Context Tuning for In-Context Optimization

Jack Lu [1] **Ryan Teehan** [1] **Zhenbang Yang** [1] **Mengye Ren** [1]

Abstract

We introduce *Context Tuning*, a simple and effective method to significantly enhance few-shot adaptation of language models (LLMs) without fine-tuning model parameters. While prompt-based adaptation techniques have demonstrated the effectiveness of lightweight adaptation methods for large language models (LLMs), they typically initialize a trainable prompt or prefix with irrelevant tokens for the task at hand. In contrast, *Context Tuning* initializes the trainable prompt or prefix with task-specific demonstration examples, leveraging the model's inherent In-Context Learning (ICL) ability to extract relevant information for improved few-shot learning performance. Extensive evaluations on benchmarks such as CrossFit, UnifiedQA, MMLU, BIG-Bench Hard, and ARC demonstrate that *Context Tuning* outperforms traditional prompt-based adaptation methods and achieves competitive accuracy to Test-Time Training with significantly higher training efficiency.

1 Introduction

Large language models (LLMs) excel on diverse natural language processing (NLP) tasks by leveraging knowledge from large-scale pretraining (Brown et al., 2020; Grattafiori et al., 2024; Jiang et al., 2023). These models can adapt to new tasks using only a few input and output examples provided in context, a process known as In-Context Learning (ICL) (Brown et al., 2020). However, ICL often struggles with complex reasoning or domain shifts, as it relies solely on a forward pass to interpret the examples. While methods like Test-Time Training (TTT) (Akyürek et al., 2024) improve adaptation with limited data, they can be computationally expensive. This motivates the development of more efficient and effective few-shot adaptation techniques for LLMs.

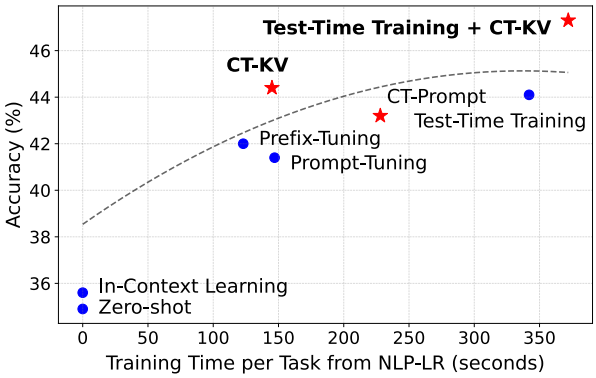

Figure 1: Comparison of training-free, prompt-based adaptation, and *In-Context Optimization* methods on solving 26 NLP-LR tasks from Table 1. Circles are baselines; stars are our methods; **bolded** methods attain the best performance-efficiency tradeoff.

Contrary to ICL's reliance on a forward pass, prompt-based adaptation methods like Prompt Tuning (Lester et al., 2021) and Prefix Tuning (Li & Liang, 2021) prepend trainable vectors to each input and optimize them via gradient descent. At a conceptual level, ICL harnesses the model's ability to extract task-relevant information from the few-shot context, while prompt-based adaptation methods optimize randomly initialized vectors to guide the model's behavior in solving each example. Given these complementary approaches, it is natural to ask whether we can bridge them by directly optimizing the context of few-shot examples to steer the model more effectively.

In this work, we introduce *Context Tuning*, a simple and effective method for few-shot learning that initializes trainable vectors from the few-shot examples of a novel task and optimizes them to solve each example. We develop two variants: *CT-Prompt* applies Prompt Tuning to a soft prompt initialized from few-shot examples, and *CT-KV* applies Prefix Tuning to optimize the key-value (KV) cache derived from those same examples. While *CT-Prompt* achieves strong performance, it suffers from a quadratic training time to the number of examples. Similarly, the recently proposed Test-Time Training (TTT) (Akyürek et al., 2024) method, which fine-tunes model parameters with LoRA (Hu et al., 2022) on permutations of few-shot examples, also incurs

[1]New York University. Correspondence to: Jack Lu <yl11330@nyu.edu>, Mengye Ren <mengye@nyu.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

quadratic cost. In contrast, *CT-KV* achieves linear training time while also outperforming *CT-Prompt* and achieving competitive performance to TTT, thanks to the efficiency and per-layer conditioning of the KV cache. In addition, because *Context Tuning* tunes the context and TTT tunes the model, the two methods are complementary: applying *CT-KV* to refine the model context after TTT's weight updates leads to additional performance gains. A high-level comparison in Figure 1 illustrates *CT-KV*'s high efficiency and accuracy, whether used alone or in combination with TTT.

