FastFit: Fast and Effective Few-Shot Text Classification with a Multitude of Classes

Asaf Yehudai ♢ ♣ **, Elron Bandel** ♢
IBM Israel Research Lab ♢, Hebrew University of Jerusalem ♣
{first.last}@ibm.com

Abstract

We present FastFit, a Python package designed to provide fast and accurate few-shot classification, especially for scenarios with many semantically similar classes. FastFit utilizes a novel approach integrating batch contrastive learning and token-level similarity score. Compared to existing few-shot learning packages, such as SetFit, Transformers, or few-shot prompting of large language models via API calls, FastFit significantly improves multi-class classification performance in speed and accuracy across various English and Multilingual datasets. FastFit demonstrates a 3-20x improvement in training speed, completing training in just a few seconds. The FastFit package is now available on GitHub, presenting a user-friendly solution for NLP practitioners.[1]

1 Introduction

Few-shot classification presents a unique challenge, especially when dealing with a multitude of classes that share similar semantic meanings. Expanding the training data can be both time-consuming and costly. To address this challenge, two primary categories of tools have been developed: few-shot prompting of large language models (LLMs) via API calls, or packages designed for fine-tuning smaller language models using the limited available data. However, we recognize the drawbacks of applying both of these approaches in practice.

Few-shot prompting of LLMs leverages their multitasking abilities to tackle data scarcity. However, in the presence of many classes, LLMs encounter three major challenges: (1) LLMs struggle to incorporate demonstrations of all classes within their context window. (2) Utilization of the long context for the classification task can be challenging (Liu et al., 2023). (3) Due to the model size, and prompt length the inference time is slow.

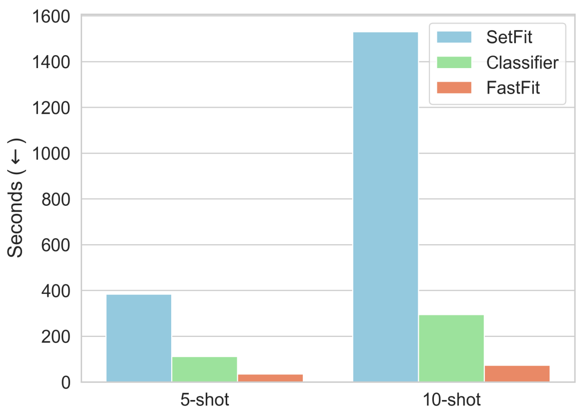

Figure 1: Training times (sec) for FastFit, SetFit, and standard classifier with MPNet model. FastFit training is 3-20x faster.

In contrast, the approach of fine-tuning smaller language models capitalizes on their adaptability to specific tasks, as demonstrated to be effective in recent works. However, these methods can be challenging to deploy as they require architectural adjustments (Yehudai et al., 2023) or, like SetFit, may prove less suitable for classification with many classes (Tunstall et al., 2022).

In this work, we present FastFit, a fast and accurate method, and a pip-installable Python package designed for fine-tuning small language models in few-shot classification tasks involving many classes. Through various experiments, we demonstrate that FastFit training is significantly faster, providing a 3-20x speedup. This enables training within seconds, as illustrated in Fig. 1. FastFit outperforms earlier packages, including SetFit, Transformer, and multi-task models like FLAN, or larger LLMs like LLama-70B, in both English and Multilingual settings.

The core contribution facilitating this speedup and improvement lies in FastFit's use of batch contrastive training, recognized for its efficiency and effectiveness (Khosla et al., 2021). This technique brings same-class texts closer while pushing apart

[1] FastFit GitHub

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations), pages 174–184
June 16-21, 2024

all other texts. FastFit also incorporates token-level text similarity measures that leverage fine-grained information (Zhang et al., 2020; Khattab and Zaharia, 2020). Additionally, we integrate text augmentation techniques to enhance the robustness of the training process (Gao et al., 2021).

The FastFit package is easy to install and use, interfacing with standard training APIs (See §2). We hope that FastFit will help make text classification easier and faster for the benefit of the whole community.

