STraTA: Self-Training with Task Augmentation for Better Few-shot Learning

Tu Vu[1,2]★, **Minh-Thang Luong**[2], **Quoc V. Le**[2], **Grady Simon**[2], **Mohit Iyyer**[1]
University of Massachusetts Amherst[1]
Google Research[2]
{tuvu,miyyer}@cs.umass.edu
{ttvu,thangluong,qvl,gradys}@google.com

Abstract

Despite their recent successes in tackling many NLP tasks, large-scale pre-trained language models do not perform as well in few-shot settings where only a handful of training examples are available. To address this shortcoming, we propose **STraTA**, which stands for **S**elf-**Tra**ining with **T**ask **A**ugmentation, an approach that builds on two key ideas for effective leverage of unlabeled data. First, STraTA uses *task augmentation*, a novel technique that synthesizes a large amount of data for auxiliary-task fine-tuning from target-task unlabeled texts. Second, STraTA performs *self-training* by further fine-tuning the strong base model created by task augmentation on a broad distribution of pseudo-labeled data. Our experiments demonstrate that STraTA can substantially improve sample efficiency across 12 few-shot benchmarks. Remarkably, on the SST-2 sentiment dataset, STraTA, with only 8 training examples per class, achieves comparable results to standard fine-tuning with 67K training examples. Our analyses reveal that task augmentation and self-training are both complementary and independently effective.

1 Introduction

Recent advances in NLP demonstrate the effectiveness of applying large-scale pretrained language models to downstream tasks (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Lan et al., 2020; Raffel et al., 2020; Brown et al., 2020; He et al., 2021). While these models have achieved state-of-the-art results on many NLP benchmarks, they struggle when given limited training data. For instance, Devlin et al. (2019) find that BERT is prone to degenerate performance on small datasets. While enormous language models like GPT-3 (Brown et al., 2020) exhibit the ability to solve a new task from only a few examples without any fine-tuning, their performance still lags far behind state-of-the-art fine-tuning results. Manually annotating large amounts of training data will likely improve performance but can also be prohibitively expensive to obtain for many tasks and domains. In this paper, we propose STraTA, an approach that combines two *complementary* methods, **S**elf-**Tra**ining and **T**ask **A**ugmentation, to effectively leverage unlabeled data, which is comparatively cheaper to obtain.[1]

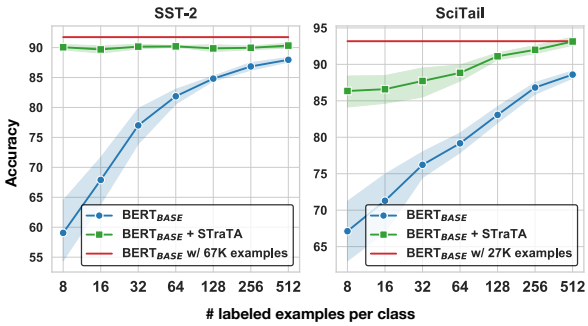

Figure 1: Our **S**elf-**Tra**ining with **T**ask **A**ugmentation (**STraTA**) approach substantially improves sample efficiency across different tasks. For example, when given only 8 labeled examples per class from the SST-2 sentiment dataset, STraTA is competitive with standard fine-tuning on 67K examples; on the SciTail entailment dataset, with 512 labeled examples per class, STraTA surpasses standard fine-tuning on 27K examples.

At a high level, task augmentation exploits unlabeled text from the domain of a given target task to simulate a large amount of *in-domain* training data for the *auxiliary* task of natural language inference (NLI), which is then used to train a given model before applying it to the target task. To achieve this, we first build an NLI data generator by fine-tuning a pre-trained generative language model on the MNLI data set (Williams et al., 2018) in a *text-to-text* format. Then, given a target task (e.g., sentiment analysis) with unlabeled

★ Work done as a student researcher at Google Brain.

[1]Our code and pre-trained models will be available at https://github.com/google-research/google-research/tree/master/STraTA.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5715–5731
November 7–11, 2021.

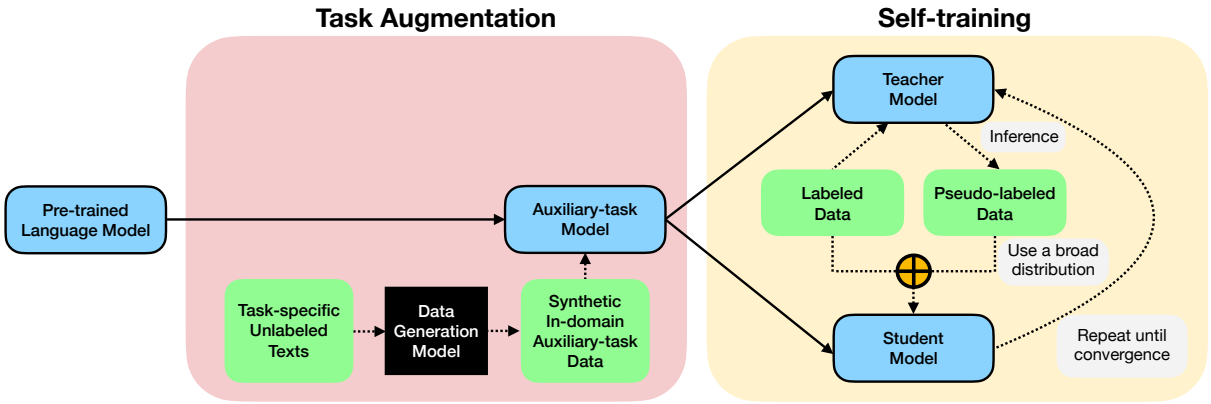

Figure 2: An illustration of our **S**elf-**Tra**ining with **T**ask **A**ugmentation (**STraTA**) approach. In task augmentation, we train an NLI data generation model and use it to synthesize a large amount of *in-domain* NLI training data for each given target task, which is then used for auxiliary (intermediate) fine-tuning. Our self-training algorithm iteratively learns a better model using a concatenation of labeled and pseudo-labeled examples. At each iteration, we always start with the auxiliary-task model produced by task augmentation and train on a broad distribution of pseudo-labeled data.

texts (e.g., *his acting was really awful*), we use the NLI data generator to generate NLI examples (e.g., [*his acting was really awful*, *he gave an incredible performance*, *contradiction*]). We show that task augmentation alone can significantly improve downstream performance across different tasks, generally outperforming other fine-tuning approaches, such as target-task language model fine-tuning (Howard and Ruder, 2018; Gururangan et al., 2020) and intermediate-task fine-tuning on MNLI (Phang et al., 2019), in both high- and low-data regimes.

Having obtained a strong auxiliary-task model with task augmentation, STraTA uses this model as a base model for self-training. Specifically, at each at iteration, the base model is fine-tuned using the available labeled data for the target task. Then, the resulting model's predictions on unlabeled examples[2] are used as pseudo-labels to augment the original labeled data set. The newly formed labeled data set is then used to learn a better model in the next iteration, and this procedure is repeated for a number of iterations until a stopping criterion is reached. While self-training has been extensively studied (Rosenberg et al., 2005; McClosky et al., 2006; He et al., 2020; Xie et al., 2020b; Du et al., 2021), our experiments reveal that using a strong base model and training on a broad distribution of pseudo-labeled data are key factors for successful deployment in NLP.

[2]We use the term *unlabeled text* to refer to pieces of text (e.g., sentences) from the target domain, and the term *unlabeled examples* to refer to examples that can be annotated using the set of class labels for the target task.

Using our STraTA approach, we are able to significantly improve sample efficiency, in terms of both performance and variance, across 12 NLP benchmark datasets. For instance, on the SST-2 sentiment dataset (Socher et al., 2013), with only 8 training examples per class, we achieve comparable results to standard fine-tuning with 67K training examples (see Figure 1).

Our main contributions are as follows:

1. We propose task augmentation, a novel data augmentation-based fine-tuning method, and show its effectiveness in comparison to other competing fine-tuning approaches.

2. We propose a simple yet effective self-training algorithm and highlight important ingredients for successful self-training, which we hope will enable the wider adoption of self-training in NLP.

3. With STraTA, we demonstrate the effectiveness of combining task augmentation and self-training in improving sample efficiency across NLP benchmarks.

2 Task augmentation

Labeled data is often expensive and time-consuming to obtain, which motivates approaches that learn from both labeled and unlabeled data. More formally, assume we are given a target task $\mathcal{T}$ with a labeled data set $\mathcal{L}_{\mathcal{T}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{M}$ and an unlabeled data set $\mathcal{U}_{\mathcal{T}} = \{(\boldsymbol{x}_j)\}_{j=1}^{N}$. The unlabeled data $\mathcal{U}_{\mathcal{T}}$ can be created artificially by

removing the ground-truth labels $\boldsymbol{y}$ from $\boldsymbol{\mathcal{L}_\mathcal{T}}$ (as in our main experiments), or it can come from additional unlabeled texts from the target domain or from related datasets/domains (see Section 5). Our methods, *task augmentation* and *self-training*, take advantage of the unlabeled data $\boldsymbol{\mathcal{U}_\mathcal{T}}$ to maximize performance on the target task $\boldsymbol{\mathcal{T}}$, even when the number of labeled examples M is small (e.g., $M = 16$). In this section, we first present a framework and implementation for task augmentation, which uses natural language inference (NLI) as an auxiliary (intermediate) training task to improve downstream performance.

