Efficient Few-Shot Learning Without Prompts

Lewis Tunstall[1], **Nils Reimers**[2], **Unso Eun Seo Jo**[1], **Luke Bates**[3],
Daniel Korat[4], **Moshe Wasserblat**[4], **Oren Pereg**[4]
[1]Hugging Face [2]cohere.ai
[3]Ubiquitous Knowledge Processing Lab (UKP Lab)
Department of Computer Science and Hessian Center for AI (hessian.AI)
Technical University of Darmstadt
[4]Emergent AI Lab, Intel Labs
[1]firstname@huggingface.com [2]info@nils-reimers.de
[3]bates@ukp.informatik.tu-darmstadt.de
[4]firstname.lastname@intel.com

Abstract

Recent few-shot learning methods, such as parameter-efficient fine-tuning (PEFT) and pattern exploiting training (PET), have achieved impressive results in label-scarce settings. However, they are difficult to employ since they are highly sensitive to handcrafted prompts, and typically require billion-parameter language models to achieve high accuracy. To address these shortcomings, we propose SETFIT (**Se**ntence **T**ransformer **Fi**ne-**t**uning), an efficient and prompt-free framework for few-shot fine-tuning of Sentence Transformers (ST). SETFIT works by first fine-tuning a pretrained ST on a small number of labeled text pairs, in a contrastive Siamese manner. The resulting model is then used to generate rich text embeddings, which are used to train a classification head. This simple framework requires no prompts or verbalizers, and achieves high accuracy with orders of magnitude less parameters and runtime than existing techniques. Our experiments show that SETFIT[1] achieves results competitive with PEFT and PET techniques, and outperforms them on a variety of classification tasks.

1 Introduction

Few-shot learning methods have emerged as an attractive solution to label-scarce scenarios, where data annotation can be time-consuming and costly. These methods are designed to work with a small number of labeled training examples, and typically involve adapting pretrained language models (PLMs) for specific downstream tasks.

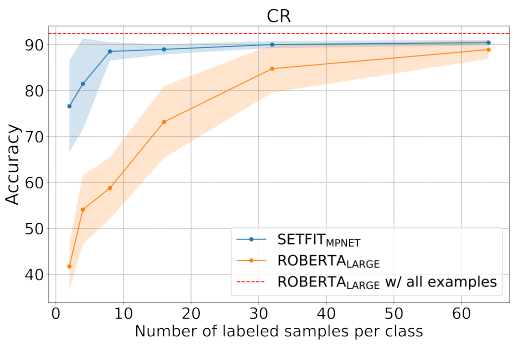

Figure 1: SETFIT is more sample efficient and less variable compared to standard fine-tuning.

There are several approaches to few-shot learning with PLMs. These include in-context learning (ICL), parameter-efficient fine-tuning (PEFT), and prompt-based tuning. Unfortunately, these approaches can be impractical for many researchers and practitioners. For instance, these approaches typically rely on the use of large-scale PLMs like T0 (Sanh et al., 2021) or GPT-3 (Brown et al., 2020a) to achieve high performance. As a result, training and deploying these few-shot methods

[1]https://github.com/huggingface/setfit

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

typically requires specialized infrastructure with limited accessibility. Moreover, existing few-shot methods often require, as part of their training, the input of manually generated prompts. This yields varying outcomes depending on the level of manual prompt-engineering.

In this paper we propose SETFIT, an approach based on Sentence Transformers (Reimers and Gurevych, 2019) that dispenses with prompts altogether and does not require large-scale PLMs to achieve high accuracy. For example, with only 8 labeled examples in the Customer Reviews (CR) sentiment dataset (Hu and Liu, 2004), SETFIT is competitive with fine-tuning on the full training set, despite the fine-tuned model being three times larger (see Figure 1). We demonstrate SETFIT's efficacy in few-shot text classification over a range of datasets. We compare our method to standard PLM fine-tuning, state-of-the-art PET- and PEFT-based methods such as ADAPET (Tam et al., 2021) and T-FEW, as well as recent prompt-free techniques such as PERFECT (Karimi Mahabadi et al., 2022a).

We summarize our contributions as follows:

1. We propose SETFIT– a simple and prompt-free method – and provide a comprehensive guide for applying it in practical few-shot settings.
2. We evaluate SETFIT's performance on a variety of few-shot text classification tasks and show that it outperforms the state-of-the-art prompt-free method and ranks alongside much larger prompt-based, few-shot models.
3. We make the code and data used in our work publicly available.[1]

2 Related Work

Few-shot approaches have recently received a great deal of interest in the research community. Specifically, we consider ICL, PEFT, and prompt-based approaches. ICL models directly generate predictions based on input-to-output training examples provided as prompts, without any parameter updates. Perhaps the best known example is GPT-3 (Brown et al., 2020b), which achieves remarkable few-shot performance. However, GPT-3 contains 175 billion parameters and requires massive computational resources, prompt engineering, and can only utilize pretrained knowledge.