2 The FastFit API

The FastFit Python package is available on PyPI and can be installed with: `$ pip install fast-fit`

To utilize FastFit, import the FastFit trainer, which inherits from the Hugging Face (HF) trainer. This enables FastFit to be customizable, inheriting all parameters from the HF trainer. FastFit supports loading datasets either by directly passing the dataset or providing file paths.

Here is a simple code example of loading and training FastFit. In App. §A, we provide a complete code example.

```
from fastfit import FastFitTrainer

trainer = FastFitTrainer(
   model_name_or_path=
      "roberta-large",
   label_column_name="label_text",
   text_column_name="text",
   dataset=dataset,
)

model = trainer.train()
results = trainer.evaluate()
```

As FastFit utilizes example texts and class names, it expects the data to have `text` and `label` fields or to map the existing fields to them using the `label_column_name` and `text_column_name` parameters of the `FastFitTrainer`. Our trainer also supports training with either CLS or token-level similarity metrics, set by the `sim_rep` parameter. The trainer allows to modify the number of augmentation repetitions with the `num_repeats` parameter. Then after training, we can easily save the model:

```
model.save_pretrained("fast-fit")
```

And later load it for inference, See App. §A.

3 Method

Given a few-shot text classification dataset containing texts and their corresponding classes denoted as $\{x_i, y_i\}_{i=1}^N$, let $C = \{c_j\}_{j=1}^M$ represent all possible classes. Our task is to classify each x_i into a class $y_i \in C$. To achieve this goal we aim to encode both texts and class names into a shared embedding space, where they are represented closely, according to a similarity metric S, when they belong to the same class and are represented further apart when they do not. To accomplish this, we optimize the following batch contrastive loss:

$$\mathcal{L} = \sum_{b \in [B]} \frac{-1}{|P(b)|} \sum_{p \in P(b)} \log \frac{e^{S(x^b, x^p)/\tau}}{\sum_{a \in [B] \setminus b} e^{S(x^b, x^a)/\tau}} \tag{1}$$

Here, $\{x_b\}_{b=1}^B$ represents a batch of B texts, and $P(b)$ refers to the set of texts in the same class as b in the batch, given by $P(b) = \{c \in [B], |, y_c = y_b\}$. The function S is the similarity metric, and τ is a scalar temperature parameter regulating the penalty for negative texts.

For each text in the batch, we augment the batch by including its class name as an additional example. Additionally, we repeat the texts in the batch r times as a data augmentation technique, following Gao et al. (2021) by treating the dropout as a minimal augmentation at the representation level. This method has demonstrated significant success in generating sentence embeddings, and we leverage it here to enhance representation for text classification.

In our data-scarce setting, we employ fine-grained token-level similarity metrics, leveraging textual details. This approach, successful in works like BERT-Score and ColBERT, defines the similarity metric between texts x_i and x_j as the sum of cosine similarities between x_i and the most similar tokens in x_j. Specifically, with tokens denoted as $x_i^1, \dots, x_i^n$ and $x_j^1, \dots, x_j^m$ respectively, the similarity score is computed as follows:

$$S(x_i, x_j) = \sum_{k=1}^{n} \max_{l=1}^{m} E_\theta(x_i^k) \cdot E_\theta(x_j^l) \tag{2}$$

where $E_\theta(x_i^k)$ is a dense representation of token x_i^k produced by a parametric encoder model with parameters θ.

During inference, when provided with a new text, x_u we classify it to the most similar class $y_i \in C$ with respect to a similarity metric S. This method draws inspiration from the way inference is conducted in retrieval systems, eliminating the need for a classification head and aligning the training and inference objectives.

4 Experiments

4.1 Datasets

We experiment with three English few-shot text classification datasets: Hwu64 (Liu et al., 2019a), Banking77 (Casanueva et al., 2020), and Clinc150 (Larson et al., 2019). The datasets have between 64 and 150 classes. Many classes are semantically similar, making the classification tasks much harder. We conduct our experiments in 5/10-shot scenarios where in the k-shot scenario the training set consisted of k examples per class. See App. §B for full data statistics.