We situate two approaches for few-shot learning with in-context examples – TTT that optimizes the model itself, and *Context Tuning* that optimizes the model's context – within a broader framework we term *In-Context Optimization* (*ICO*). Under *ICO*, adaptation leverages ICL and either updates its parameters or its context representation. We evaluate ICL, prompt-based adaptation methods, and *ICO* techniques across a wide range of benchmarks. *CT-KV* significantly outperforms both ICL and prompt-based adaptation methods, while remaining competitive with the more computationally intensive TTT. Furthermore, we show that *CT-KV* can serve as a post-hoc refinement step following TTT, leading to improved few-shot adaptation performance compared to either method used in isolation.

2 Background

We introduce the definition of ICL, Prefix Tuning, and Prompt Tuning. For single-task few-shot adaptation, we consider a language model p_ϕ with parameters ϕ, d hidden dimensions, L layers, a demonstration set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^k$, and the goal of solving a new query x_q from the same task. We denote the concatenated context of all demonstration pairs as $\mathcal{C} = [x_1; y_1; \ldots; x_k; y_k]$.

In-Context Learning. ICL concatenates all k demonstration pairs followed by the query x_q. The model then predicts $\hat{y}_q$ conditioned on this context:

$$\hat{y}_q = \arg\max_y \; p_\phi\big(y \mid [\,\mathcal{C}; x_q\,]\big).$$

In ICL, there is no gradient-based optimization; instead, the model adapts by attending to the tokens of the demonstration pairs provided in context.

Prompt Tuning. In Prompt Tuning, the model parameters ϕ remain fixed. Instead, m trainable soft prompt tokens P are prepended to each input and optimized via gradient descent:

$$P^* = \arg\min_P \sum_{i=1}^{k} -\log p_\phi\big(y_i \mid [\,P; x_i\,]\big).$$

After optimizing on the demonstration pairs, the optimized soft prompt P^* can be used for inference:

$$\hat{y}_q = \arg\max_y \; p_\phi\big(y \mid [\,P^*; x_q\,]\big).$$

Prefix Tuning. Prefix Tuning also keeps ϕ fixed but learns layer-wise prefixes of m trainable vectors for the keys and values in each transformer layer: $\Theta = \{K_j, V_j\}_{j=1}^L$. Each layer's attention uses these prefixes by prepending K_j to its keys and V_j to its values. Prefix Tuning's optimization and inference equations are analogous to those of Prompt Tuning, but with Θ instead of P.

3 Context Tuning for In-Context Optimization

In this section, we introduce the mathematical formulation of *In-Context Optimization* (*ICO*), a few-shot adaptation scheme that uses demonstrations in the context and performs gradient-based optimization on either the model parameters or a context representation. We then show that Test-Time Training (TTT) (Akyürek et al., 2024) is an instance of *ICO*. Finally, we present Context Tuning, formalizing its *CT-Prompt* and *CT-KV* variants along with the two additional design choices that drive their strong performance.

3.1 In-Context Optimization

ICO unifies two prevalent techniques for few-shot learning: ICL and gradient-based optimization. Formally, *ICO*'s optimization objective is to minimize the loss

$$\sum_{i=1}^{k} -\log p_\phi\left(y_i \mid [\,\theta^{(i)}_{\text{context}}; x_i\,]\right), \tag{1}$$

where $\theta^{(i)}_{\text{context}}$ is a context representation derived from the set of demonstration pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^k$. While Prompt Tuning and Prefix Tuning also prepend additional contexts during optimization, their contexts are randomly initialized instead of utilizing the demonstration pairs $\mathcal{D}$. Therefore, they are not instances of *ICO*. ICL is also not *ICO* by not performing gradient-based optimization.