2.1 A framework for task augmentation

Task augmentation builds on a recent body of NLP research on intermediate-task training (Phang et al., 2019; Vu et al., 2020), in which a pre-trained language model, such as BERT, is fine-tuned on an auxiliary task before the target task.[3] In previous work on intermediate fine-tuning, the auxiliary dataset used is a fixed target task-independent dataset, such as MNLI or SQuAD (Rajpurkar et al., 2016). An obvious limitation of this choice is the domain mismatch between the auxiliary and target tasks, which our proposed task augmentation method addresses. More specifically, we fine-tune a pre-trained generative language model and use it to synthesize a large amount of *in-domain* training data from $\boldsymbol{\mathcal{U}_\mathcal{T}}$ for an auxiliary task $\boldsymbol{\mathcal{A}}$, which is then used to improve performance of a model on the target task $\boldsymbol{\mathcal{T}}$ (Figure 2, left).[4] In this work, we choose NLI as the auxiliary task for two main reasons: (1) NLI has been shown to be an effective auxiliary task for a variety of target tasks (Conneau et al., 2017; Phang et al., 2019), and (2) existing NLI datasets contain large training sets, which allows us to train a reliable data generator.

Generating synthetic NLI data: To obtain an NLI data generator, we fine-tune the pre-trained T5-3B model (Raffel et al., 2020) on MNLI, which contains 393K sentence pairs labeled as {*entailment, contradiction, neutral*}. We cast each MNLI training example $(\boldsymbol{sent}_A, \boldsymbol{sent}_B) \rightarrow \boldsymbol{label}$ into a *text-to-text* format $(\boldsymbol{label}, \boldsymbol{sent}_A) \rightarrow \boldsymbol{sent}_B$ to obtain fine-tuning examples that look like [*entailment, the facts are accessible to you* $\rightarrow$ *you have access to the facts*].[5] We fine-tune T5 on this dataset with a constant learning rate of 0.001 for $2^{16} = 65,536$ steps using the Adafactor optimizer (Shazeer and Stern, 2018). The fine-tuned T5 data generator produces augmented examples for all target datasets. Specifically, at inference time, we feed the model an NLI label (e.g., *entailment*) and an unlabeled sentence $\boldsymbol{x}_j$ from the target domain to produce some output sentence $\boldsymbol{x}_k$: $(entailment, \boldsymbol{x}_j) \rightarrow \boldsymbol{x}_k$ (see Appendix B for example outputs). Data for intermediate fine-tuning is then formed by creating examples like $(\boldsymbol{x}_j, \boldsymbol{x}_k) \rightarrow entailment$. This approach has several advantages: (1) training labels are *free*, and (2) by overgeneration, a large amount of *in-domain* NLI training data can be produced even for target tasks with small datasets.

Overgeneration and filtering: Following Puri et al. (2020), we perform overgeneration and filtering to increase the quantity and quality of synthetic NLI training data. Concretely, we generate 100 output samples per input with top-k ($k = 40$) sampling (duplicates are removed) and use a BERT model fine-tuned on MNLI (in the original format) as an NLI classifier to filter synthetic training examples. We keep a synthetic example if the NLI classifier produces the same label as that fed to the NLI data generator and is also confident about its prediction.[6] For all experiments, we perform intermediate fine-tuning on examples from both the original MNLI dataset and the final filtered task augmentation dataset.[7]

3 Self-training

While task augmentation uses unlabeled texts to produce synthetic data for an intermediate task, *self-training* is a *complementary* approach that improves a model by training directly on the target task using pseudo-labeled examples. In this section, we explore a simple self-training algorithm in which a model learns to improve itself

[3] This process differs from traditional data augmentation approaches (e.g., lexical substitution, or back-translation), which yield negligible improvements when combined with large-scale pre-trained language models (Wei and Zou, 2019; Yang et al., 2020).

[4] Traditional data augmentation is a special case of our framework where the auxiliary task is identical to the target task ($\boldsymbol{\mathcal{A}} \equiv \boldsymbol{\mathcal{T}}$).

[5] We fine-tune a separate T5 model per class label. To overcome biases in MNLI where the hypotheses are usually shorter than the premises, we also include reversed examples: $(\boldsymbol{reversed\ label}, \boldsymbol{sent}_B) \rightarrow \boldsymbol{sent}_A$.

[6] We use an example when its predicted probability exceeding a certain threshold τ. We choose a value for τ in $[0.3, 0.4, \ldots, 0.9]$ for each target task based on performance on the original MNLI development set.

[7] A two-stage intermediate fine-tuning procedure where the model is first trained on the synthetic data before being fine-tuned on the original data typically works better, and this is used in our experiments.

Algorithm 1: Our self-training algorithm

initialization

$t = 0$

Form a base model $\boldsymbol{f}_0$, which is initialized with pre-trained parameters from a pre-training/intermediate fine-tuning stage, and then learn a teacher model $\boldsymbol{f}_1$ by training $\boldsymbol{f}_0$ on the original labeled data set $\boldsymbol{\mathcal{L}}$.

repeat

$t = t + 1$

1. Use the current teacher model $\boldsymbol{f}_t$ to annotate (for $t = 1$) or re-annotate (for $t > 1$) all of the examples in $\boldsymbol{\mathcal{U}}$ to obtain a set $\widetilde{\boldsymbol{\mathcal{U}}}$ of pseudo-labeled examples.

2. Add the whole set $\widetilde{\boldsymbol{\mathcal{U}}}$ of pseudo-labeled examples to the original labeled data set $\boldsymbol{\mathcal{L}}$ to form a new labeled data set.

3. Learn a student model $\boldsymbol{f}_{t+1}$ by training the base model $\boldsymbol{f}_0$ on the current labeled data set and optionally fine-tune it on $\boldsymbol{\mathcal{L}}$. The resulting student model $\boldsymbol{f}_{t+1}$ is used as a teacher for the next iteration.

until *convergence or the maximum number of iterations is reached*

using its predictions on unlabeled examples from a given target task. Our method differs from traditional self-training methods in that we leverage a strong base model and allow it to learn from all available pseudo-labeled examples at every iteration, regardless of model confidence. Formally, given a target task $\boldsymbol{\mathcal{T}}$ with a small labeled data set $\boldsymbol{\mathcal{L}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{M}$ and an unlabeled data set $\boldsymbol{\mathcal{U}} = \{(\boldsymbol{x}_j)\}_{j=1}^{N}$, where $M \ll N$, we summarize our self-training algorithm in Algorithm 1.

Starting with a strong base model: An important ingredient in self-training algorithms is the base model $\boldsymbol{f}_0$. Successful self-training typically requires a good base model, which can provide a large proportion of "correct" predictions or pseudo-labels on unlabeled examples; otherwise, errors can be propagated or magnified by later stages of self-training. At each self-training iteration, we always start from the same base model $\boldsymbol{f}_0$, which is initialized with pre-trained parameters from a pre-training/intermediate fine-tuning stage (e.g., the auxiliary task training stage in task augmentation),[8] and then fine-tune all of its parameters using the available labeled and pseudo-labeled data.[9]

Self-training on a broad distribution of pseudo-labeled data: Another important factor is the selection of pseudo-labeled examples at each self-training iteration. Traditional self-training approaches usually select a small set of examples where the current teacher model $\boldsymbol{f}_t$ is sufficiently confident (e.g., the probability of the predicted class label is above a threshold) to add to the labeled data set at each iteration until the unlabeled data pool $\boldsymbol{\mathcal{U}}$ is exhausted. This can be problematic as state-of-the-art language models like BERT are overconfident and poorly calibrated (Jiang et al., 2021). In preliminary experiments, we tried several calibration methods, including temperature scaling (Guo et al., 2017), label smoothing (Müller et al., 2019), and confidence penalties (Pereyra et al., 2017), but all of which failed to fully address this problem. Instead, we encourage learning from a "natural" broad distribution of pseudo-labeled data by adding the whole set $\widetilde{\boldsymbol{\mathcal{U}}}$ of pseudo-labeled examples to the original labeled data set $\boldsymbol{\mathcal{L}}$ at each self-training iteration.[10] At each iteration $t > 1$, we also re-annotate all of the examples in the original unlabeled data pool $\boldsymbol{\mathcal{U}}$ with $\boldsymbol{f}_t$, as we expect $\boldsymbol{f}_t$ is better than $\boldsymbol{f}_{t-1}$.