PEFT methods, such as adapters (Rebuffi et al., 2017), hold the majority of parameters fixed during training and only update small feed-forward networks that are inserted within the larger model architecture. A recent example is T-FEW (Liu et al., 2022), which outperforms GPT-3 at much lower computational cost. It accomplishes this by adding learned vectors that rescale the network's internal activations. T-FEW is 16 times smaller than GPT-3, but is still too large to be utilized as a practical tool. It also requires a set of handcrafted prompts for each dataset.

Another alternative to ICL is prompt-based fine-tuning. This approach converts the downstream classification task into a masked-language modeling (MLM) objective. The model outputs tokens in a cloze-style format that maps to the corresponding labels via a predefined template. A well known example of this method is Pattern Exploiting Training (PET) (Schick and Schütze, 2021b,a) . Like GPT-3, PET relies on manually-crafted prompts, but since the model can be fine-tuned to specific tasks, PET-based approaches typically outperform GPT-3 in few-shot scenarios, even with far smaller PLM backbones. PET has since been extended in two main directions: ADAPET (Tam et al., 2021), which improves PET with a decoupled label objective and label-conditioned MLM objective, and PERFECT (Karimi Mahabadi et al., 2022b) which uses task-specific adapters (Houlsby et al., 2019; Pfeiffer et al., 2021) and multi-token label-embeddings eliminate task prompts and verbalizers.

3 SetFit

SETFIT is based on Sentence Transformers (ST) which are modifications of pretrained transformer models that use Siamese network structures to derive semantically meaningful sentence embeddings. The goal of these models is to minimize (maximize) the distance between pairs of semantically similar (distant) sentences. SETFIT works by first fine-tuning a pretrained ST on a small number of labeled text pairs, in a contrastive manner. The resulting model is then used to generate rich text embeddings, which are used to train a classification head (see Figure 2).

ST fine-tuning To better handle the limited amount of labeled training data, we adopt a contrastive training approach that is often used for image similarity detection (Koch et al., 2015). Formally,

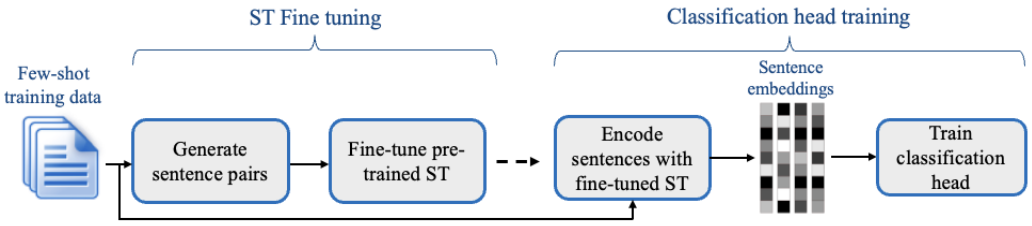

Figure 2: SETFIT 's fine-tuning and training block diagram.

given a small set of K labeled examples $D = \{(x_i, y_i)\}$, where x_i and y_i are sentences and their class labels, respectively. For each class label $c \in C$, we generate a set of R positive triplets $T_p^c = \{(x_i, x_j, 1)\}$, where x_i and x_j are pairs of randomly chosen sentences from the same class c such that $(y_i = y_j = c)$. Similarly, we generate a set of R negative triplets $T_n^c = \{(x_i, x_j, 0)\}$, such that $(y_i = c,\ y_j \neq c)$. Finally, the contrastive fine tuning data set T is produced by concatenating the positive and negative triplets across all class labels; $T = \{(T_p^0, T_n^0), ..., (T_p^{|C|}, T_n^{|C|})\}$, where $|C|$ is the number of class labels, $|T| = 2R|C|$ is the number of pairs in T and R is a hyperparameter. Unless stated otherwise, we used $R = 20$ in all of the evaluations.

Classification head training In this second step, the fine-tuned ST encodes the original labeled training data $\{x_i\}$, yielding one sentence embedding per training sample- $Emb^{x_i} = ST(x_i)$ where $ST()$ is the function representing the fine-tuned ST. The embeddings along with their class labels constitute the training set for the classification head $T^{CH} = \{(Emb^{x_i}, y_i)\}$ where $|T^{CH}| = |D|$. A logistic regression model is used as the text classification head throughout this work.

Inference At inference time, the fine-tuned ST encodes an unseen input sentence (x_i) and produces a sentence embedding. Next, the classification head that was trained in the training step, produces the class prediction of the input sentence based on its sentence embedding. Formally, $x_i^{pred} = CH(ST(x_i))$ where CH represents the classification head prediction function.

4 Experiments

Data We conduct experiments on a variety of text classification datasets and split them into development and test datasets (See Table 5 in appendix A.1). The development datasets are utilized for setting SETFIT's hyperparameters such as the number of training pairs ($|T|$). In addition we evaluate SETFIT on the RAFT benchmark (Alex et al., 2021), a real-world few-shot text-classification benchmark composed of 11 practical tasks, where each task has only 50 training examples.

