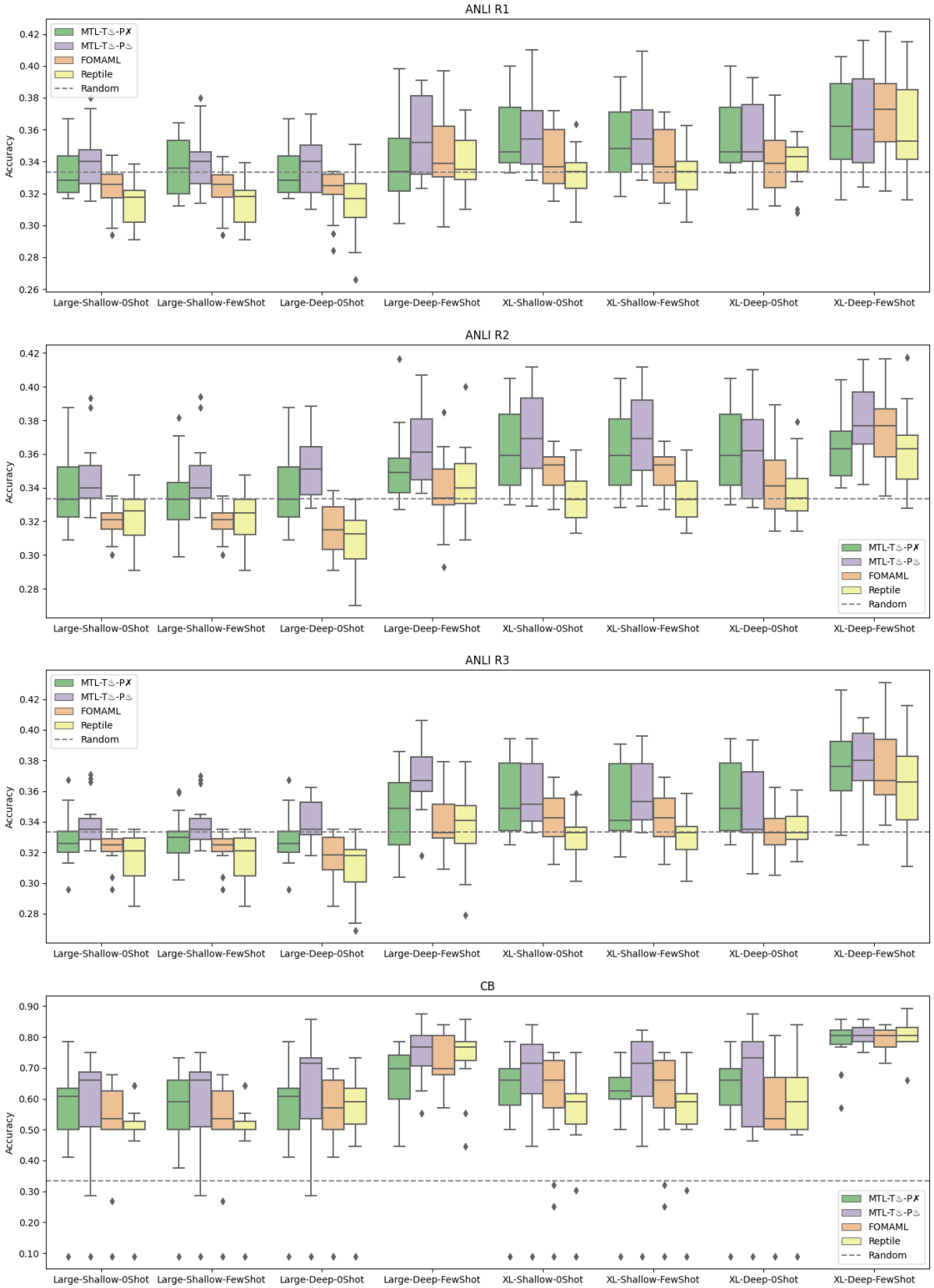

Figure 1: Per-dataset accuracy of **MTL-T🔥P✗** (our reproduction), **MTL-T🔥P🔥**, **FOMAML**, **Reptile**, and the random baseline (part 1).