4 Experiments

We perform experiments across 12 different NLP datasets and three different data regimes (including a few-shot setting). Task augmentation consistently improves over prior fine-tuning approaches in all three regimes, and the combination of self-training and task augmentation, STraTA, results in higher performance and lower variance than competing approaches when given only 8 labeled examples per class from each dataset.

4.1 Datasets & data regimes

The datasets used in our study (Table 1)[11] come from two common language understanding benchmarks: GLUE (Wang et al., 2019b) and SentEval (Conneau and Kiela, 2018). Due to restricted test set access for GLUE datasets, we held out a small subset of the training set for validation and

[8] We find empirically that starting from the base model $\boldsymbol{f}_0$ works better than from the model $\boldsymbol{f}_{t-1}$ obtained in the previous iteration.

[9] He et al. (2020) find that further fine-tuning the resulting model on the original labeled data set $\boldsymbol{\mathcal{L}}$ improves machine translation models. We use development set performance to determine whether or not to perform this fine-tuning step for each dataset.

[10] We find that removing examples with the lowest-confidence pseudo labels can be helpful for some tasks. One can use a development set, upon availability, to assess if this filtering is necessary.

[11] Appendix A contains more details about characteristics and associated evaluation metrics for each dataset.

Task	|Train|
text classification/regression	
SNLI (Bowman et al., 2015)	570K
MNLI (Williams et al., 2018)	393K
QQP (Iyer et al., 2017)	364K
QNLI (Wang et al., 2019b)	105K
SST-2 (Socher et al., 2013)	67K
SciTail (Khot et al., 2018)	27K
SST-5 (Socher et al., 2013)	8.5K
STS-B (Cer et al., 2017)	7K
SICK-E (Marelli et al., 2014)	4.5K
SICK-R (Marelli et al., 2014)	4.5K
CR (Hu and Liu, 2004)	4K
MRPC (Dolan and Brockett, 2005)	3.7K
RTE (Dagan et al., 2005, et seq.)	2.5K

Table 1: Datasets used in our experiments.

report results on the original development set. The training set without ground-truth labels is used as unlabeled data $\mathcal{U}_\mathcal{T}$ for each task.

We consider three data regimes by varying the amount of labeled training data across the downstream tasks: FULL (all labeled training data), LIMITED (1024 random labeled training examples), and FEW-SHOT (8 random labeled training examples per class).[12] Since fine-tuning BERT can be unstable on small datasets (Devlin et al., 2019), we perform 10 random restarts where there are less than 10K training examples and report the mean and standard deviation.[13] Since large development sets are impractical in low-resource settings (Oliver et al., 2018; Kann et al., 2019), we randomly sample 256 development examples for each task in the LIMITED and FEW-SHOT regimes. Additionally, in the FEW-SHOT regime, we experiment with a real-world scenario where there is no development set access.

4.2 Setup

As in Devlin et al. (2019), our input format for all tasks contains a [CLS] token followed by a single text segment or a concatenation of text segments (e.g., a premise-hypothesis pair) separated with a [SEP] token. We feed the final [CLS] representation into a task-specific classification layer and fine-tune all the parameters end-to-end on the downstream tasks. For both fine-tuning and self-training, we perform early stopping based on development set performance. We use the Transformers library (Wolf et al., 2019) and its recommended hyperparameters for all experiments.[14]

4.3 Methods

We experiment with task augmentation (**TA**) and self-training (**ST**) individually, as well as the combined approach **STraTA**, which uses the auxiliary-task model from task augmentation as the base model for self-training. We compare our methods to the following baselines:

LMFT & ITFT$_{\text{MNLI}}$: We compare our methods against commonly-used fine-tuning approaches, including target-task language model fine-tuning (**LMFT**; Howard and Ruder, 2018; Gururangan et al., 2020)—in which a model is first trained with the language model objective on task-specific unlabeled data before being fine-tuned on the target task—and intermediate-task fine-tuning on MNLI (**ITFT$_{\text{MNLI}}$**; Phang et al., 2019)—which first trains a model on MNLI before fine-tuning it on the target task.

Prompt-based/entailment-based fine-tuning approaches: We also include results from recent work on prompt-based (**LM-BFF**; Gao et al., 2021) and entailment-based (**EFL**; Wang et al., 2021) fine-tuning,[15] which has been shown to outperform the GPT-3-style "in-context learning" approach (Brown et al., 2020) for few-shot learning. These approaches do not assume access to task-specific unlabeled data and are not directly comparable to our methods due to differences in model architecture and experimental settings.

Du et al. (2021)'s self-training approach: Most related to our work, Du et al. (2021) propose a data augmentation method called SentAugment, which retrieves a large amount of "in-domain" data for a given task from a large bank of Web sentences. A base model trained using task-specific labeled data is then applied to obtain pseudo-labels for the retrieved sentences, which is then added to the original training set to train a better model. Their approach is complementary to ours, and combining these approaches is a promising direction for future work.

[12] For regression tasks, we partition the output interval $[0, 5]$ into five bins and sample 8 examples from each bin.

[13] We resample examples for each restart.

[14] While individual task performance can likely be further improved with more involved hyperparameter tuning, we standardize hyperparameters across tasks to cut down on computational expense. Our experiments were conducted on Google Cloud with 100% renewable energy.

[15] Results taken from Wang et al. (2021).