SETFIT models We evaluate three variations of SETFIT with different underlying ST model sizes (Shown in Table 1). We fine-tune SETFIT's ST model using cosine-similarity loss with a learning rate of $1e^{-3}$, a batch size of 16 and a maximum sequence length of 256 tokens, for 1 epoch.

Baselines We select the following prominent methods as baselines (see implementation details in appendix A.2):[2]

LINEARPROBE We run linear probe experiments using a pretrained ST (paraphrase-mpnet-base-v2) to generate embeddings for a linear classification head. This baseline provides an ablation on the effect of constrastive learning in SETFIT.

Variation	Underlying ST Model	Size
SETFIT$_{\text{ROBERTA}}$	all-roberta-large-v1[y]	355M
SETFIT$_{\text{MPNET}}$	paraphrase-mpnet-base-v2[y]	110M
SETFIT$_{\text{MINILM}}$	paraphrase-MiniLM-L3-v2[y]	15M

Table 1: SETFIT model variations using three different underlying ST models. Number of parameters. [y]`https://huggingface.co/sentence-transformers`

FINETUNE We use ROBERTA$_{\text{LARGE}}$ (Liu et al., 2019) as a standard fine-tuning baseline.

ADAPET (Tam et al., 2021) We use ADAPET with different PLM backbones, and found albert-xxlarge-v2[3] produced the best performance.

[2]Unless otherwise stated, we use the default hyperparameters provided in each baseline's original paper.

[3]`https://huggingface.co/albert-xxlarge-v2`

Method	SST-5	AmazonCF	CR	Emotion	EnronSpam	AGNews	Average
			$\|N\| = 8$				
LINEARPROBE	$36.7_{2.0}$	$22.1_{7.4}$	$84.2_{2.3}$	$37.8_{3.0}$	$90.4_{1.8}$	$74.3_{2.8}$	$57.6_{3.2}$
FINETUNE	$33.5_{2.1}$	$9.2_{4.9}$	$58.8_{6.3}$	$28.7_{6.8}$	$85.0_{6.0}$	$81.7_{3.8}$	$43.0_{5.2}$
PERFECT	$34.9_{3.1}$	$18.1_{5.3}$	$81.5_{8.6}$	$29.8_{5.7}$	$79.3_{7.4}$	$80.8_{5.0}$	$48.7_{6.0}$
ADAPET	$50.0_{1.9}$	$19.4_{7.3}$	$91.0_{1.3}$	$46.2_{3.7}$	$85.1_{3.7}$	$\mathbf{85.1}_{2.7}$	$58.3_{3.6}$
T-FEW	$\mathbf{55.4}_{1}$	27.4_{6}	$\mathbf{91.6}_{1.2}$	$\mathbf{54.7}_{1.3}$	$\mathbf{94.3}_{2.3}$	–	$\mathbf{64.7}_{2.4}$
SETFIT$_{\text{MPNET}}$	$43.6_{3.0}$	$\mathbf{40.3}_{11.8}$	$88.5_{1.9}$	$48.8_{4.5}$	$90.1_{3.4}$	$82.9_{2.8}$	$62.3_{4.9}$
			$\|N\| = 64$				
LINEARPROBE	$42.2_{1.6}$	$40.7_{2.2}$	$88.2_{0.7}$	$50.8_{1.5}$	$95.4_{0.6}$	$84.4_{0.7}$	$67.0_{1.2}$
FINETUNE	$45.9_{6.9}$	$52.8_{12.1}$	$88.9_{1.9}$	$65.0_{17.2}$	$95.9_{0.8}$	$88.4_{0.9}$	$69.7_{7.8}$
PERFECT	$49.1_{0.7}$	$\mathbf{65.1}_{5.2}$	$92.2_{0.5}$	$61.7_{2.7}$	$95.4_{1.1}$	$\mathbf{89.0}_{0.3}$	$72.7_{1.9}$
ADAPET	$54.1_{0.8}$	$54.1_{6.4}$	$92.6_{0.7}$	$72.0_{2.2}$	$96.0_{0.9}$	$88.0_{0.6}$	$73.8_{2.2}$
T-FEW	$\mathbf{57.2}_{0.4}$	$47.1_{7.6}$	$\mathbf{92.9}_{1.5}$	$72.5_{0.9}$	$\mathbf{97.4}_{0.3}$	–	$73.4_{2.1}$
SETFIT$_{\text{MPNET}}$	$51.9_{0.6}$	$61.9_{2.9}$	$90.4_{0.6}$	$\mathbf{76.2}_{1.3}$	$96.1_{0.8}$	$88.0_{0.7}$	$\mathbf{75.3}_{1.3}$

Table 2: SETFIT performance score and stdev compared to the baselines across 6 test datasets for two training set sizes *jNj*. The AGNews dataset is excluded from the average score to enable fair comparison with T-FEW (which included AGNews in its training set).

PERFECT To run PERFECT on our test datasets, we adapted the configurations provided in the PERFECT codebase.[4]

T-FEW We run the 11 billion model variant described in the T-FEW paper (Liu et al., 2022).