3.2 Test-Time Training as ICO

TTT (Akyürek et al., 2024) can be viewed as an instance of *ICO*. Specifically, TTT minimizes Equation 1 by first initializing the model weights ϕ from a pretrained model, then updating them with LoRA layers for parameter efficiency. At each optimization iteration, TTT dynamically sets $\theta^{(i)}_{\text{context}} = \mathcal{C}^{-i}$ where $\mathcal{C}^{-i}$ represents the concatenated tokens of a random permutation of demonstration pairs except for the i-th pair. The optimization equation becomes:

$$\phi^* = \arg\min_\phi \sum_{i=1}^{k} -\log p_\phi\left(y_i \mid [\,\mathcal{C}^{-i}; x_i\,]\right).$$

To perform inference on the query input x_q, TTT uses the optimized model weights and the concatenation of all demonstration pairs as context:

$$\hat{y}_q = \arg\max_y \; p_{\phi^*}\left(y \mid [\,\mathcal{C}; x_q\,]\right).$$

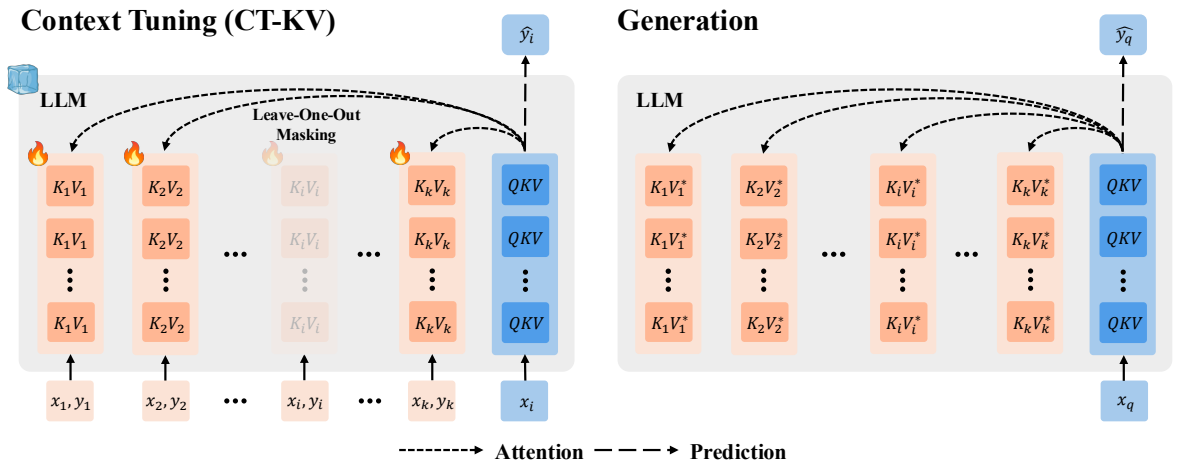

Figure 2: *CT-KV*, the variant of *Context Tuning* that optimizes the key-value prefixes derived from in-context demonstration pairs. *CT-KV* (left) first initializes a prefix $\{K_i, V_i\}_{i=1}^k$ from demonstration pairs $\{(x_i, y_i)\}_{i=1}^k$, then trains it to solve each pair. To prevent the model from simply retrieving the demonstration pair from the prefix, Leave-One-Out Masking prevents the model from attending to K_i, V_i when solving pair i. At generation time (right), the model conditions on all optimized prefixes $\{K_i^*, V_i^*\}_{i=1}^k$ to solve query x_q.

3.3 Context Tuning

We design our *Context Tuning* approach to be an instantiation of the *ICO* framework. In contrast to TTT, *Context Tuning* freezes model parameters ϕ and instead directly optimizes the lightweight context representation θ_{context} of the demonstration pairs.