Model	SNLI	QQP	QNLI	SST-2	SciTail	SST-5	STS-B	SICK-E	SICK-R	CR	MRPC	RTE
						FULL						
$\text{BERT}_{\text{LARGE}}$	91.1	88.4	91.9	92.4	95.3	$53.7_{0.9}$	$89.6_{0.2}$	$87.9_{0.6}$	$84.4_{0.4}$	$91.7_{0.6}$	$89.0_{0.8}$	$68.6_{7.2}$
+ LMFT	91.0	88.1	90.4	93.5	95.3	$54.0_{0.4}$	$89.5_{0.2}$	$87.7_{0.5}$	$84.0_{0.5}$	$91.6_{0.8}$	$89.5_{1.0}$	$66.5_{7.3}$
+ $\text{ITFT}_{\text{MNLI}}$	91.1	88.2	91.6	93.5	96.5	$54.0_{0.8}$	$90.3_{0.3}$	$89.9_{0.2}$	$86.3_{0.3}$	$92.0_{0.6}$	$89.7_{0.9}$	$82.3_{1.4}$
+ TA	**91.9**	**88.5**	**92.5**	**94.7**	**96.9**	$\mathbf{55.7}_{0.8}$	$\mathbf{90.9}_{0.2}$	$\mathbf{90.7}_{0.3}$	$\mathbf{87.0}_{0.3}$	$\mathbf{93.3}_{0.6}$	$\mathbf{90.8}_{0.7}$	$\mathbf{83.8}_{1.1}$
						LIMITED *(1024 total training examples)*						
$\text{BERT}_{\text{LARGE}}$	$77.4_{0.6}$	$74.1_{1.0}$	$81.7_{0.9}$	$89.8_{0.6}$	$90.9_{0.7}$	$49.1_{1.3}$	$88.2_{0.4}$	$84.8_{0.7}$	$80.2_{0.4}$	$91.2_{0.6}$	$85.7_{1.7}$	$66.8_{2.7}$
+ LMFT	$75.8_{1.5}$	$71.6_{0.5}$	$80.5_{2.0}$	$88.9_{0.8}$	$87.7_{2.3}$	$49.2_{3.1}$	$88.4_{0.4}$	$83.2_{0.6}$	$78.5_{0.6}$	$90.9_{0.7}$	$84.9_{1.1}$	$65.2_{3.4}$
+ $\text{ITFT}_{\text{MNLI}}$	$85.2_{0.4}$	$74.0_{0.5}$	$83.5_{0.5}$	$90.0_{0.8}$	$92.1_{1.1}$	$49.4_{1.2}$	$87.8_{0.8}$	$88.8_{0.5}$	$83.2_{0.7}$	$91.3_{0.7}$	$86.4_{0.9}$	$81.1_{1.3}$
+ TA	$\mathbf{87.3}_{0.3}$	$\mathbf{75.7}_{0.5}$	$\mathbf{85.0}_{0.5}$	$\mathbf{91.7}_{0.7}$	$\mathbf{92.3}_{1.1}$	$\mathbf{51.4}_{1.0}$	$\mathbf{89.0}_{0.6}$	$\mathbf{89.4}_{0.4}$	$\mathbf{84.3}_{0.4}$	$\mathbf{92.6}_{0.6}$	$\mathbf{88.0}_{0.8}$	$\mathbf{82.9}_{1.8}$
						FEW-SHOT *(8 training examples per class)*						
$\text{BERT}_{\text{BASE}}$	$43.7_{2.2}$	$55.9_{6.5}$	$59.0_{10.9}$	$59.1_{8.4}$	$67.1_{6.6}$	$30.5_{2.0}$	$73.6_{4.5}$	$61.3_{4.1}$	$59.7_{2.7}$	$65.2_{8.2}$	$72.4_{10.2}$	$51.4_{2.5}$
+ LMFT	$45.2_{3.9}$	$57.2_{6.2}$	$57.6_{9.1}$	$64.9_{8.7}$	$64.0_{8.0}$	$33.4_{1.9}$	$75.4_{4.4}$	$59.3_{4.0}$	$58.3_{2.0}$	$72.4_{6.0}$	$73.9_{8.6}$	$50.9_{3.9}$
+ $\text{ITFT}_{\text{MNLI}}$	$75.2_{5.7}$	$63.7_{7.0}$	$62.8_{5.1}$	$76.8_{7.2}$	$75.8_{5.6}$	$35.0_{2.6}$	$80.2_{1.1}$	$80.4_{1.9}$	$73.5_{2.7}$	$79.2_{3.6}$	$74.3_{8.0}$	$62.2_{13.5}$
+ TA	$83.3_{0.8}$	$68.7_{1.5}$	$70.1_{3.4}$	$80.3_{6.6}$	$78.5_{3.2}$	$37.4_{3.0}$	$80.7_{1.5}$	$81.1_{2.4}$	$75.9_{1.8}$	$86.5_{2.2}$	$74.5_{6.5}$	$67.6_{7.1}$
+ ST	$65.0_{5.8}$	$69.9_{5.9}$	$71.6_{11.3}$	$62.7_{10.4}$	$68.6_{8.3}$	$33.9_{3.5}$	$80.5_{2.2}$	$68.1_{4.5}$	$64.0_{2.4}$	$78.2_{6.3}$	$80.5_{1.8}$	$50.7_{3.1}$
+ $\text{ITFT}_{\text{MNLI}}$ + ST	$83.2_{0.3}$	$70.7_{5.9}$	$81.5_{1.2}$	$88.0_{2.1}$	$83.7_{4.4}$	$39.5_{2.0}$	$84.2_{0.8}$	$81.8_{2.6}$	$75.8_{2.2}$	$85.6_{2.3}$	$80.6_{1.2}$	$62.5_{12.0}$
+ STraTA	$\mathbf{85.7}_{0.2}$	$\mathbf{74.5}_{0.4}$	$\mathbf{82.1}_{0.5}$	$\mathbf{90.1}_{0.8}$	$\mathbf{86.3}_{3.5}$	$\mathbf{41.3}_{1.5}$	$\mathbf{84.7}_{0.5}$	$\mathbf{84.9}_{1.2}$	$\mathbf{77.6}_{1.6}$	$\mathbf{90.5}_{0.8}$	$\mathbf{81.0}_{0.8}$	$\mathbf{70.6}_{2.4}$
$\text{BERT}_{\text{LARGE}}$	$43.1_{4.4}$	$58.5_{4.7}$	$64.4_{6.1}$	$66.1_{8.7}$	$68.8_{9.5}$	$35.2_{1.3}$	$74.6_{3.8}$	$66.5_{4.5}$	$66.6_{3.3}$	$72.0_{6.0}$	$79.9_{2.0}$	$53.1_{3.3}$
+ LMFT	$39.6_{2.6}$	$52.7_{4.7}$	$52.2_{1.6}$	$66.3_{9.3}$	$66.4_{10.6}$	$36.8_{2.9}$	$75.4_{9.4}$	$58.8_{6.9}$	$51.6_{7.0}$	$75.6_{5.9}$	$80.5_{2.4}$	$52.8_{4.8}$
+ $\text{ITFT}_{\text{MNLI}}$	$79.9_{3.1}$	$62.6_{9.0}$	$64.5_{4.4}$	$80.7_{5.0}$	$72.3_{11.2}$	$36.4_{2.1}$	$75.5_{4.0}$	$77.8_{3.8}$	$73.5_{2.8}$	$82.6_{3.0}$	$72.8_{7.9}$	$69.7_{14.6}$
+ TA	$84.8_{0.7}$	$64.6_{6.3}$	$71.5_{4.0}$	$85.5_{1.4}$	$79.0_{4.5}$	$38.5_{3.0}$	$78.9_{2.4}$	$81.2_{3.9}$	$77.5_{1.4}$	$88.6_{1.3}$	$78.2_{6.6}$	$77.0_{6.3}$
+ ST	$69.3_{9.2}$	$74.3_{1.2}$	$85.4_{1.7}$	$81.9_{12.2}$	$79.9_{4.8}$	$42.0_{1.5}$	$82.8_{2.3}$	$77.3_{3.1}$	$73.1_{2.3}$	$88.1_{1.3}$	$81.2_{0.5}$	$53.9_{4.3}$
+ $\text{ITFT}_{\text{MNLI}}$ + ST	$85.4_{0.3}$	$74.8_{0.7}$	$86.1_{1.1}$	$89.7_{0.7}$	$86.2_{4.2}$	$42.2_{2.0}$	$84.1_{1.7}$	$84.3_{2.0}$	$78.4_{1.3}$	$89.3_{1.0}$	$81.4_{1.2}$	$72.7_{5.4}$
+ STraTA	$\mathbf{87.3}_{0.3}$	$\mathbf{75.1}_{0.2}$	$\mathbf{86.4}_{0.8}$	$\mathbf{91.7}_{0.7}$	$\mathbf{87.3}_{2.9}$	$\mathbf{43.0}_{2.3}$	$\mathbf{84.5}_{1.6}$	$\mathbf{86.3}_{1.8}$	$\mathbf{79.0}_{1.0}$	$\mathbf{90.0}_{0.6}$	$\mathbf{81.5}_{0.7}$	$\mathbf{77.1}_{5.4}$
Prompt-based (LM-BFF; Gao et al., 2021) and entailment-based (EFL; Wang et al., 2021) fine-tuning approaches												
$\text{RoBERTa}_{\text{LARGE}}$	$38.4_{1.3}$	$58.8_{9.9}$	$52.7_{1.8}$	$60.5_{3.1}$	–	–	$24.5_{8.4}$	–	–	$61.9_{5.1}$	$76.1_{3.9}$	$55.0_{1.3}$
+ LM-BFF	$52.0_{1.7}$	$\mathbf{68.2}_{1.2}$	$61.8_{3.2}$	$79.9_{6.0}$	–	–	$66.0_{3.2}$	–	–	$88.6_{2.3}$	$\mathbf{78.5}_{2.3}$	$63.3_{2.1}$
+ EFL	$\mathbf{81.0}_{1.1}$	$67.3_{2.6}$	$\mathbf{68.0}_{3.4}$	$\mathbf{90.8}_{1.0}$	–	–	$\mathbf{71.0}_{1.3}$	–	–	$\mathbf{92.3}_{0.4}$	$76.2_{1.3}$	$\mathbf{85.8}_{0.9}$

Table 2: STraTA significantly improves results across 12 NLP benchmark datasets (numbers in the subscript indicate the standard deviation across 10 random seeds). See Appendix C for full results.

4.4 Results and Discussion

Table 2 shows the main results of our experiments with task augmentation and self-training. Below, we first provide an overview of these results before analyzing them in more detail.

Baselines: **LMFT** is not always helpful and can even hurt performance (e.g., on QNLI, a task built from Wikipedia, which is part of BERT's pre-training data). Du et al. (2021) also observe a decrease in performance when using **LMFT** with task-specific in-domain unlabeled data retrieved from Web data. $\textbf{ITFT}_{\textbf{MNLI}}$ significantly outperforms **LMFT** in many cases, particularly on target tasks closely related to MNLI.

Task augmentation significantly improves results on downstream tasks: The first three blocks of Table 2 show the results for **TA**, which improves almost all target tasks across all three data regimes. **TA** even improves results on SNLI in the FULL regime, where there is a large amount of labeled data available (570K examples). Changing the data regimes significantly impacts the average absolute performance gain over the vanilla $\text{BERT}_{\text{LARGE}}$ across target tasks, which is lowest in FULL regime (+2.7%) and highest in the FEW-SHOT regime (+13.0%). SNLI (+41.7%) and RTE (+23.9%) benefit the most from **TA** in the FEW-SHOT regime. **TA** also significantly outperforms both **LMFT** and $\textbf{ITFT}_{\textbf{MNLI}}$, particularly in the low-data regimes (+16.4% and +4.8%, respectively).

Adding self-training further boosts downstream performance when task-specific unlabeled examples are available: The third block of Table 2 shows that in the FEW-SHOT regime, adding **ST** to **TA**, which results in **STraTA**, further boosts downstream performance. In particular, **STraTA** performs the best across target tasks, achieving up to +44.2% absolute improvement on SNLI over $\text{BERT}_{\text{LARGE}}$. Overall, **STraTA** provides an average absolute performance gain of +20.9% and +18.4% for $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, respectively. Using **ST** alone also leads to large improvements over the vanilla BERT models; however, the performance gain largely depends on the target task.