Experimental Setup Systematically evaluating few-shot performance can be challenging, because fine-tuning on small datasets may incur instability (Dodge et al., 2020; Zhang et al., 2021). To address this issue, in our experiments we use 10 random training splits for each dataset and sample size. These splits are used as training data across all tested methods. For each method, we report the average measure (depending on the dataset) and the standard deviation across these splits.

5 Results and Discussion

Table 2 shows a comparison between SETFIT$_{\text{MPNET}}$ and the baselines for $N = 8$ and $N = 64$ labeled training samples per class. We find that SETFIT$_{\text{MPNET}}$ significantly outperforms the FINETUNE baseline for $N = 8$ by an average of 19.3 points. However, as the number of training samples increases to $N = 64$, the gap decreases to 5.6 points. Similarly, we find that SETFIT$_{\text{MPNET}}$ outperforms PERFECT by 13.6 and 2.6 points. SETFIT$_{\text{MPNET}}$ also outperforms ADAPET by 4.0 and 1.5 points for $N = 8$ and $N = 64$ respectively. Interestingly, the LINEARPROBE baseline outperforms PERFECT for $N = 8$ and is competitive with ADAPET. For $N = 8$, T-FEW outperforms SETFIT$_{\text{MPNET}}$ by 2.4 points, whereas for $N = 64$ SETFIT$_{\text{MPNET}}$ outperforms T-FEW by 1.9 average accuracy points, despite being prompt-free and 100 times smaller.

RAFT results The test datasets listed in Table 2 were not specifically designed for few-shot benchmarking. In order to better benchmark SETFIT, we used the RAFT benchmark (Alex et al., 2021) which is specifically designed for benchmarking few-shot methods.

Table 3 shows the average accuracy points of SETFIT$_{\text{MPNET}}$ and SETFIT$_{\text{ROBERTA}}$ and four other prominent methods. SETFIT$_{\text{ROBERTA}}$ outperforms GPT3 and PET by 8.6 and 1.7 points respectively. SETFIT$_{\text{ROBERTA}}$ falls short of T-FEW by 4.5 points. however, SETFIT$_{\text{ROBERTA}}$ is more than 30 times smaller than T-FEW, does not require manual prompt crafting and is much more efficient in training and inference (see Table 4).

Rank	Method	Score	Size
1	YIWISE^{y}	76.8	-
2	T-FEW	75.8	11B
4	Human baseline	73.5	-
6	SETFIT$_{\text{ROBERTA}}$	71.3	355M
9	PET	69.6	235M
11	SETFIT$_{\text{MPNET}}$	66.9	110M
12	GPT-3	62.7	175B

Table 3: SETFIT compared to prominent methods on the RAFT leaderboard (Sept. 5, 2022). Model size (#parameters). ^{y}We could not find information regarding YIWISE.

[4] https://github.com/facebookresearch/perfect

Limitations of this work Although SETFIT achieves strong performance on text classification, it is unclear whether the method can be extended to token classification or two-sentence classification tasks such as natural language inference. Another limitation of our method compared to T-FEW is that it requires fine-tuning when applied to new tasks, whereas T-FEW does not require task-specific tuning.

6 Computational Costs

In order to compare the computational costs of SETFIT versus other methods, we follow the approach adopted by Liu et al. (2022) and use FLOPs-per-token estimates to compare SETFIT to T-FEW. See appendix A.4 for detailed description of FLOPs-per-token estimations. Table 4 shows the computational cost comparison between SETFIT and T-FEW. The $\text{SETFIT}_{\text{MPNET}}$ model is an order of magnitude faster at inference and training than T-FEW, despite having comparable performance on the test datasets of Table 2. $\text{SETFIT}_{\text{MINILM}}$ is two orders of magnitude faster than T-FEW with average score reduction of 4.4 points. Moreover, the storage cost of the SETFIT models (70MB and 420MB respectively) is 636 to 106 times smaller than the T0-11B checkpoint used by T-FEW (44.5GB), making these models much better suited for real-world deployment.

Method	Inf. FLOPs	Train FLOPs	Speed-up	Score
T-FEW	5.9e11	1.4e16	1x	$64.7_{1.9}$
$\text{SETFIT}_{\text{MPNET}}$	8.3e9	2.0e14	71x	$62.3_{4.9}$
$\text{SETFIT}_{\text{MINILM}}$	1.3e9	3.2e13	450x	$60.3_{1.6}$

Table 4: Relative computational cost and average scores of SETFIT and T-FEW using $jNj = 8$ on the test datasets listed in Table 2. Trained in a distillation setup (see A.6).

These estimates are borne out by comparing the time needed to train each method to convergence on $N = 8$ examples. For our datasets, $\text{SETFIT}_{\text{MPNET}}$ takes approximately 30 seconds to train on a p3.2xlarge AWS instance (16GB GPU memory), at a cost of $0.025 per split. On the other hand, T-FEW requires at least 80GB GPU memory, and training on a p4d.24xlarge AWS instance takes approximately 736 seconds, at a cost of $1.05 per split.