- *CT-Prompt* initializes $\theta_{\text{context}} = P_{\text{CT}}$ as the model's prompt embeddings on $\mathcal{C}$, the concatenation of demonstration pairs.
- *CT-KV* initializes $\theta_{\text{context}} = \Theta_{\text{CT}}$ as a key-value prefix $\Theta_{\text{CT}} = \{K_j, V_j\}_{j=1}^L$ obtained from the model's layer-wise activations on $\mathcal{C}$.

We also introduce two design choices for *Context Tuning*.

Leave-One-Out Masking. To prevent the model from simply retrieving the answer y_i of the ith demonstration pair embedded in θ_{context} when predicting the output for x_i, we construct

$$\theta_{\text{context}}^{(i)} = \begin{cases} P_{\text{CT}}^{-i} & \text{for } \textit{CT-Prompt}, \\ \Theta_{\text{CT}}^{-i} & \text{for } \textit{CT-KV}, \end{cases}$$

and use it instead of θ_{context} in optimization. When conditioning on P_{CT}^{-i} or Θ_{CT}^{-i}, the trainable soft prompt tokens in *CT-Prompt* or prefix tokens in *CT-KV* corresponding to the in-context demonstration pair (x_i, y_i) are masked out from the attention view of the model.

Token Dropout. Since *Context Tuning* generally introduces a larger number of prompt or prefix tokens than traditional prompt-based adaptation techniques, we regularize training by randomly dropping tokens in $\theta_{\text{context}}^{(i)}$ with a fixed probability, denoted as TokenDrop.

Altogether, we arrive at the optimization (top) and inference (bottom) equations for *CT-KV*:

$$\Theta_{\text{CT}}^* = \arg\min_{\Theta_{\text{CT}}} \sum_{i=1}^{k} -\log p_\phi \left(y_i \mid [\,\text{TokenDrop}\left(\Theta_{\text{CT}}^{-i}\right); x_i] \right),$$

$$\hat{y}_q = \arg\max_y p_\phi \left(y \mid [\,\Theta_{\text{CT}}^*; x_q\,] \right).$$

CT-Prompt's optimization and inference equations are analogous to *CT-KV* but with P_{CT} instead of Θ_{CT}. In practice, *CT-Prompt* requires recomputing layer-wise keys and values corresponding to P_{CT}, while *CT-KV* does not for Θ_{CT}. In the Appendix, we formally prove that for each optimization step, *CT-KV* has lower time complexity than both TTT and *CT-Prompt* with respect to the number of demonstration pairs. Finally, we introduce TTT+*CT-KV*, which first performs TTT to update model weights ϕ, then applies *CT-KV* to refine the model's demonstration context for improved performance.

4 Experiments

4.1 Experiment Setup

Datasets. We evaluate on a diverse set of challenging datasets: NLP-LR by Min et al. (2022a) (subsets of CrossFit (Ye et al., 2021) and UnifiedQA (Khashabi et al., 2020)), MMLU (Hendrycks et al., 2021), and the Abstraction and Reasoning Corpus (ARC) (Chollet, 2019).

Models. Following Min et al. (2022a) and Akyürek et al. (2024), we use GPT-2 for NLP-LR and Llama3-8B for BBH. We select Llama3.2-3B for MMLU and Llama3.2-1B for ARC to handle ARC's long sequences. Since the pretrained Llama3.2-1B model cannot solve any ARC evaluation tasks, we follow Akyürek et al. (2024) and Franzen et al. (2024)