4.2 Baselines

We compare FastFit with a few classification methods, including fine-tuning methods, like Standard and SetFit classifiers, and few-shot promoting of LLMs including Flan-XXL (Wei et al., 2022), Flan-ul2 (Tay et al., 2023), llama-2-70b-chat (Touvron et al., 2023), and Mistral-7b (Jiang et al., 2023). For all fine-tuning methods, we use small and large versions, where small is MPNet (110M parameters) (Song et al., 2020), and large is Roberta-large (355M parameters)(Liu et al., 2019b) or equivalent.

Standard classifier. A simple yet strong baseline is a standard fine-tuning of an encoder-only model. Since we assume no validation sets, we use best practices as described in previous works, and train for 40 epochs, with a learning rate of $1e-5$, and batch size of 16 (Lin et al., 2023). We recovered runs that didn't converge.

SetFit. Sentence Transformer Fine-tuning (SetFit) (Tunstall et al., 2022) is a two-stage method for training a Sentence Transformer model (Reimers and Gurevych, 2019), specifically designed for few-shot classification tasks. In the first stage, the encoder undergoes fine-tuning using triplet loss, and in the second stage, the classification head is trained. For the small model we use `paraphrase-mpnet-base-v2`[2], and for the large model, we used `all-Roberta-Large-v1`[3], both trained with sentence transformer objective before. We trained the model with a learning rate of $1e-5$, a batch size of 16, for one epoch, based on the parameters defined in SetFit's paper.

Flan. Flan language models are fine-tuned on a diverse range of NLP tasks and datasets, making them adaptable for various NLP tasks in a few-shot manner. Here, we experimented with Flan-XXL (11B) and Flan-ul2 (20B) models. These models have a 4K tokens context window.

Llama. Llama-2-chat is a set of large language models developed for conversational applications and has strong multi-task few-shot capabilities. Here, we experimented with a Llama model that supports a 4K tokens context window.

Mistral. Mistral is a strong 7B open-source large language model. Here, we used the instruct-tuned version. Mistral supports an 8K tokens context window.

4.3 Experimental Setup

Training Setup. We fine-tune the FastFit model with a learning rate of $1e-5$, a batch size of 32, and a maximum sequence length of 128 tokens, for 40 epochs. We used AdamW optimizer, 16-bit floating-point (FP16) precision, and applied 4 batch repetitions that acts as augmentations.

All LLMs, except Mistral, have a context window of 4K. We were able to fit 1 example into their context for Clinc150 and Banking77, and 3 examples for Hwu64. Mistral, with an 8K context window allows for 2, 3, and 5 examples from Clinc150, Banking77, and Hwu64, respectively.

Evaluation Setup. Few-shot evaluations can be noisy due to variations in the small datasets (Dodge et al., 2020; Zhang et al., 2021). To address this challenge, we perform all our experiments using 5 random training split variations and report the mean results.

4.4 Results

In Table 1, we present the results of FastFit, SetFit, and the standard classifier for three datasets under 5/10-shot settings. FastFit large outperforms SetFit by 2.1% and the standard classifier by 3.4%. FastFit small outperforms SetFit by 3.4% and the standard classifier by 5.1%, achieving comparable results to SetFit large. Notably, FastFit shows

[2] ST-MPNet
[3] ST-Roberta-Large

Method	Size	CLINC150		BANKING77		HWU64		Average
		5	10	5	10	5	10	
FastFit	S	90.2	93.3	80.1	85.4	79.8	84.7	85.6
	L	**92.2***	**94.8***	**83.0***	**87.9***	**82.9***	**86.3***	**87.9***
SetFit	S	86.9	90.5	74.3	81.9	77.8	81.8	82.2
	L	90.7	93.1	79.1	86.4	81.0	84.6	85.8
Classfier	S	86.0	91.4	68.1	80.4	74.4	82.9	80.5
	L	89.2	94.0	75.9	86.1	76.3	85.5	84.5

Table 1: Accuracy results of FastFit and baselines on 5/10-shot text classification. Results show that FastFit outperforms SetFit and standard classfier. Moreover, FastFit small is comparable to SetFit large. Results with * are statistically significant by t-test ($p < 0.05$) compared to the large standard classifier.