7 Conclusion

This paper introduces SETFIT, a new few-shot text classification approach. We show that SETFIT has several advantages over comparable approaches such as T-FEW, ADAPET and PERFECT. In particular, SETFIT is much faster at inference and training; SETFIT requires much smaller base models to be performant, and does not require manual prompting.

Acknowledgements

The authors would like to thank Hugging Face Inc and Intel Inc. for providing computing resources and the German Federal Ministry of Education and Research and the Hessian Ministry of Science and the Arts (HMWK) within the projects "The Third Wave of Artificial Intelligence - 3AI", hessian.AI, and within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

References

Neel Alex, Eli Lifland, Lewis Tunstall, Abhishek Thakur, Pegah Maham, C. Jess Riedel, Emmie Hine, Carolyn Ashurst, Paul Sedille, Alexis Carlier, Michael Noetel, and Andreas Stuhlmüller. 2021. RAFT: A real-world few-shot text classification benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662. Curran Associates, Inc.

Stephen H. 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 Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru

Tang, Dragomir Radev, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. Cite arxiv:1503.02531 Comment: NIPS 2014 Deep Learning Workshop.

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. *CoRR*, abs/1902.00751.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 168–177, New York, NY, USA. Association for Computing Machinery.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Lambert Mathias, Marzieh Saeidi, Veselin Stoyanov, and Majid Yazdani. 2022a. Prompt-free and efficient few-shot learning with language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3638–3652, Dublin, Ireland. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Lambert Mathias, Marzieh Saeidi, Veselin Stoyanov, and Majid Yazdani. 2022b. Prompt-free and efficient few-shot learning with language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3638–3652, Dublin, Ireland. Association for Computational Linguistics.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020. The multilingual amazon reviews corpus.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

James O'Neill, Polina Rozenshtein, Ryuichi Kiryo, Motoko Kubota, and Danushka Bollegala. 2021. I wish I would have loved this one, but I didn't - A multilingual dataset for counterfactual detection in product reviews. *CoRR*, abs/2104.06893.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

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. 2021. Multitask prompted training enables zero-shot task generalization. *CoRR*, abs/2110.08207.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

I. Androutsopoulos V. Metsis and G. Paliouras. 2006. Spam filtering with naive bayes - which naive bayes? In *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS 2006)*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015a. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *NIPS*.

A Appendix

This appendix describes additional evaluations of the SETFIT method.

A.1 Datasets

Table 5 shows the development and test datasets that are used for setting SETFIT's hyperparameters. Following is a description of the datasets used:

Dataset Name	Type of Task	Cls.	Label Dist.**	Metric	Split
SST5	Sentiment	5	Approx. equal	Accuracy	Test
Amazon Counterfactual	Counterfactual	2	10% counterfactual	MCC	Test
CR	Sentiment	2	Equal	Accuracy	Test
Emotion	Emotion	6	Equal	Accuracy	Test
Enron Spam	Unwanted Language	2	Equal	Accuracy	Test
AG News	Topic	4	Equal	Accuracy	Test
SST2	Sentiment	2	Equal	Accuracy	Dev
IMDB	Sentiment	2	Equal	Accuracy	Dev
BBC News	Topic	5	Equal	Accuracy	Dev
Student Question Categories	Topic	4	Approx.Equal	Accuracy	Dev
TREC-QC	Topic	50	N/A	Accuracy	Dev
Toxic Conversations	Unwanted Language	2	8% Toxic	Avg. Precision	Dev
Amazon Polarity	Sentiment	2	Equal	Accuracy	Dev

Table 5: English datasets used for development and test experiments. No. of classes per dataset. Distribution of the examples across classes.

SST2 The Stanford Sentiment Treebank 2 is a collection of single sentence movie reviews with positive-negative sentiment class labels. Socher et al. (2013).

IMDB The Internet Movie Database dataset is a collection of single sentence movie reviews with positive-negative sentiment class labels. Maas et al. (2011).

BBC News The BBC News dataset is a collection of articles from the news outlet BBC with one of 5 topic classifications: Politics, Sports, Entertainment, Tech, and Business. Greene and Cunningham (2006).

Enron Spam The Enron spam email dataset consists of emails from the internal Enron correspondence channel where emails are classified as spam or not spam. V. Metsis and Paliouras (2006).

Student Question Categories This dataset[5] is a set of questions from university entrance exams in India that are classified into 4 subjects: Math, Biology, Chemistry, Physics.

TREC-QC The Text Retrieval Conference Question Answering dataset.

[5] `www.kaggle.com/datasets/mrutyunjaybiswal/iitjee-neet-aims-students-questions-data`

Toxic Conversations The Toxic Conversations dataset[6] is set of comments from Civil Comments, a platform for reader comments for independent news outlets. Human raters have given them toxicity attributes.

Amazon Polarity The Amazon Polarity dataset[7] consists of customer reviews from *Amazon* taken over 18 years with binary sentiment labels. Examples are either positive ("Great Read") or negative ("The Worst!") labelled. Zhang et al. (2015a).