Method	NLP-LR		MMLU		BBH		ARC	
	Acc. (%)	T (s)	Acc. (%)	T (s)	Acc. (%)	T (s)	Acc. (%)	T (s)
Baselines								
Zero-Shot	34.9 ± 0.62	0	35.8 ± 0.71	0	40.9 ± 0.43	0	1.0	0
ICL	35.6 ± 0.65	0	41.2 ± 0.57	0	50.4 ± 0.78	0	13.3	0
Prompt Tuning (m = 32)	41.4 ± 1.02	147	39.2 ± 1.04	15	50.8 ± 1.59	7	12.0	13
Prompt Tuning (m = # demo)	38.8 ± 1.23	231	37.3 ± 1.23	29	47.5 ± 1.84	16	14.5	49
Prefix Tuning (m = 32)	42.0 ± 0.85	123	39.9 ± 0.94	5	52.7 ± 1.12	7	9.3	14
Prefix Tuning (m = # demo)	41.1 ± 0.89	144	38.8 ± 0.81	8	52.8 ± 1.15	9	20.5	24
TTT	44.1 ± 0.65	342	43.6 ± 0.55	30	57.8 ± 1.13	14	<u>23.8</u>	56
Our Methods								
CT-Prompt	43.2 ± 0.61	228	43.6 ± 0.67	33	56.3 ± 0.98	14	22.5	52
CT-KV	<u>44.2 ± 0.55</u>	145	<u>43.7 ± 0.54</u>	9	<u>57.9 ± 0.78</u>	7	<u>23.8</u>	26
TTT + *CT-KV*	**47.6** ± 0.53	372	**44.1** ± 0.38	34	**58.2** ± 0.73	17	**25.8**	63

Table 1: Few-shot learning performance on NLP-LR, MMLU, BBH, and ARC benchmarks. Each cell contains the accuracy (%) and training time per task (seconds), delimited by /. We show the means and standard deviations of accuracies over 5 seeds with different sets of demonstration pairs per task (except for ARC because it has fixed demonstration pairs). The best accuracy is **bolded** and second best is <u>underlined</u> for each benchmark.

by fine-tuning it on the ARC training split.

Implementation. We run all experiments in Table 1 over 5 sets of randomly selected demonstration pairs, except ARC because it has a fixed set of demonstration pairs for every task. We initialize trainable prompts and prefixes with sampled token embeddings from the model for their best performance. For *Context Tuning*, we apply Leave-One-Out Masking for all datasets but ARC. Details on this decision, datasets, models, and hyperparameters are in the Appendix.

4.2 Comparing Context Tuning to Baselines

Table 1 reports the performance and training time per task of our baselines and methods on our benchmarks. To fairly compare Prompt and Prefix Tuning with *Context Tuning*, "Prompt Tuning (m = # demo)" and "Prefix Tuning (m = # demo)" match the number of trainable parameters of *CT-Prompt* and *CT-KV* respectively by setting the number of prompt/prefix tokens m to the length of demonstration pairs.

Context Tuning outperforms Prompt and Prefix Tuning. *CT-Prompt* outperforms Prompt Tuning (m = 32), and *CT-KV* outperforms Prefix Tuning (m = 32), both by a wide margin across all benchmarks. Despite increasing m to match the number of trainable parameters of *Context Tuning*, Prompt and Prefix Tuning still underperform. This highlights the effectiveness of leveraging the model's ICL capabilities by initializing the prompt or prefix with demonstration tokens.

CT-KV is more efficient than CT-Prompt. *CT-KV* has lower training time than *CT-Prompt*. This observation aligns with the time complexity discussion in the Appendix: *CT-Prompt* incurs quadratic time to k, while *CT-KV* scales linearly. *CT-KV* also outperforms *CT-Prompt* in accuracy by conditioning each transformer layer's activations with layer-specific key and value vectors, rather than relying solely on input-level soft prompts.

CT-KV offers an efficient alternative to TTT, and the two are complementary. *CT-KV* achieves competitive performance to TTT but uses at most half the training time. Furthermore, *CT-KV* can be applied for a few training iterations after TTT training of model weights to attain higher performance by adding minimal training time, suggesting that context and model adaptation methods under *ICO* can be combined for strong few-shot learning performance.

CT-KV outperforms MetaICL on NLP-LR. *CT-KV* achieves 44.2% accuracy on NLP-LR, surpassing the reported 43.3% accuracy of MetaICL (Min et al., 2022a). This demonstrates that inference-time, single-task optimization with *CT-KV* can rival the performance of approaches that fine-tune model weights across many tasks.