Model	SST-2	SST-5	CR
Ours (8 examples per class)			
$\text{BERT}_{\text{BASE}}$	$69.8_{6.5}$	$32.8_{2.0}$	$73.1_{0.5}$
+ TA	$85.5_{0.6}$	$41.0_{0.8}$	$88.7_{0.2}$
+ ST	$74.9_{9.0}$	$38.3_{0.8}$	$85.6_{1.8}$
+ STraTA	$\mathbf{90.8}_{0.6}$	$\mathbf{43.1}_{1.1}$	$\mathbf{91.4}_{0.2}$
$\text{BERT}_{\text{LARGE}}$	$75.6_{3.3}$	$36.6_{0.4}$	$79.3_{0.7}$
+ TA	$87.3_{0.3}$	$41.7_{1.1}$	$90.0_{0.4}$
+ ST	$90.6_{0.3}$	$43.8_{0.4}$	$89.0_{1.1}$
+ STraTA	$\mathbf{92.4}_{0.1}$	$\mathbf{45.5}_{0.7}$	$\mathbf{90.6}_{0.0}$
Du et al. (2021) (20 examples per class)			
$\text{RoBERTa}_{\text{LARGE}}$	$83.6_{2.7}$	$42.3_{1.6}$	$88.9_{1.7}$
+ SentAugST	$\mathbf{86.7}_{2.3}$	$\mathbf{44.4}_{1.0}$	$\mathbf{89.7}_{2.0}$

Table 3: Compared to Du et al. (2021), our approach leads to better downstream performance, despite using a weaker base model (BERT vs. RoBERTa) and with less labeled examples.

Using a better base model leads to better self-training results: Our experiment results show that self-training is complementary to different BERT base models across target tasks—the better the BERT base model, the better self-training results. BERT + **TA** yields better self-training results than BERT + $\textbf{ITFT}_{\text{MNLI}}$, and both are better than the vanilla BERT. Combinations of $\text{BERT}_{\text{LARGE}}$ and **ST** typically outperform that of $\text{BERT}_{\text{BASE}}$ and **ST**. Interestingly, $\text{BERT}_{\text{LARGE}}$+ **ST** is competitive with $\text{BERT}_{\text{LARGE}}$+ **STraTA** on several tasks (e.g., QQP and QNLI), and this does not hold for $\text{BERT}_{\text{BASE}}$.

Comparison to recent published work: The last three rows of Table 2 and the last two rows of Table 3 show results from recent published work.[16] Broadly, our methods lead to better performance compared to these approaches. However, due to differences in evaluation methodology (e.g., models, training/development data subsets, number of random restarts, and other factors), we refrain from explicitly ranking the approaches.

5 Analysis of few-shot learning results

Having established the effectiveness of both task augmentation and self-training in the few-shot setting, we conduct a series of analysis experiments in this section to explore the source of the observed improvements.

[16]While Wang et al. (2021) report results for LM-BFF and EFL across 5 random data subsets using a fixed set of seeds, Du et al. (2021) tried 10 seeds for each of their 5 random data subsets and report the mean of the top 3 seeds. To be more comparable to (Du et al., 2021), we report the mean of our top 3 random seeds in Table 3.

Model	SST-2	SciTail
$\text{RAND}_{\text{BASE}}$	$50.0_{1.6}$	$50.7_{2.4}$
+ STraTA	$78.6_{0.9}$	$64.4_{3.1}$
$\text{BERT}_{\text{BASE}}$	$59.1_{8.4}$	$67.1_{6.6}$
+ STraTA	$90.1_{0.8}$	$86.3_{3.5}$
$\text{BERT}_{\text{LARGE}}$	$66.1_{8.7}$	$68.8_{9.5}$
+ STraTA	$91.7_{0.7}$	$87.3_{2.9}$

Table 4: Our approach yields improvements even when starting with a randomly-initialized model, but pre-training helps considerably.

Sample efficiency with task augmentation and self-training: Figure 1 illustrates how our **STraTA** approach improves sample efficiency as the number of examples per class increases. For the SST-2 sentiment dataset, despite using only $K = 8$ training examples per class, **STraTA** has already nearly saturated its performance, achieving results competitive with standard fine-tuning over the whole dataset of 67K labeled examples. On the harder task of SciTail, **STraTA** continues to improve as K increases, and surpasses the performance of standard fine-tuning with the whole dataset of 27K labeled examples at $K = 512$.

STraTA improves a randomly-initialized base model: Table 4 shows that our **STraTA** approach does not require a powerful pre-trained base model to exhibit improvements: when applied to a randomly initialized Transfomer model ($\text{RAND}_{\text{BASE}}$) with the same architecture as $\text{BERT}_{\text{BASE}}$, $\text{RAND}_{\text{BASE}}$+ **STraTA** outperforms the vanilla $\text{BERT}_{\text{BASE}}$ by a large margin on SST-2, while being competitive on SciTail. Additionally, $\text{BERT}_{\text{BASE}}$+ **STraTA** substantially outperforms the vanilla $\text{BERT}_{\text{LARGE}}$ by 24% and 17.5% on SST-2 and SciTail, respectively.

Self-training on a broad distribution of pseudo-labeled data: Previous self-training algorithms (Rosenberg et al., 2005; McClosky et al., 2006; Sohn et al., 2020; Du et al., 2021) typically add a small set of unlabeled examples with the highest-confidence pseudo labels to the labeled data set $\mathcal{L}$ at each iteration. In contrast, our approach adds *all* pseudo-labeled examples to $\mathcal{L}$ at every iteration regardless of confidence. We compare the two approaches in

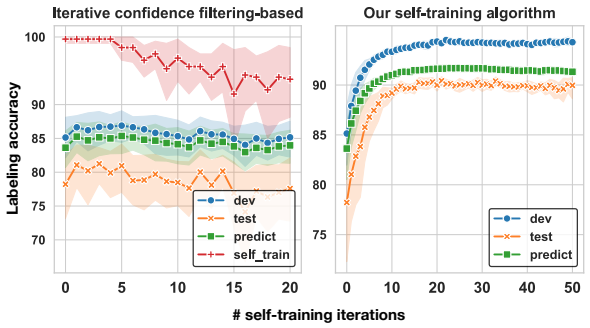

Figure 3: On the SST-2 sentiment dataset, traditional confidence filtering-based self-training (left) yields poor results compared to our approach, which trains on all pseudo-labels at each iteration (right).

Figure 3, which shows the labeling accuracy (% of unlabeled examples that are labeled correctly) on the development set (**dev**), the test set (**test**), and the unlabeled data pool (**predict**) of the SST-2 sentiment dataset. In the iterative confidence filtering-based approach (left plot), a fixed number (in this plot, 32) of most confidently labeled examples are added to the labeled set $\mathcal{L}$ at each iteration (the **self-train** line shows the labeling accuracy of these examples); once they have been added, they are not removed, and this process is repeated until the unlabeled set $\mathcal{U}$ is exhausted. As can be seen, this approach works well for the several first self-training iterations (3-5), but then labeling accuracy begins to degrade. In contrast, our algorithm (right plot) gradually and consistently improves labeling accuracy before converging at some iteration. These results suggest that strong base models benefit from including even significantly noisy pseudo-labels in self-training, as opposed to training on a narrow distribution of high-confidence predictions.

Does self-training work with out-of-domain/distribution (OOD) unlabeled examples? We investigate this question by applying self-training on top of $\text{BERT}_{\text{BASE}}$+ **TA**. We consider SOURCE → TARGET task pairs where training data from the source task without ground-truth labels is used as OOD unlabeled data for the target task. We experiment with several task pairs, including MNLI → SciTail, SST-2 → CR, QQP → MRPC, and MNLI → RTE. As shown in Table 5, self-training with OOD unlabeled examples ($\textbf{ST}_{\text{OUT}}$) is also helpful, offering an average absolute performance gain of +3.5% over the strong $\text{BERT}_{\text{BASE}}$+ **TA** baseline. However, using OOD unlabeled examples typically leads to worse self-training results compared to using in-domain unlabeled examples ($\textbf{ST}_{\text{IN}}$), except for the case MNLI → RTE, and combining the two types of unlabeled examples ($\textbf{ST}_{\text{IN+OUT}}$) does not bring further improvements over $\textbf{ST}_{\text{IN}}$.

Model	SciTail	CR	MRPC	RTE
$\text{BERT}_{\text{BASE}}$	$67.1_{6.6}$	$65.2_{8.2}$	$72.4_{10.2}$	$51.4_{2.5}$
$\text{BERT}_{\text{BASE}}$+ TA	$78.5_{3.2}$	$86.5_{2.2}$	$74.5_{6.5}$	$67.6_{7.1}$
+ ST_{IN}	$86.3_{3.5}$	$90.5_{0.8}$	$81.0_{0.8}$	$70.6_{2.4}$
+ ST_{OUT}	$81.4_{3.7}$	$88.3_{1.9}$	$80.3_{1.9}$	$71.2_{3.2}$
+ $\text{ST}_{\text{IN+OUT}}$	$82.6_{2.6}$	$88.3_{1.5}$	$80.2_{1.1}$	$69.9_{4.0}$

Table 5: Self-training with out-of-domain unlabeled examples also results in improvements, but using in-domain data works significantly better.