Following is a description of the test datasets:

Stanford Sentiment Treebank-5 (SST5) The SST-5 dataset is the fine-grained version of the Stanford Sentiment Treebank, where each example is given one of five labels: very positive, positive, neutral, negative, very negative.

Amazon Counterfactual The Amazon Counterfactual dataset is set of *Amazon* customer reviews with professionally labeled binary labels of counterfactual detection. Counterfactual statements are statements that denote something that did not happen or cannot (e.g. "They are much bigger than I thought they would be."). We used the English subset for our experiments. O'Neill et al. (2021).

Emotion The Emotion dataset[8] consists of tweets from *Twitter* that display clear emotions (e.g. "i am now nearly finished [with] the week detox and i feel amazing"). Labels are one of six categories: anger, fear, joy, love, sadness, and surprise. Saravia et al. (2018).

AG News AG News is a dataset of news titles from AG news with one of 4 classifications (World, Entertainment, Sports, and Business). Zhang et al. (2015b).

A.2 Baselines Implementation Details

Following are the implementation details that were used for fine-tuning our baselines:

FINETUNE We perform a hyperparameter search on the number of epochs in the range [25,75] and pick the best performing model on a validation split. We use a learning rate of $2e^{-5}$ and batch size of 4 in all our experiments.

ADAPET By default, ADAPET assumes access to a training, development, and test dataset. It trains for $1,000$ batches, runs predictions on the development data every 250 batches and checkpoints, keeping the model state which performed best on the development dataset. In our case, where we assume few-shot training and no development data, we ran ADAPET for $1,000$ batches and disabled the checkpointing, using the model state that resulted after training for $1,000$ batches. For the English data in Table 2, we used the pattern "[TEXT1] this is [LBL]", where "[TEXT1]" and "[LBL]" are placeholders for a given piece of text and the corresponding label, respectively. We chose this pattern as a variation of the pattern the ADAPET authors provide in their code[9]. In our development phase, this pattern empirically yielded stronger performance compared to the handful of patterns we tried. We constructed the verbalizer from the "label" and "label text" columns that are available in all of our datasets.

We used the default hyperparameters and examined its performance with different PLM backbones, reporting the PLM which resulted in the best performance, albert-xxlarge-v2[10].

PERFECT To run PERFECT on our test datasets, we adapted the configurations provided in the PERFECT codebase.

T-FEW Running tests on T-FEW as opposed to SETFIT posed several hurdles. First, because T-FEW's performance varies significantly depending on the input prompts, we run each experiment

[6] https://www.kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification/data

[7] https://huggingface.co/datasets/amazon_polarity

[8] https://huggingface.co/datasets/emotion

[9] https://github.com/rrmenon10/ADAPET/blob/master/config/sst-2.json

[10] https://huggingface.co/albert-xxlarge-v2

using 5 random seeds, and report the median result, as in the original paper. Second, T-FEW relies on dataset-specific prompts, made available on P3 (Public Pool of Prompts) (Bach et al., 2022). The **Emotion** dataset is the only one that had existing prompts in P3. For the rest of the datasets, we adapt standardized P3 prompts of similar tasks or implement prompts ourselves (appendix A.3 lists the prompts we created).

A.3 Prompts used in T-FEW

Following are the prompts we created for

- Prompts for **Enron Spam**, a spam e-mail detection dataset, were adapted from `sms_spam` dataset prompts.
- **CR** prompts were adapted from `amazon_polarity`.
- **SST5** prompts were adapted from `yelp_review_full`.

The **Amazon Counterfactual** dataset does not have any relevant prompts on P3. Hence, we manually generated prompts ourselves, based on standard practices for prompt creation published in P3. We also added two new prompts for **SST5**, to make it compatible with the label names of **SST5**. Following is a list of prompts we created for each dataset:

Amazon Counterfactual Prompts

Input template:

```
{{text}} Is the statement factual?
```

Target template:

```
{{ answer_choices [label] }}
```

Answer choices template:

```
Yes ||| No
```

Input template:

```
{{text}} Does the statement describe a fact?
```

Target template:

```
{{ answer_choices[label] }}
```

Answer choices template:

```
Yes ||| No
```

Input template:

```
{{text}} Is the statement non-counterfactual or counterfactual?
```

Target template:

```
{{ answer_choices[label] }}
```

Answer choices template:

```
non-counterfactual ||| counterfactual
```

Input template:

```
{{text}} Is the statement counterfactual?
```

Target template:

```
{{ answer_choices[label] }}
```

Answer choices template:

```
No ||| Yes
```

Input template:

```
{{text}} Does the sentence express an event that did not happen?
```

Target template:

```
{{ answer_choices[label] }}
```

Answer choices template:

```
No ||| Yes
```

Input template:

```
{{text}} Does this describe an actual event?
```

Target template:

```
{{ answer_choices[label] }}
```

Answer choices template:

```
Yes ||| No
```

Input template:

```
{{text}} Does the sentence contain events that did not or cannot take place?
```

Target template:

```
{{ answer_choices[label] }}
```

Answer choices template:

```
Yes ||| No
```

Input template:

```
Is the label for the following sentence non-counterfactual or counterfactual?
{{text}}
```