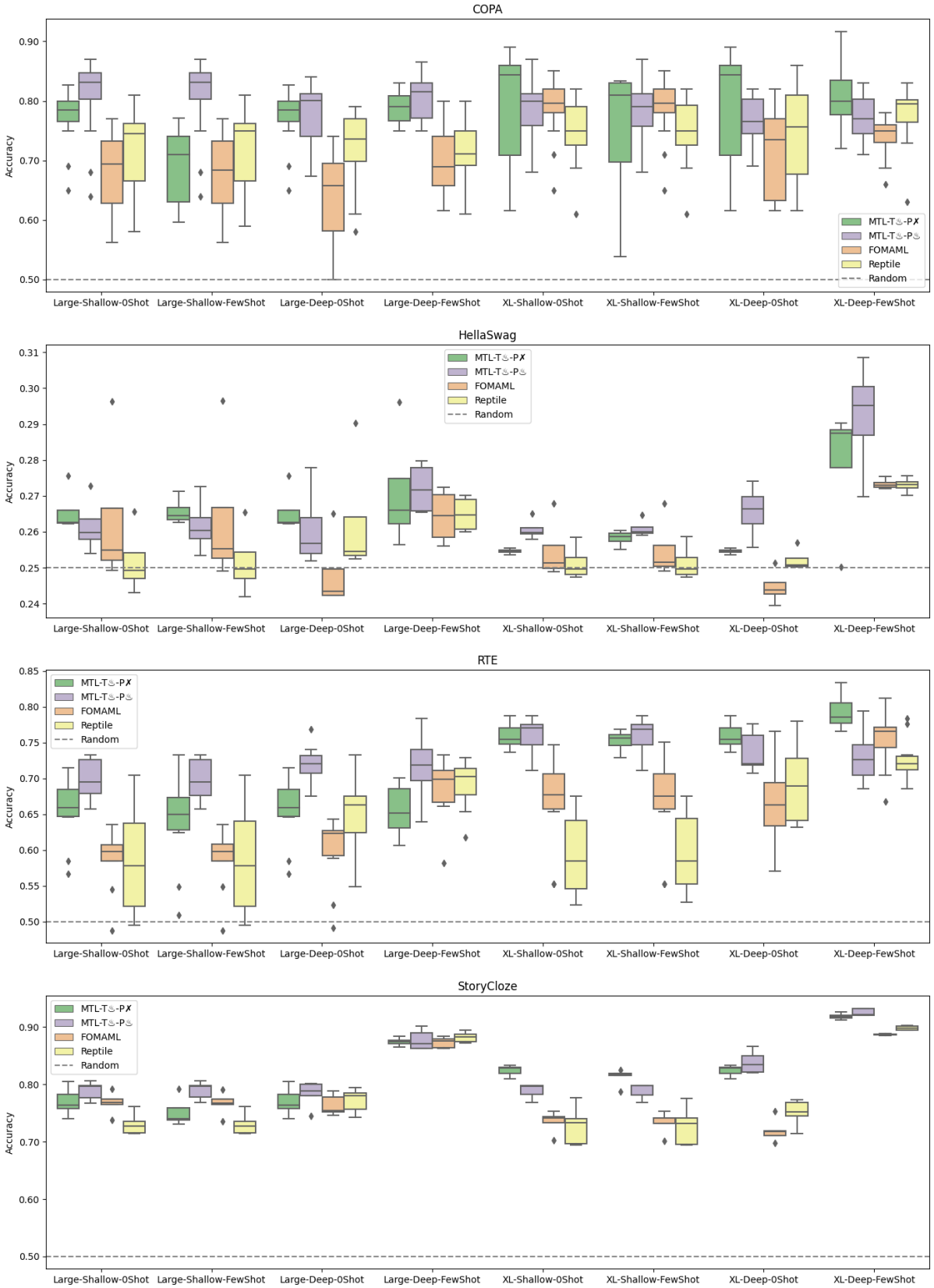

Figure 2: Per-dataset accuracy of **MTL-T🔥P✗** (our reproduction), **MTL-T🔥P🔥**, **FOMAML**, **Reptile**, and the random baseline (part 2).