5 Conclusion

We introduced Context Tuning, a lightweight approach to language model few-shot adaptation that optimizes a prompt or prefix initialized from demonstration tokens, combining the strengths of ICL and prompt-based adaptation. We propose two variants: *CT-Prompt* and *CT-KV*, tuning prompts and layer-wise prefix initialized from demonstration tokens, respectively. Experiments on NLP-LR, MMLU, BBH and ARC benchmarks show both variants outperform ICL and Prompt/Prefix Tuning, with *CT-KV* offering the best performance-efficiency trade-off. Framed within the *In-Context Optimization* framework, our results demonstrate that optimizing the context itself is a powerful, scalable alternative to model-weight updates. Moreover, applying *CT-KV* after TTT yields further gains, suggesting that context and model adaptation can be effectively combined.

Acknowledgement

We thank members of the NYU Agentic Learning AI Lab for their helpful discussions. The work is supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under grant RS-2024-00469482, funded by the Ministry of Science and ICT (MSIT) of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research. Jack Lu is supported by the NSERC PGS-D Scholarship. The compute is supported by the NYU High Performance Computing resources, services, and staff expertise.

Societal Impacts

Context Tuning enables language models to make more accurate predictions on novel tasks using only a few task-specific examples. This adaptability can help researchers and practitioners rapidly tailor their models to new domains. However, there is a potential risk if the method is used to adapt models to harmful content. For example, using *Context Tuning* to generate medical advice from unreliable or biased examples could result in misleading or dangerous outputs. We recommend taking extra care to verify, correct, and filter demonstration examples when applying *Context Tuning*.

References

Akyürek, E., Damani, M., Zweiger, A., Qiu, L., Guo, H., Pari, J., Kim, Y., and Andreas, J. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

Chen, Y., Zhong, R., Zha, S., Karypis, G., and He, H. Meta-learning via language model in-context tuning. In *ACL*, 2022.

Chollet, F. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Dalal, K., Koceja, D., Hussein, G., Xu, J., Zhao, Y., Song, Y., Han, S., Cheung, K. C., Kautz, J., Guestrin, C., Hashimoto, T., Koyejo, S., Choi, Y., Sun, Y., and Wang, X. One-minute video generation with test-time training. In *CVPR*, 2025.

Dhariwal, P. and Nichol, A. Q. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

Franzen, D., Disselhoff, J., and Hartmann, D. The llm architect: Solving the arc challenge is a matter of perspective. *arXiv preprint arXiv:2505.07859*, 2024.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., and et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hardt, M. and Sun, Y. Test-time training on nearest neighbors for large language models. In *ICLR*, 2024.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *ICLR*, 2021.

Ho, J. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Jang, J., Jang, S., Kweon, W., Jeon, M., and Yu, H. Rectifying demonstration shortcut in in-context learning. In *ACL*, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., and Hajishirzi, H. UNIFIEDQA: Crossing format boundaries with a single QA system. In *EMNLP (Findings)*, 2020.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

Li, X. and Qiu, X. Finding supporting examples for in-context learning. In *EMNLP (Findings)*, 2023.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. What makes good in-context examples for gpt-3? In *ACL*, 2021.

Lu, J., Teehan, R., and Ren, M. Procreate, don't reproduce! propulsive energy diffusion for creative generation. In *ECCV*, 2024.

Min, S., Lewis, M., Zettlemoyer, L., and Hajishirzi, H. MetaICL: Learning to learn in context. In *NAACL*, 2022a.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022b.

Nichol, A. Q., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI*, 2019.

Shin, T., Razeghi, Y., IV, R. L. L., Wallace, E., and Singh, S. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, 2020.

Srivastava, A., Rastogi, A., Rao, A., Md-Shoeb, A.-A., Abid, A., Fisch, A., Brown, A., Santoro, A., Gupta, A., and et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. In *TMLR*, 2023.

Sun, Y., Wang, X., Zhuang, L., Miller, J., Hardt, M., and Efros, A. A. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

Suzgun, M., Scales, N., Scharli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them. In *ACL*, 2022.

Wallace, B., Gokul, A., Ermon, S., and Naik, N. End-to-end diffusion latent optimization improves classifier guidance. In *ICCV*, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *ICML*, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Ye, Q., Lin, B. Y., and Ren, X. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *EMNLP*, 2021.