Target template:

```
{{ answer_choices[label] }}
```

Answer choices template:

```
non-counterfactual ||| counterfactual
```

New prompts for SST5

Input template:

```
How do you feel about the following sentence? {{ text }}
```

Target template:

```
{{ answer_choices[label] }}
```

Answer choices template:

```
very negative ||| negative ||| neutral ||| positive ||| very positive
```

Input and target templates:

```
{{ text }} This movie is a very ||| {{ answer_choices[label] }} one
```

Answer choices template:

```
terrible ||| bad ||| okay ||| good ||| great
```

A.4 Computational costs

Comparing the relative computational costs of SETFIT versus methods such as T-FEW is complicated by the fact that each method typically has different hardware and memory requirements. To simplify the comparison, we follow the approach adopted by (Liu et al., 2022) and use FLOPs-per-token estimates to compare SETFIT to T-FEW. These estimates can be obtained from Kaplan et al. (2020), who show that encoder-only models with N parameters have approximately $2N$ FLOPs-per-token for inference and $6N$ FLOPs-per-token for training. For a given input sequence length ℓ_{seq}, the final estimate for inference is $2N \quad \ell_{\text{seq}}$, while training is $6N \quad \ell_{\text{seq}} \quad n_{\text{steps}} \quad n_{\text{batch}}$, where n_{steps} is the number of training steps and n_{batch} is the batch size. For encoder-decoder models like T-FEW, these estimates are halved.

For the inference and training estimates shown in Table 4 in Section 6, we use 38 (54) tokens as the input sequence length for $\text{SETFIT}_{\text{MPNET}}$ (T-FEW), which is the median taken across all the test datasets in Table 2 in Section 5.

A.5 Multilingual Experiments

To determine SETFIT's performance in a multilingual, few-shot text classification scenario, we conducted development and test experiments on multilingual datasets and compared SETFIT to standard transformer fine-tuning and ADAPET. To the best of our knowledge, this is the first work to examine ADAPET on non-English data.

Experimental Setup For the multilingual experiments, we use the Multilingual Amazon Reviews Corpus (MARC) Keung et al. (2020). This dataset consists of Amazon reviews in six languages (English, Japanese, German, French, Spanish, and Chinese), where each review is labeled according to a 5-star rating scale. We chose this corpus for its typological diversity in order to examine the generalizability of SETFIT and other methods across a variety of languages.

For the SETFIT underlying model, we use paraphrase-multilingual-mpnet-base-v2,[11] which is a multilingual version of paraphrase-mpnet-base-v2 that is trained on parallel data in over 50 languages.

For the FINETUNE and ADAPET baselines, we use $\text{XLM-ROBERTA}_{\text{BASE}}$ (Conneau et al., 2019),[12] which has a similar size to the SETFIT model. We compare the performance of each method using the same settings as (Conneau et al., 2019):

- **each:** Train and evaluate on monolingual data to measure per-language performance.
- **en:** Train on the English training data and then evaluate on each language's test set.
- **all:** Train on all the training data and evaluate on each language's test set.

Method For SETFIT standard fine-tuning, and ADAPET, we adopt the same methodology and hyperparameters used for the monolingual English experiments in 4. We evaluate each method in the few-shot regime ($N = 8$ samples per class) and compare against performance of fine-tuning on the full training set of 20,000 examples.

[11] huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2

[12] huggingface.co/xlm-roberta-base

Method	Train	En	De	Ja	Zh	Fr	Es	Average
				$jNj = 8$				
FINETUNE	each	$122.9_{14.0}$	$119.9_{13.6}$	$120.5_{8.0}$	$128.6_{10.7}$	$123.2_{13.0}$	$116.3_{8.3}$	$121.9_{11.3}$
	en	$115.9_{11.3}$	$115.2_{12.0}$	$121.6_{12.3}$	$123.0_{8.8}$	$117.3_{13.0}$	$113.1_{12.4}$	$117.7_{11.6}$
	all	$117.8_{4.9}$	$116.3_{9.7}$	$121.5_{12.4}$	$120.5_{6.7}$	$117.3_{9.9}$	$110.1_{9.5}$	$117.2_{8.8}$
ADAPET	each	$129.9_{13.6}$	$136.4_{10.6}$	$130.4_{13.4}$	$135.0_{10.9}$	$141.8_{10.1}$	$136.0_{10.4}$	$134.9_{11.5}$
	en	$138.9_{17.8}$	$151.5_{17.8}$	$160.8_{16.7}$	$158.8_{16.3}$	$152.0_{15.7}$	$149.8_{17.1}$	$152.0_{16.9}$
	all	$150.8_{12.0}$	$136.2_{7.0}$	$150.8_{10.0}$	$152.8_{10.2}$	$140.0_{14.0}$	$145.1_{4.5}$	$146.0_{11.3}$
SETFIT	each	$82.9_{4.3}$	$80.0_{2.4}$	$95.5_{2.8}$	$95.3_{2.8}$	$85.3_{6.0}$	$80.8_{5.4}$	$86.6_{4.9}$
	en	$82.6_{4.8}$	$83.4_{5.9}$	$93.2_{6.6}$	$93.9_{3.6}$	$82.2_{4.8}$	$83.4_{5.9}$	$86.4_{5.2}$
	all	$83.0_{5.3}$	$84.0_{7.6}$	$97.1_{9.2}$	$97.4_{6.5}$	$83.5_{6.5}$	$84.9_{6.1}$	$88.3_{6.9}$
				jNj =*Full*				
FINETUNE	each	46.2	43.7	46.8	56.6	47.8	45.3	47.7
	en	46.1	46.6	61.0	69.4	55.6	52.9	55.3
	all	46.6	49.4	61.0	69.4	55.6	55.0	56.2