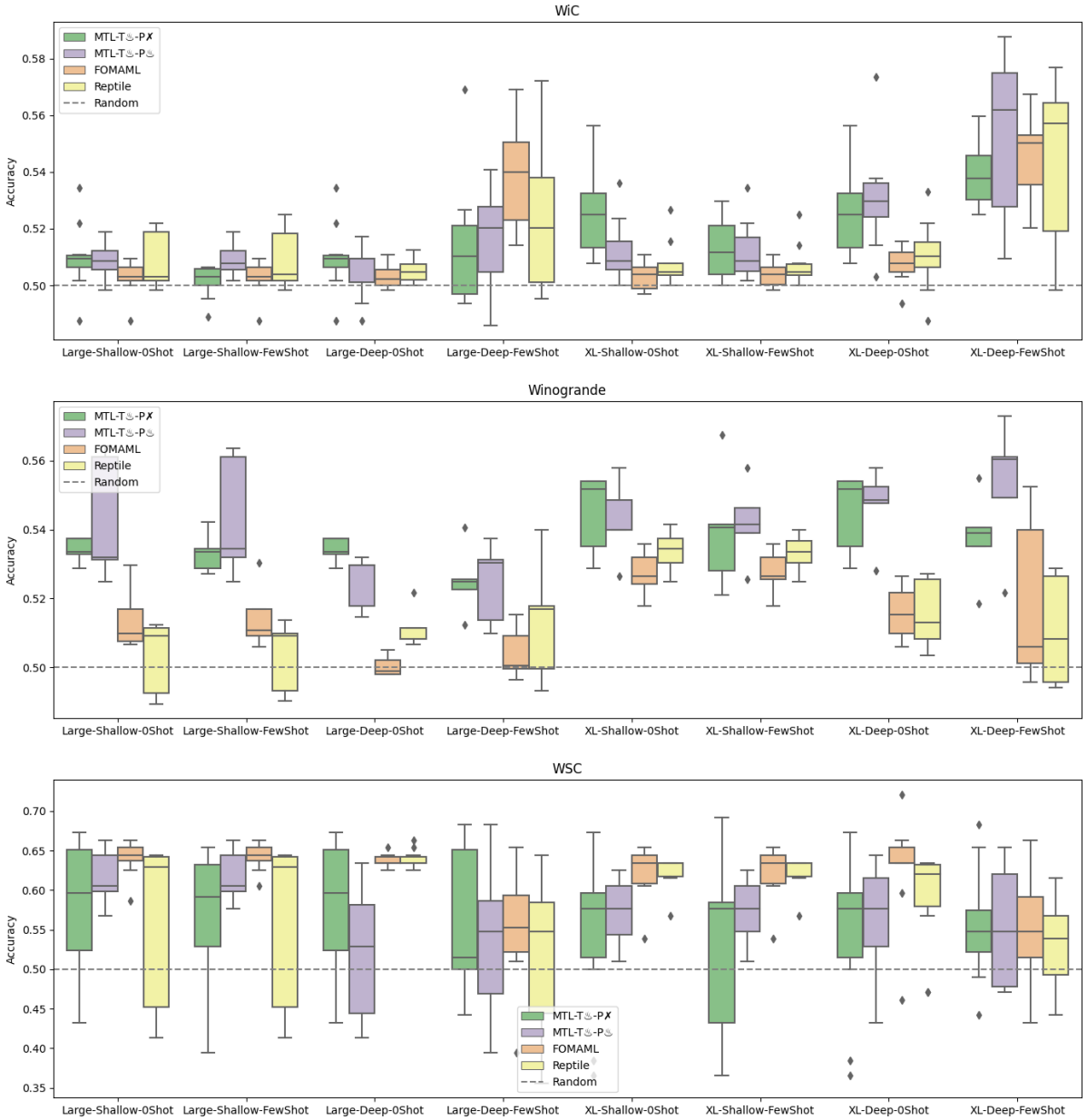

Figure 3: Per-dataset accuracy of **MTL-T🔥P✗** (our reproduction), **MTL-T🔥P🔥**, **FOMAML**, **Reptile**, and the random baseline (part 3).