Model	SST-2	SciTail
$\text{BERT}_{\text{BASE}}$	$58.8_{8.4}$ (↓ 0.3)	$61.5_{5.4}$ (↓ 5.6)
+ LMFT	$64.0_{8.1}$ (↓ 0.9)	$59.3_{5.6}$ (↓ 4.7)
+ $\text{ITFT}_{\text{MNLI}}$	$76.5_{7.2}$ (↓ 0.3)	$76.2_{5.4}$ (↑ 0.4)
+ TA	$79.8_{6.3}$ (↓ 0.5)	$77.8_{3.3}$ (↓ 0.7)
+ STraTA	$86.6_{2.6}$ (↓ 3.5)	$80.6_{3.0}$ (↓ 5.7)

Table 6: In a realistic evaluation without a development set, our **STraTA** approach still leads to significant improvements on top of $\text{BERT}_{\text{BASE}}$. In parentheses, we show the absolute increase (↑) or decrease (↓) in performance compared to the same method used with a development set.

Towards realistic evaluation in few-shot learning: In real-world low-resource scenarios, it is often impractical to rely on a development set (Oliver et al., 2018; Kann et al., 2019). With so little data, it may be more effective to use all labeled data for training. To examine the applicability of our methods to this real-world setting, here we consider an evaluation that does not make use of a development set. Rather than using early stopping, we fine-tune each model for a fixed number of 512 steps. We checkpoint every 30 steps and evaluate a single model obtained by averaging the last 5 model checkpoints. For self-training, we perform a fixed number of 30 self-training iterations, each following the same fine-tuning procedure.

Table 6 summarizes our results. Broadly, all models perform worse in this setting than when a development set is available. Our **STraTA** approach still provides significant improvements over $\text{BERT}_{\text{BASE}}$, but much worse than the same method used with a development set. We conjecture that this is because without a development set, the

model achieves somewhat lower accuracy in each self-training iteration, and these errors compound through later iterations.

6 Related Work

Improving language model fine-tuning: Fine-tuning has been the most common approach for applying pre-trained language models to downstream tasks. However, it typically requires a target dataset of thousands to hundreds of thousands of examples to work well (Yogatama et al., 2019; Brown et al., 2020). Many methods have been proposed to improve performance and stability of pre-trained language models on small datasets, including language model fine-tuning on unlabeled data from the target domain (Howard and Ruder, 2018; Gururangan et al., 2020), intermediate-task fine-tuning (Phang et al., 2019), multi-task pre-finetuning (Aghajanyan et al., 2021a), better design choices and training strategies (Mosbach et al., 2021; Zhang et al., 2021), and regularization-oriented techniques (Jiang et al., 2020; Aghajanyan et al., 2021b). More related to our work is research on intermediate-task fine-tuning that makes use of data-rich tasks (Phang et al., 2019), tasks that require complex reasoning and inference (Pruksachatkun et al., 2020), and beneficial relationships among tasks (Vu et al., 2020).

Few-shot learning: Our work also relates to research in few-shot learning. In previous work, fine-tuning is combined with other learning strategies to improve few-shot performance, including consistency training (Xie et al., 2020a), meta-learning (Bansal et al., 2020), self-training (Du et al., 2021; Sun et al., 2020), and contrastive learning (Gunel et al., 2021). Other work has focused on prompt-based/entailment-based few-shot learning approaches (Brown et al., 2020; Schick and Schütze, 2021; Gao et al., 2021; Tam et al., 2021; Wang et al., 2021). Notably, Brown et al. (2020) demonstrate remarkable few-shot learning performance with a single *frozen* GPT-3 model, although its performance still lags far behind state-of-the-art fine-tuning results.

Generative data augmentation: Recent work explores the generation capabilities of large-scale generative language models, such as GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2020), to generate synthetic training data for different tasks, including text classification (Anaby-Tavor et al., 2020; Lee et al., 2021; Schick and Schütze, 2021), question answering (Puri et al., 2020), and commonsense reasoning (Yang et al., 2020). Yang et al. (2020) show that such a generative approach consistently outperforms previous data augmentation methods based on back-translation (Sennrich et al., 2016; Xie et al., 2020a).

Semi-supervised learning: Another area upon which our work builds is semi-supervised learning (SSL). Recent work has combined self-training with other techniques, e.g., noise injection (He et al., 2020; Xie et al., 2020b), consistency regularization and pseudo-labeling (Sohn et al., 2020), to develop powerful SSL algorithms. Du et al. (2021) show that self-training improves upon language model pre-training.

7 Limitations & Conclusion

Task augmentation and self-training provide complementary ways to leverage task-specific unlabeled data for improved downstream performance. While task augmentation utilizes unlabeled texts to synthesize a large amount of in-domain data for an auxiliary training task, self-training uses a model's predictions on unlabeled examples to improve the model itself. When combining these methods in STraTA, we are able to substantially improve sample efficiency across 12 NLP benchmark datasets. That said, each method has its own limitations. While our implementation uses NLI as an auxiliary task in task augmentation, there are target tasks for which NLI may not be helpful (e.g., on grammatical acceptability judgments, as shown in Wang et al. (2019a)). Additionally, other auxiliary tasks may increase improvements (e.g., QNLI benefits more from QA tasks (Vu et al., 2020)). We leave exploration of other auxiliary tasks to future work. Finally, our self-training algorithm (like prior approaches) assumes access to task-specific unlabeled examples, which might be non-trivial to acquire for some applications.

Acknowledgments

We thank David Berthelot, Colin Raffel, Kalpesh Krishna, Zi Yang, Jenny Lee, Guolong Su, and Nan Hua for useful discussions and valuable feedback at different stages of this project. We would also like to thank the anonymous reviewers, Kenton Lee, Zihang Dai, Ed H. Chi, Nader Akoury, Brendan O'Connor, Zhiyang Xu, Andrew Drozdov, and the rest of the UMass NLP group for their thoughtful comments and suggestions.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021a. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*.

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021b. Better fine-tuning by reducing representational collapse. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! *Proceedings of the 33th AAAI Conference on Artificial Intelligence (AAAI 2019)*, 34(05):7383–7390.

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

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 632–642.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pages 1877–1901.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, pages 1–14.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 670–680.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the 1st International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment (MLCW 2005)*, page 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2019)*, pages 4171–4186.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the 3rd International Workshop on Paraphrasing (IWP 2005)*.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2021)*, pages 5408–5418.

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

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2021. Supervised contrastive learning for pre-trained language model fine-tuning. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (PMLR 2017)*, volume 70, pages 1321–1330.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 8342–8360.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 328–339.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, page 168–177.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First Quora Dataset Release: Question pairs.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 2177–2190.

Zhengbao Jiang, Haibo Ding, Jun Araki, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics (TACL 2021)*.

Katharina Kann, Kyunghyun Cho, and Samuel R. Bowman. 2019. Towards realistic practices in low-resource natural language processing: The development set. 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 2019)*, pages 3342–3349.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation. *arXiv preprint arXiv:2102.01335*.

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*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 216–223.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2006)*, pages 152–159.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In *Proceedings of the 33th Conference on Neural Information Processing Systems (NeurIPS 2019)*, volume 32. Curran Associates, Inc.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. 2018. Realistic evaluation of deep semi-supervised learning algorithms. In *Proceedings of the 32th Conference on Neural Information Processing Systems (NeurIPS 2018)*, volume 31, page 3239–3250.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2019. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 5231–5247.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical*

Methods in Natural Language Processing (EMNLP 2020), pages 5811–5826.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2383–2392.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. In *Proceedings of the 7th IEEE Workshops on Application of Computer Vision (WACV-MOTION 2005)*, volume 1, pages 29–36.

Timo Schick and Hinrich Schütze. 2021. 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 (NAACL 2021)*, pages 2339–2352.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. *arXiv preprint arXiv:2104.07540*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 86–96.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

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 (EMNLP 2013)*, pages 1631–1642.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pages 596–608.

Zijun Sun, Chun Fan, Xiaofei Sun, Yuxian Meng, Fei Wu, and Jiwei Li. 2020. Neural semi-supervised learning for text classification under large-scale pretraining. *arXiv preprint arXiv:2011.08626*.

Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. *arXiv preprint arXiv:2103.11955*.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 7882–7926.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, Berlin Chen, Benjamin Van Durme, Edouard Grave, Ellie Pavlick, and Samuel R. Bowman. 2019a. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 4465–4476.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019b. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. *arXiv preprint arXiv:2104.14690*.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*, pages 6382–6388.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2018)*, pages 1112–1122.