Table 6: Average performance (MAE 100) on the Multilingual Amazon Reviews Corpus for two training set sizes jNj. No. of training samples per class. Entire available training data used (20,000 samples).

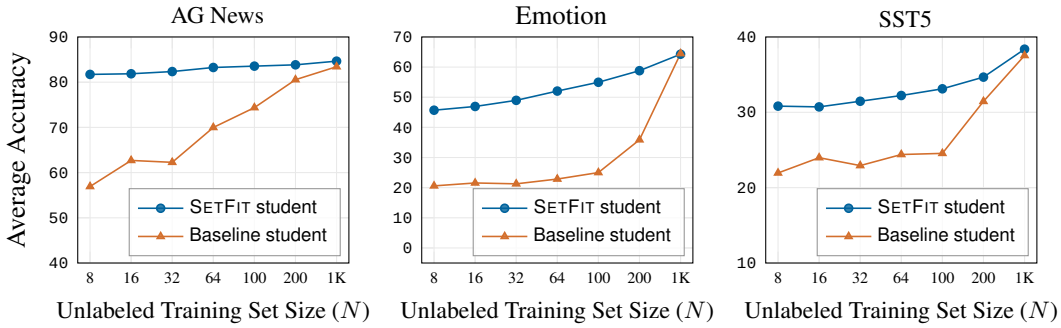

Figure 3: Average accuracy as a function of the unlabeled training set size N of the SETFIT student and the baseline student on AG News, Emotion and SST5 datasets.

Results Table 6 shows the results of SETFIT standard fine-tuning, and ADAPET on each language in MARC, where a higher MAE indicates weaker performance. In the few-shot regime of $N = 8$ samples per class, we find that SETFIT significantly outperforms FINETUNE and ADAPET in all settings (each, en, all), with the best average performance obtained when training on English data only.

A.6 Few-shot distillation

We have shown that SETFIT achieves state-of-the-art results in few-shot setups using underlying base models such as paraphrase-mpnet-base-v2 and ROBERTA$_{\text{LARGE}}$, containing 110M parameters and 355M parameters respectively; but in real-world deployments, where cost and sustainability are prioritized, the use of even more efficient models is desirable. Previous works have shown model distillation to be effective in reducing computational load while preserving much of the original model's performance (Ba and Caruana, 2014; Hinton et al., 2015). In this section we evaluate the performance of SETFIT as a student model compared to a standard transformer student model in few-shot distillation setups when the amount of unlabeled training data is limited.

Experimental Setup For the distillation tests we use the datasets AGNews, Emotion and SST-5 shown in Table 5. For the SETFIT teacher we chose SETFIT$_{\text{MPNET}}$, which contains 110M parameters, whereas for the SETFIT student we chose SETFIT$_{\text{MINILM}}$, which is a much smaller model (15M parameters). For fair comparison, we use as the baseline student MiniLM-L3-H384-uncased[13], a

[13] `huggingface.co/nreimers/MiniLM-L3-H384-uncased`

standard transformer encoder of the same size as our SETFIT student model. For each of the three datasets we train the SETFIT teacher model using only 16 labeled samples per class, and the student models are trained using the same 16 labeled samples per class together with various amounts of additional unlabeled data. We follow the same data-split policy and SETFIT training parameters' settings described in Section 4.

Method The SETFIT student is trained using sentence pairs and the level of similarity between each pair as input. The similarity is generated by using the underlying ST of the teacher to produce sentence embeddings for each pair and to calculate the cosine-similarity between them. The underlying ST of the SETFIT student is trained to mimic the ST of the teacher output by minimizing the error between the SETFIT teacher-produced cosine-similarity and its output. The classification head of the student is then trained using the embeddings produced by the student's ST and the logits produced by the SETFIT teacher. The baseline student is trained to mimic the teacher output by minimizing the error between the logits produced by the SETFIT teacher and its its output.

Results Figure 3 shows a comparison between the SETFIT student model and the baseline student model for various amounts of unlabeled training data (N). The SETFIT student significantly outperforms the baseline student when only small amounts of unlabeled data are available. For example, for $N = 8$, the SETFIT student outperforms the baseline student by **24.8**, **25.1**, and **8.9** average accuracy on the AG News, Emotion and SST5 datasets, respectively. As N increases, the performance gains decrease and are on par for $N = 1K$.