Model	C150	B77	H64	Avg.
Flan-ul2	80.3	71.5	**76.2**	76.0
Flan-XXL	**82.1**	**72.1**	74.9	**76.3**
Llama-2-13B-chat	53.0	42.6	53.2	49.6
Llama-2-70B-chat	60.8	45.7	62.8	56.4
Mistral-7B	63.5	46.8	71.7	60.7

Table 2: Accuracy results of a few LLMs models. The Flan models outperform the other LLMs. Llama-70B scores higher than Llama-13B but less than Mistral, which has a larger context window.

greater improvement in the 5-shot case compared to the 10-shot case and for the small model compared to the large one.

Table 2 displays the results of few-shot prompting for several LLMs. The Flan models exhibit higher performance than other LLMs, likely due to the presence of many classification datasets in the Flan dataset, which do not include our test sets. This observation aligns with findings in zero-shot classification (Gretz et al., 2023). Although Llama-70B outperforms Llama-13B, it falls short of Mistral-7B's performance, possibly due to Mistral's larger context length, allowing it to incorporate more examples per class.

The results suggest that in our setting, where numerous classes are present, even the best-performing LLMs we tested (Flan's) underperform compared to large standard classifiers and face challenges compared to FastFit. It's important to note that, due to the model's size and the length of the few-shot prompt, inference time can be slow, with throughput exceeding 1 second per input, in contrast to about 1 millisecond with FastFit.

5 Multilingual Experiments

5.1 Datasets

To evaluate FastFit's multilingual classification abilities we adopt Amazon Multilingual MASSIVE dataset (FitzGerald et al., 2022). From the 51 available languages, we selected six typologically diverse languages: English, Japanese, German, French, Spanish, and Chinese. MASSIVE is a parallel dataset, with 60 classes (See App. §B).

5.2 Baselines

For multilingual training, we utilized paraphrase-multilingual-mpnet-base-v2 as a small model and XLM-Roberta-Large as a large model. Both models underwent pretraining in a large number of languages. Notably, to the best of our knowledge, there is no multilingual sentence transformer model equivalent to Roberta-Large for SetFit training. Monolingual and XLM-Roberta-Large models were tested, but they yielded lower performance than the small model; hence, their results are detailed in Appendix §C. In English experiments, we maintained the use of monolingual models (see §4.2), conducting training and evaluation with the same setup outlined in §4.3.

5.3 Results

In Table 3, we present the results on MASSIVE in 5/10-shot scenarios using FastFit, SetFit, and the standard classifier. FastFit consistently outperforms both SetFit and the standard classifier in both 5-shot and 10-shot settings, across small and large models. In the 5-shot scenario, FastFit large achieve an 8% improvement over SetFit small and a 12.4% improvement over the standard classifier. Meanwhile, FastFit small shows a 2.7% improvement over SetFit small and a 7.1% improvement over the standard classifier. In the 10-shot case,

Method	Size	En	De	Ja	Es	Fr	Zh	Average
				5-shot				
FastFit	S	<u>72.3</u>	<u>65.0</u>	<u>68.7</u>	<u>65.9</u>	<u>68.0</u>	<u>68.4</u>	<u>68.1</u>
	L	**77.6***	**70.5***	**73.7***	**71.7***	**73.1***	**73.7***	**73.4***
SetFit	S	67.9	62.2	66.8	64.0	65.0	66.7	65.4
Classfier	S	61.2	56.8	59.7	58.4	59.8	61.4	59.5
	L	66.4	56.0	65.3	56.6	60.0	61.9	61.0
				10-shot				
FastFit	S	<u>77.6</u>	70.5	73.7	<u>71.7</u>	<u>73.1</u>	<u>73.7</u>	<u>73.4</u>
	L	**79.2***	**74.8***	**77.4**	**74.1***	**75.7***	**74.9***	**76.0***
SetFit	S	74.7	69.8	73.5	71.4	72.0	72.9	72.4
Classfier	S	72.2	67.7	71.0	68.6	69.7	70.0	69.9
	L	77.5	<u>71.2</u>	<u>74.3</u>	71.3	72.5	72.7	73.3

Table 3: Accuracy results for FastFit and baselines across six languages, under 5/10-shot settings. Results show that FastFit consistently outperforms SetFit and the standard classifier. Notably, FastFit small consistently surpasses SetFit's small and standard large classifiers. Results marked with an asterisk (*) are statistically significant according to t-test ($p < 0.05$) when compared to the large standard classifier.