Thomas Wolf, L Debut, V Sanh, J Chaumond, C Delangue, A Moi, P Cistac, T Rault, R Louf, M Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020a. Unsupervised data augmentation for consistency training. In *Proceedings of*

the 34th Conference on Neural Information Processing Systems (NeurIPS 2020, volume 33, pages 6256–6268.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. 2020b. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, pages 10687–10698.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020 (Findings of EMNLP 2020)*, pages 1008–1025.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the 33th Conference on Neural Information Processing Systems (NeurIPS 2019)*, volume 32, pages 5753–5763.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2021. Revisiting few-sample bert fine-tuning. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Appendices

A Additional details for the datasets used in our study

The datasets used in our experiments come from two common language understanding benchmarks: GLUE (Wang et al., 2019b) and SentEval (Conneau and Kiela, 2018). See Table 7 for details about dataset characteristics. We report F1 scores for QQP and MRPC, Spearman correlations for STS-B and SICK-R, and accuracy scores for the other tasks.

B Example outputs of our NLI data generator

Table 8 shows example outputs of our NLI data generator using unlabeled texts from SST-2 and SciTail.

C Full experiment results across data regimes

We report the full results for our experiments in the FULL, LIMITED, and FEW-SHOT data regimes in Table 9, Table 10, and Table 11, respectively.

Task	| Train |	Task type	Domain
text classification/regression (CR)			
SNLI (Bowman et al., 2015)	570K	NLI	misc.
MNLI (Williams et al., 2018)	393K	NLI	misc.
QQP (Iyer et al., 2017)	364K	paraphrase identification	social QA
QNLI (Wang et al., 2019b)	105K	QA-NLI	Wikipedia
SST-2 (Socher et al., 2013)	67K	sentiment analysis	movie reviews
SciTail (Khot et al., 2018)	27K	NLI	science QA
SST-5 (Socher et al., 2013)	8.5K	sentiment analysis	movie reviews
STS-B (Cer et al., 2017)	7K	semantic similarity	misc.
SICK-E (Marelli et al., 2014)	4.5K	NLI	misc.
SICK-R (Marelli et al., 2014)	4.5K	semantic similarity	misc.
CR (Hu and Liu, 2004)	4K	sentiment analysis	product reviews
MRPC (Dolan and Brockett, 2005)	3.7K	paraphrase identification	news
RTE (Dagan et al., 2005, et seq.)	2.5K	NLI	news, Wikipedia

Table 7: Datasets used in our experiments and their characteristics, sorted by training data set size.

unlabeled texts from SST-2
Input: a cast of a-list brit actors **+ entailment** → **Output:** the cast includes british actors. **+ neutral** → **Output:** the cast includes a-list brit actors such as nicole kidman and charlie chaplin. **+ contradiction** → **Output:** the cast was all new and unknown.
Input: both heartbreaking and heartwarming ... just a simple fable done in an artless sytle , but it 's tremendously moving **+ entailment** → **Output:** the story is moving and touching. **+ neutral** → **Output:** the artless style used to make the fable was a favorite of many. **+ contradiction** → **Output:** the fable is artful and sophisticated but it is also heartbreaking.
Input: gaping plot holes sink this ' sub ' - standard thriller and drag audience enthusiasm to crush depth **+ entailment** → **Output:** this movie's plot holes seriously affect the enjoyment of the movie. **+ neutral** → **Output:** the movies pace is too slow for most audiences. **+ contradiction** → **Output:** the movie is a fun, easy romp of a thriller.
Input: putting the primitive murderer inside a high-tech space station unleashes a pandora 's box of special effects that run the gamut from cheesy to cheesier to cheesiest **+ entailment** → **Output:** the special effects are all over the place. **+ entailment** → **Output:** putting a murderer in a space station leads to all sorts of cheesy special effects. **+ entailment** → **Output:** the murderer was inside a space station.
Input: vera has created a provocative , absorbing drama that reveals the curse of a self-hatred instilled by rigid social mores . **+ neutral** → **Output:** vera has created a provocative drama that is absorbing and has a great cast. **+ neutral** → **Output:** vera has created a provocative drama that shows the curse of self-hatred and shows the bonds between women. **+ neutral** → **Output:** vera's drama, which has been loved by millions, is about the curse of self-hatred.
Input: filmmaking from one of french cinema 's master craftsmen **+ contradiction** → **Output:** the filmmaking is from a newcomer to the french cinema. **+ contradiction** → **Output:** the filmmaking is from a new blood in the american cinema. **+ contradiction** → **Output:** the filmmaking of one of italian cinema's master craftsmen
unlabeled texts from SciTail
Input: Surface tension is responsible for the fact that small insects can walk on water. **+ entailment** → **Output:** small insects can walk on water due to surface tension. **+ neutral** → **Output:** it's difficult for large insects to walk on water. **+ contradiction** → **Output:** small insects cannot walk on water because of surface tension.
Input: Tamiflu inhibits spread of virus. **+ entailment** → **Output:** the virus is less spread when tamiflu is used. **+ neutral** → **Output:** tamiflu is a good way to treat hiv. **+ contradiction** → **Output:** tamiflu promotes viral spread.
Input: Animals can transport the waste material known as urea in the circulatory system and store it safely at high concentrations. **+ entailment** → **Output:** animals can transport waste material in their circulatory systems. **+ neutral** → **Output:** animals can store urea in their bloodstreams for up to a year. **+ contradiction** → **Output:** only plants can transport the waste material known as urea in their circulatory systems.
Input: A number of applications of biomass, wind, hydropower and solar thermal are presently cost competitive with fossil fuels. **+ entailment** → **Output:** many biomass applications are now cost competitive with fossil fuels. **+ entailment** → **Output:** many alternatives to fossil fuels are cost competitive. **+ entailment** → **Output:** some biofuels are now cost competitive with fossil fuels.
Input: A cell wall is not present in animal cells. **+ neutral** → **Output:** in contrast, plant cells have a cell wall. **+ neutral** → **Output:** in addition to not having a cell wall, animal cells also lack mitochondria. **+ neutral** → **Output:** in animal cells, there is no cell wall, said the biologist.
Input: A bathymetric map can show the features of the bottom of a body of water. **+ contradiction** → **Output:** a bathymetric map shows the top of a body of water. **+ contradiction** → **Output:** a bathymetric map shows the features of the sky. **+ contradiction** → **Output:** a bathymetric map shows what is on the surface of the water.

Table 8: Example outputs of our NLI data generator using unlabeled texts from SST-2 and SciTail.

Model	SNLI	QQP	QNLI	SST-2	SciTail	SST-5	STS-B	SICK-E	SICK-R	CR	MRPC	RTE
$\text{BERT}_{\text{BASE}}$	90.3	87.8	90.6	91.7	93.2	$52.7_{0.8}$	$88.9_{0.3}$	$86.7_{0.5}$	$82.9_{0.5}$	$91.0_{0.9}$	$87.9_{1.0}$	$63.5_{2.3}$
+ LMFT	90.8	87.8	90.2	91.3	92.9	$52.8_{0.9}$	$89.3_{0.3}$	$86.8_{0.8}$	$82.7_{0.5}$	$90.5_{1.0}$	$87.9_{0.6}$	$63.9_{3.7}$
+ $\text{ITFT}_{\text{MNLI}}$	91.0	87.7	90.3	93.0	95.8	$53.8_{0.8}$	$90.1_{0.1}$	$89.5_{0.3}$	$85.3_{0.6}$	$91.7_{0.7}$	$89.8_{1.1}$	$78.1_{1.9}$
+ TA	91.2	88.1	90.9	93.9	96.3	$54.3_{0.9}$	$90.1_{0.1}$	$90.1_{0.3}$	$85.6_{0.3}$	$92.2_{0.5}$	$90.1_{0.8}$	$79.3_{0.9}$
$\text{BERT}_{\text{LARGE}}$	91.1	88.4	91.9	92.4	95.3	$53.7_{0.9}$	$89.6_{0.2}$	$87.9_{0.6}$	$84.4_{0.4}$	$91.7_{0.6}$	$89.0_{0.8}$	$68.6_{7.2}$
+ LMFT	91.0	88.1	90.4	93.5	95.3	$54.0_{0.4}$	$89.5_{0.2}$	$87.7_{0.5}$	$84.0_{0.5}$	$91.6_{0.8}$	$89.5_{1.0}$	$66.5_{7.3}$
+ $\text{ITFT}_{\text{MNLI}}$	91.1	88.2	91.6	93.5	96.5	$54.0_{0.8}$	$90.3_{0.3}$	$89.9_{0.2}$	$86.3_{0.3}$	$92.0_{0.6}$	$89.7_{0.9}$	$82.3_{1.4}$
+ TA	91.9	88.5	92.5	94.7	96.9	$55.7_{0.8}$	$90.9_{0.2}$	$90.7_{0.3}$	$87.0_{0.3}$	$93.3_{0.6}$	$90.8_{0.7}$	$83.8_{1.1}$

Table 9: Our experiment results in the FULL data regime.