FastFit large outperforms SetFit small by 3.6% and the standard large classifier by 2.7%. Similarly, FastFit small exhibits improvements of 1.9% and 3.5% over SetFit small and the standard classifier, respectively.

It is noteworthy that FastFit demonstrates improvement when scaling from a small to a large model, with gains of 5.3% and 2.6% in the 5-shot and 10-shot settings, respectively. This enhancement highlights the fact that FastFit is not model-specific and thus is highly flexible for different sizes and types of models, unlike SetFit. Such flexibility is particularly crucial in few-shot settings where limited examples are available, highlighting the potential to train enhanced classifiers using domain- or language-specific models. Moreover, if unlabeled or pairwise data is available, using it for pretraining can lead to even further improvement.

Training Times for FastFit, SetFit, and the standard classifier are illustrated in Figure 1. FastFit exhibits faster training times compared to both SetFit and the standard classifier, with a 3-20x decrease, and training ranging between 35-70 seconds (See more results at App. §D). This can be attributed to a combination of technical and methodological factors. The improved implementation includes pretraining tokenization and FP16 training. Furthermore, the methodological advantage stems from using batch contrastive training, which leverages in-batch examples as negatives, in contrast to the triplet loss utilized by SetFit.

6 FastFit Ablation & Full Training

To further examine the contribution of some of our method modifications, we compare training with CLS and token-level similarity metrics, as well as training with a different number of batch repetitions. We conduct these experiments on three datasets: Hwu64, Banking77, and Clinc150, with 5 random splits, and average their results. We assess the effect of these modifications for both small and large models, with 5 and 10 shots.

In Table 4, we present the differences in performance caused by our changes; full results are available in App. §E. The Token-level similarity metric proves beneficial across all settings, with a more pronounced effect for smaller models and when less data is available (5-shot compared to 10-shot). Concerning the number of repetitions, we observe that, in most cases, adding repetitions helps. Additionally, it appears that overall, four repetitions are more effective than two. Regarding the relationship between the number of shots and the effectiveness of repetition, no clear connection is apparent. While an increase in the number of shots enhances effectiveness in small models, the opposite is observed for large models, where the effect decreases. Nevertheless, it seems that, in general, larger models benefit more from batch repetition.

Although our primary focus is few-shot classification, we also wanted to examine the effectiveness of FastFit when training on the full dataset. We conducted two sets of experiments. In the first,

Model	Shot	Similarity Level	Repetitions	
		Token	2	4
FastFit-S	5	1.33	-0.28	0.09
FastFit-S	10	0.85	0.09	0.24
FastFit-L	5	0.65	0.72	1.04
FastFit-L	10	0.36	0.55	0.78

Table 4: FastFit ablation experiments; Accuracy differences in training with token-level versus CLS similarity metrics and increasing augmentations repetitions. Token-level enhancements are more prominent in smaller models, especially in the 5-shot setting.

Model	C150	B77	H64	Avg.
Classfier-L	96.8	93.7	92.1	94.2
FastFit-S	97.1	93.8	92.7	94.5
FastFit-L	**97.5**	**94.2**	**93.0**	**94.9**

Table 5: FastFit accuracy results when training on the full data.

Model	EN	DE	JP	ES	FR	CN	Avg.
Classfier-B	88.3	85.7	83.9	86.9	86.3	84.9	86.0
mT5-B T2T	87.9	86.2	83.5	86.7	86.9	85.2	86.1
mT5-B Enc	89.0	86.8	85.8	86.8	87.2	85.8	86.9
FastFit-S	88.8	87.4	87.0	**87.9**	87.6	**86.7**	87.6
FastFit-L	**89.5**	**88.5**	**88.5**	87.4	**88.5**	**86.7**	**88.2**

Table 6: FastFit and baselines accuracy results on MASSIVE with full data training.

we compared FastFit-small, FastFit-large, and a large standard classifier on Hwu64, Banking77, and Clinc150. In the second, we compared FastFit-small and FastFit-large with a few base-sized multilingual baseline models on Msstive, using the set of six languages mentioned in §5.1. These baselines are based on the Msstive paper, where Classifier-B and mT5-B Encoder are standard classifiers based on XLM-R-BASE and mT5-Base with 270M and 258M parameters, respectively. mT5-B T2T is a text-2-text classifier with 580M parameters.

Results in Table 5 demonstrate that when training on all the data, FastFit-Small outperforms the large Classifier, and FastFit-Large performs even better. From Table 6, we can see that FastFit-Small outperforms all other baselines even with fewer than half the number of parameters. Moreover, FastFit-Large further improves performances by 0.6% on average. These results indicate that FastFit is not only a fast few-shot classifier but can also outperform even larger classifiers when training on the full dataset.

7 Related Work

For fine-tuning baselines, we focus on readily available methods. , including SetFit with its package, a standard classifier accessible through HF Transformers (Wolf et al., 2019), or LLMs through API calls. However, there are various few-shot classifiers, and we will briefly discuss a couple of them. QAID (Yehudai et al., 2023) proposed pre- and fine-tuning training stages with unsupervised and supervised loss, using ColBERT architecture, achieving SOTA results. T-Few (Liu et al., 2022), a parameter-efficient fine-tuning method based on T0 (Sanh et al., 2021), claims to be better and cheaper than In-Context Learning.

Regarding few-shot prompting of LLMs approaches, a question arises about whether our results will withstand stronger LLMs or improved prompting techniques. According to Loukas et al. (2023) we can deduce that FastFit outperforms GPT4 (OpenAI et al., 2023) with a fraction of the cost. Additionally, Milios et al. (2023) demonstrate that retrieval-based few-shot prompts can lead to improved results. However, it's worth noting that currently, these models remain slow and costly.

8 Conclusions

In this paper, we introduce FastFit, a novel few-shot text classification method accompanied by a Python package. Our results demonstrate that FastFit outperforms large language models (LLMs) such as Flan-XXL and Llama-2-chat-70B, as well as fine-tuning methods, including both standard and SetFit classifiers, readily available in existing packages. Notably, FastFit exhibits fast training and inference. We provide evidence that these results hold for both Multilingual and full-data training setups. We hope that FastFit's speed and simplicity will enhance its usability.

References

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

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.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron

Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shai Gretz, Alon Halfon, Ilya Shnayderman, Orith Toledo-Ronen, Artem Spector, Lena Dankin, Yannis Katsis, Ofir Arviv, Yoav Katz, Noam Slonim, and Liat Ein-Dor. 2023. Zero-shot topical text classification with LLMs - an experimental study. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9647–9676, Singapore. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2021. Supervised contrastive learning.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*.

Yen-Ting Lin, Alexandros Papangelis, Seokhwan Kim, Sungjin Lee, Devamanyu Hazarika, Mahdi Namazifar, Di Jin, Yang Liu, and Dilek Z. Hakkani-Tür. 2023. Selective in-context data augmentation for intent detection using pointwise v-information. In *Conference of the European Chapter of the Association for Computational Linguistics*.

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. *ArXiv*, abs/2205.05638.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019a. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566*.

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

Lefteris Loukas, Ilias Stogiannidis, Odysseas Diamantopoulos, Prodromos Malakasiotis, and Stavros Vassos. 2023. Making llms worth every penny: Resource-limited text classification in banking. *Proceedings of the Fourth ACM International Conference on AI in Finance*.