Model	SNLI	QQP	QNLI	SST-2	SciTail	SST-5	STS-B	SICK-E	SICK-R	CR	MRPC	RTE
$\text{BERT}_{\text{BASE}}$	$71.7_{1.1}$	$71.7_{0.4}$	$78.7_{0.7}$	$87.4_{1.0}$	$88.3_{1.2}$	$47.1_{1.3}$	$86.8_{0.6}$	$81.5_{0.6}$	$76.7_{0.8}$	$89.9_{0.7}$	$83.9_{1.1}$	$61.7_{1.3}$
+ LMFT	$73.4_{2.1}$	$72.1_{0.6}$	$76.3_{3.1}$	$86.9_{1.2}$	$88.4_{1.4}$	$47.5_{1.3}$	$87.2_{0.7}$	$81.1_{0.6}$	$75.9_{0.7}$	$91.1_{0.8}$	$84.4_{0.6}$	$63.2_{2.3}$
+ $\text{ITFT}_{\text{MNLI}}$	$82.9_{0.3}$	$73.3_{0.7}$	$81.6_{0.9}$	$87.8_{0.6}$	$90.3_{1.1}$	$48.8_{1.0}$	$88.5_{0.3}$	$87.6_{0.5}$	$81.7_{0.8}$	$90.0_{0.6}$	$87.0_{0.9}$	$78.0_{1.3}$
+ TA	$85.7_{0.3}$	$75.3_{0.4}$	$82.5_{0.8}$	$90.4_{0.7}$	$90.7_{0.8}$	$49.2_{1.3}$	$88.5_{0.3}$	$88.5_{0.5}$	$82.4_{0.6}$	$91.4_{1.0}$	$87.3_{0.7}$	$78.7_{1.2}$
$\text{BERT}_{\text{LARGE}}$	$77.4_{0.6}$	$74.1_{1.0}$	$81.7_{0.9}$	$89.8_{0.6}$	$90.9_{0.7}$	$49.1_{1.3}$	$88.2_{0.4}$	$84.8_{0.7}$	$80.2_{0.4}$	$91.2_{0.6}$	$85.7_{1.7}$	$66.8_{2.7}$
+ LMFT	$75.8_{1.5}$	$71.6_{0.5}$	$80.5_{2.0}$	$88.9_{0.8}$	$87.7_{2.3}$	$49.2_{3.1}$	$88.4_{0.4}$	$83.2_{0.6}$	$78.5_{0.6}$	$90.9_{0.7}$	$84.9_{1.1}$	$65.2_{3.4}$
+ $\text{ITFT}_{\text{MNLI}}$	$85.2_{0.4}$	$74.0_{0.5}$	$83.5_{0.5}$	$90.0_{0.8}$	$92.1_{1.1}$	$49.4_{1.2}$	$87.8_{0.8}$	$88.8_{0.5}$	$83.2_{0.7}$	$91.3_{0.7}$	$86.4_{0.9}$	$81.1_{1.3}$
+ TA	$87.3_{0.3}$	$75.7_{0.5}$	$85.0_{0.5}$	$91.7_{0.7}$	$92.3_{1.1}$	$51.4_{1.0}$	$89.0_{0.6}$	$89.4_{0.4}$	$84.3_{0.4}$	$92.6_{0.6}$	$88.0_{0.8}$	$82.9_{1.8}$

Table 10: Our experiment results in the LIMITED data regime.

Model	SNLI	QQP	QNLI	SST-2	SciTail	SST-5	STS-B	SICK-E	SICK-R	CR	MRPC	RTE
$\text{BERT}_{\text{BASE}}$	$43.7_{2.2}$	$55.9_{6.5}$	$59.0_{10.9}$	$59.1_{8.4}$	$67.1_{6.6}$	$30.5_{2.0}$	$73.6_{4.5}$	$61.3_{4.1}$	$59.7_{2.7}$	$65.2_{8.2}$	$72.4_{10.2}$	$51.4_{2.5}$
+ LMFT	$45.2_{3.9}$	$57.2_{6.2}$	$57.6_{9.1}$	$64.9_{8.7}$	$64.0_{8.0}$	$33.4_{1.9}$	$75.4_{4.4}$	$59.3_{4.0}$	$58.3_{2.0}$	$72.4_{6.0}$	$73.9_{8.6}$	$50.9_{3.9}$
+ $\text{ITFT}_{\text{MNLI}}$	$75.2_{5.7}$	$63.7_{7.0}$	$62.8_{5.1}$	$76.8_{7.2}$	$75.8_{5.6}$	$35.0_{2.6}$	$80.2_{1.1}$	$80.4_{1.9}$	$73.5_{2.7}$	$79.2_{3.6}$	$74.3_{8.0}$	$62.2_{13.5}$
+ TA	$83.3_{0.8}$	$68.7_{1.5}$	$70.1_{3.4}$	$80.3_{6.6}$	$78.5_{3.2}$	$37.4_{3.0}$	$80.7_{1.5}$	$81.1_{2.4}$	$75.9_{1.8}$	$86.5_{2.2}$	$74.5_{6.5}$	$67.6_{7.1}$
+ ST	$65.0_{5.8}$	$69.9_{5.9}$	$71.6_{11.3}$	$62.7_{10.4}$	$68.6_{8.3}$	$33.9_{3.5}$	$80.5_{2.2}$	$68.1_{4.5}$	$64.0_{2.4}$	$78.2_{6.3}$	$80.5_{1.8}$	$50.7_{3.1}$
+ $\text{ITFT}_{\text{MNLI}}$ + ST	$83.2_{0.3}$	$70.7_{5.9}$	$81.5_{1.2}$	$88.0_{2.1}$	$83.7_{4.4}$	$39.5_{2.0}$	$84.2_{0.8}$	$81.8_{2.6}$	$75.8_{2.2}$	$85.6_{2.3}$	$80.6_{1.2}$	$62.5_{12.0}$
+ STraTA	$85.7_{0.2}$	$74.5_{0.4}$	$82.1_{0.5}$	$90.1_{0.8}$	$86.3_{3.5}$	$41.3_{1.5}$	$84.7_{0.5}$	$84.9_{1.2}$	$77.6_{1.6}$	$90.5_{0.8}$	$81.0_{0.8}$	$70.6_{2.4}$
$\text{BERT}_{\text{LARGE}}$	$43.1_{4.4}$	$58.5_{4.7}$	$64.4_{6.1}$	$66.1_{8.7}$	$68.8_{9.5}$	$35.2_{1.3}$	$74.6_{3.8}$	$66.5_{4.5}$	$66.6_{3.3}$	$72.0_{6.0}$	$79.9_{2.0}$	$53.1_{3.3}$
+ LMFT	$39.6_{2.6}$	$52.7_{4.7}$	$52.2_{1.6}$	$66.3_{9.3}$	$66.4_{10.6}$	$36.8_{2.9}$	$75.4_{9.4}$	$58.8_{6.9}$	$51.6_{7.0}$	$75.6_{5.9}$	$80.5_{2.4}$	$52.8_{4.8}$
+ $\text{ITFT}_{\text{MNLI}}$	$79.9_{3.1}$	$62.6_{9.0}$	$64.5_{4.4}$	$80.7_{5.0}$	$72.3_{11.2}$	$36.4_{2.1}$	$75.5_{4.0}$	$77.8_{3.8}$	$73.5_{2.8}$	$82.6_{3.0}$	$72.8_{7.9}$	$69.7_{14.6}$
+ TA	$84.8_{0.7}$	$64.6_{6.3}$	$71.5_{4.0}$	$85.5_{1.4}$	$79.0_{4.5}$	$38.5_{3.0}$	$78.9_{2.4}$	$81.2_{3.9}$	$77.5_{1.4}$	$88.6_{1.3}$	$78.2_{6.6}$	$77.0_{6.3}$
+ ST	$69.3_{9.2}$	$74.3_{1.2}$	$85.4_{1.7}$	$81.9_{12.2}$	$79.9_{4.8}$	$42.0_{1.5}$	$82.8_{2.3}$	$77.3_{3.1}$	$73.1_{2.3}$	$88.1_{1.3}$	$81.2_{0.5}$	$53.9_{4.3}$
+ $\text{ITFT}_{\text{MNLI}}$ + ST	$85.4_{0.3}$	$74.8_{0.7}$	$86.1_{1.1}$	$89.7_{0.7}$	$86.2_{4.2}$	$42.2_{2.0}$	$84.1_{1.7}$	$84.3_{2.0}$	$78.4_{1.3}$	$89.3_{1.0}$	$81.4_{1.2}$	$72.7_{5.4}$
+ STraTA	$87.3_{0.3}$	$75.1_{0.2}$	$86.4_{0.8}$	$91.7_{0.7}$	$87.3_{2.9}$	$43.0_{2.3}$	$84.5_{1.6}$	$86.3_{1.8}$	$79.0_{1.0}$	$90.0_{0.6}$	$81.5_{0.7}$	$77.1_{5.4}$

Table 11: Our experiment results in the FEW-SHOT data regime.