Aristides Milios, Siva Reddy, and Dzmitry Bahdanau. 2023. In-context learning for text classification with many labels. *ArXiv*, abs/2309.10954.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

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 Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan D. 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. *ArXiv*, abs/2110.08207.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. Ul2: Unifying language learning paradigms.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Asaf Yehudai, Matan Vetzler, Yosi Mass, Koren Lazar, Doron Cohen, and Boaz Carmeli. 2023. Qaid: Question answering inspired few-shot intent detection.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

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*.

A Full Code Example

Any dataset can be loaded directly from Huggingface Hub, For example:

```
from datasets import load_dataset
dataset =
load_dataset("mteb/banking77")
```

Then fast fit library can sample it down to the 5 or 10 shot format:

```
from fastfit import sample_dataset

dataset["train"] =sample_dataset(
   dataset["train"],
   label_column="label",
   num_samples=5
)
```

Then once the data is ready it can be serve as input to the Fast-Fit trainer together with other important inputs:

```
from fastfit import FastFitTrainer

trainer = FastFitTrainer(
   model_name_or_path=
      "roberta-large",
   label_column_name="label_text",
   text_column_name="text",
   dataset=dataset,
)

model = trainer.train()
results = trainer.evaluate()
```

Then we can save the model:

```
model.save_pretrained("fast-fit")
```

And could be loaded for inference with:

```
from fastfit import FastFit from
transformers import (
   AutoTokenizer,
   pipeline
)

model = FastFit.from_pretrained(
   "fast-fit"
)
tokenizer =
AutoTokenizer.from_pretrained(
   "roberta-large"
)

classifier = pipeline(
   "text-classification",
   model=model,
   tokenizer=tokenizer
)

print(classifier("Hello World!"))
```

B Data Statistics

In Table 7, we provide the data statistics for the classification datasets used in our work.

Dataset	#Train	#Vaild	#Test	#Intents	#Domains
Clinc150	15,000	3,000	4,500	150	10
BankingG77	8,622	1,540	3,080	77	1
Hwu64	8,954	1,076	1,076	64	21
MASSIVE	11,514	2,033	2,974	60	18

Table 7: Data statistics of the few-shot classification datasets.

C Multilingual Results

In Table 10, we present the experimental results using various backbone models for SetFit. We evaluated three models: (1) Monolingual sentence-transformer (ST) large, referred to as ST-L. (2) Regular Multilingual RoBERTa-large, denoted as XLM-R-L or simply L. (3) RoBERTa-Base Multilingual sentence-transformer model, labeled as ST-XB.

The results indicate that ST-L encounters difficulties with all non-English datasets, resulting in overall inferior performance. XLM-R-L exhibits lower proficiency in English but demonstrates improved results across all other languages. Lastly, ST-XB, with a comparable size to the small models

(125M vs. 110M), achieved similar, albeit slightly lower, results. These findings underscore SetFit's dependence on ST pre-trained models and highlight its limitations when such a model is unavailable, as in this experiment.

D Training Run Times Results

Here we present more training run time results for FastFit, SetFit, and a standard classifier. In 2 we present the run time for the small and large settings. In Table 9 we show the average training run time results.

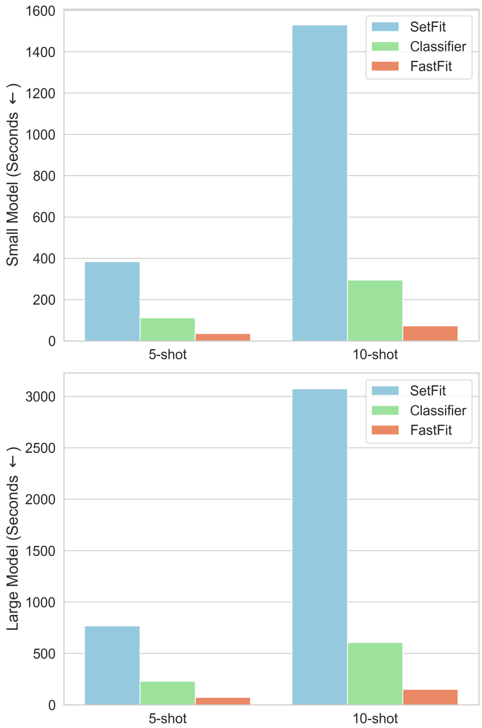

Figure 2: Training times (sec) for FastFit, SetFit, and standard classifier. FastFit training is faster both for the small model (top) and for the large model (bottom).

Table 8: Results

Model	Small		Large	
	5-shot	10-shot	5-shot	10-shot
FastFit	35.5	73.2	72.7	151.0
SetFit	384.1	1530.5	767.1	3073.7
classifier	112.0	294.8	230.6	606.7

Table 9: Training times (sec) for FastFit, SetFit, and standard classifier.

E Ablation Results

Here, we present the results for the ablations associated with Table 4. The first ablation is designed to measure the effect of the similarity metrics. Table 11 shows the results of the experiments with both CLS and token-level similarity metrics. In Table 12, we present the results without augmentation repetitions (1), and with 2 and 4 repetitions. Both ablations support our claim that the token-level similarity metric and an increased number of augmentation repetitions help.

F Short Video

Click here for our short presentation.

Method	Model	En	De	Ja	Es	Fr	Zh	Average
				5-shot				
FastFit	S	72.3	65.0	68.7	65.9	68.0	68.4	68.1
	L	**77.6***	**70.5***	**73.7***	**71.7***	**73.1***	**73.7***	**73.4***
SetFit	S	67.9	62.2	66.8	64.0	65.0	66.7	65.4
	ST-L	74.0	50.3	41.3	53.6	52.1	39.6	51.8
	L	66.1	60.8	64.8	50.1	61.3	43.6	57.8
	ST-XB	74.0	62.3	64.8	62.0	62.3	65.1	65.1
				10-shot				
FastFit	S	77.6	70.5	73.7	71.7	73.1	73.7	73.4
	L	**79.2***	**74.8***	**77.4**	**74.1***	**75.7***	**74.9***	**76.0***
SetFit	S	74.7	69.8	73.5	71.4	72.0	72.9	72.4
	ST-L	78.3	61.4	53.4	64.0	63.2	48.3	61.4
	L	74.5	69.1	72.5	69.7	70.7	59.2	69.3
	ST-XB	78.3	68.7	72.9	70.1	70.5	72.3	72.1

Table 10: Accuracy results for FastFit and baselines across six languages, under 5/10-shot settings. Results with few SetFit versions but no one surpasses SetFit small. We experimenting here with sentence-transformer (ST) large monolingual, multilingual base, and non-ST multilingual large.

Method	Shots	Sim. metric	C150	B77	H64	Average
FastFit-small	5	CLS	88.9	78.6	78.5	82.0
	5	TOK.	90.2	80.0	79.7	**83.3**
	10	CLS	92.4	84.7	83.8	86.9
	10	TOK.	93.3	85.4	84.7	**87.8**
FastFit-large	5	CLS	91.6	81.7	82.4	85.2
	5	TOK.	92.3	82.9	82.4	**85.9**
	10	CLS	94.1	87.6	86.3	89.4
	10	TOK.	94.8	88.0	86.4	**89.7**

Table 11: Ablation results with CLS and token-level similarity metrics. The average results that scored the highest for each model size and shot number are highlighted in bold.

Method	Shots	Repet.	C150	B77	H64	Average
FastFit-small	5	1	90.3	80.3	79.1	83.2
	5	2	89.8	79.8	79.2	82.9
	5	4	90.2	80.0	79.7	**83.3**
FastFit-small	10	1	93.3	85.3	84.1	87.6
	10	2	93.2	85.3	84.5	87.6
	10	4	93.3	85.4	84.7	**87.8**
FastFit-Large	5	1	91.6	82.0	81.0	84.8
	5	2	92.0	82.4	82.3	85.6
	5	4	92.3	82.9	82.4	**85.9**
FastFit-Large	10	1	94.2	87.3	85.2	88.9
	10	2	94.6	87.7	86.1	89.5
	10	4	94.8	88.0	86.4	**89.7**

Table 12: Ablation results with varying repetition numbers. The bolded values represent the highest-scoring average results for each model size and shot number.