Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks

Jason Phang[1,*]
jasonphang@nyu.edu

Thibault Févry[1,*]
thibault.fevry@nyu.edu

Samuel R. Bowman[1,2,3]
bowman@nyu.edu

[1]Center for Data Science
New York University
60 Fifth Avenue
New York, NY 10011

[2]Dept. of Linguistics
New York University
10 Washington Place
New York, NY 10003

[3]Dept. of Computer Science
New York University
60 Fifth Avenue
New York, NY 10011

Abstract

Pretraining sentence encoders with language modeling and related unsupervised tasks has recently been shown to be very effective for language understanding tasks. By supplementing language model-style pretraining with further training on data-rich supervised tasks, such as natural language inference, we obtain additional performance improvements on the GLUE benchmark. Applying supplementary training on BERT (Devlin et al., 2018), we attain a GLUE score of 81.8—the state of the art[1] and a 1.4 point improvement over BERT. We also observe reduced variance across random restarts in this setting. Our approach yields similar improvements when applied to ELMo (Peters et al., 2018a) and Radford et al. (2018)'s model. In addition, the benefits of supplementary training are particularly pronounced in data-constrained regimes, as we show in experiments with artificially limited training data.

1 Introduction

Recent work has shown mounting evidence that pretraining sentence encoder neural networks on unsupervised tasks like language modeling, and then fine-tuning them on individual target tasks, can yield significantly better target task performance than could be achieved using target task training data alone (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2018). Large-scale unsupervised pretraining in works like these seems to produce sentence encoders with substantial knowledge of the target language (which, so far, is generally English). These works have shown that the one-size-fits-all approach of fine-tuning a large pretrained model with a thin output layer for a given task can achieve results as good or better than carefully-designed task-specific models without such pretraining.

However, it is not obvious that the model parameters obtained during unsupervised pretraining should be *ideally suited* to supporting this kind of transfer learning. Especially when only a small amount of training data is available for the target task, fine-tuning experiments are potentially brittle, and rely on the pretrained encoder parameters to be reasonably close to an ideal setting for the target task. During target task training, the encoder must learn and adapt enough to be able to solve the target task—potentially involving a very different input distribution and output label space than was seen in pretraining—but it must also avoid overfitting or catastrophic forgetting of what was learned during pretraining.

This work explores the possibility that the use of a second stage of pretraining with data-rich intermediate supervised tasks might mitigate this brittleness, improving both the robustness and effectiveness of the resulting target task model. We name this approach, which is meant to be combined with existing approaches to pretraining, *Supplementary Training on Intermediate Labeled-data Tasks* (STILTs).

Experiments with sentence encoders on STILTs take the following form: (i) A model is first trained on an unlabeled-data task like language modeling that can teach it to reason about the target language; (ii) The model is then further trained on an intermediate, labeled-data task for which ample data is available; (iii) The model is finally fine-tuned further on the target task and evaluated. Our experiments evaluate STILTs as a means of improving target task performance on the GLUE benchmark suite (Wang et al., 2018)—a collection of language understanding tasks drawn from the NLP literature.

We apply STILTs to three separate pretrained

[*] Equal contribution.

[1] As of 02/24/2019.

sentence encoders: BERT (Devlin et al., 2018), GPT (Radford et al., 2018), and a variant of ELMo (Peters et al., 2018a). We follow Radford et al. and Devlin et al. in our basic mechanism for fine-tuning both for the intermediate and final tasks, and use the following four intermediate tasks: (i) the Multi-Genre NLI Corpus (MNLI; Williams et al., 2018), (ii) the Stanford NLI Corpus (SNLI; Bowman et al., 2015), (iii) the Quora Question Pairs[2] (QQP) dataset, and (iv) a custom fake-sentence-detection task based on the BooksCorpus dataset (Zhu et al., 2015a) using a method adapted from Warstadt et al. (2018). The use of MNLI and SNLI is motivated by prior work on using natual language inference tasks to pretrain sentence encoders (Conneau et al., 2017; Subramanian et al., 2018; Bowman et al., 2019). QQP has a similar format and dataset scale, while requiring a different notion of sentence similarity. The fake-sentence-detection task is motivated by Warstadt et al.'s analysis on CoLA and linguistic acceptability, and adapted for our experiments. These four tasks are a sample of data-rich supervised tasks that we can use to demonstrate the benefits of STILTs, but they do not represent an exhaustive exploration of the space of promising intermediate tasks.

We show that using STILTs yields significant gains across most of the GLUE tasks, across all three sentence encoders we used, and claims the state of the art on the overall GLUE benchmark. In addition, for the 24-layer version of BERT, which can require multiple random restarts for good performance on target tasks with limited training data, we find that STILTs substantially reduces the number of runs with degenerate results across random restarts. For instance, using STILTs with 5k training examples, we reduce the number of degenerate runs from five to one on SST and from two to none on STS.

As we expect that any kind of pretraining will be most valuable in a limited training data regime, we also conduct a set of experiments where a model is fine-tuned on only 1k- or 5k-example subsamples of the target task training set. The results show that STILTs substantially improves model performance across most tasks in this downsampled data setting, even more so than in the full-data setting.

2 Related Work

In the area of pretraining for sentence encoders, Zhang and Bowman (2018) compare several pretraining tasks for syntactic target tasks, and find that language model pretraining reliably performs well. Peters et al. (2018b) investigate the architectural choices behind ELMo-style pretraining with a fixed encoder, and find that the precise choice of encoder architecture strongly influences training speed, but has a relatively small impact on performance. Bowman et al. (2019) compare a variety of tasks for pretraining in an ELMo-style setting with no encoder fine-tuning. They conclude that language modeling generally works best among candidate single tasks for pretraining, but show some cases in which a cascade of a model pretrained on language modeling followed by another model pretrained on tasks like MNLI can work well. The paper introducing BERT (Devlin et al., 2018) briefly mentions encouraging results in a direction similar to ours: One footnote notes that unpublished experiments show "substantial improvements on RTE from multitask training with MNLI."

Most prior work uses features from frozen, pretrained sentence encoders in downstream tasks. A more recent trend of fine-tuning the whole model for the target task from a pretrained state (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2018) has led to state-of-the-art results on several benchmarks. For that reason, we focus our analysis on the paradigm of fine-tuning the whole model for each task.

In the area of sentence-to-vector encoding, Conneau et al. (2018) offer one of the most comprehensive suites of diagnostic tasks, and highlight the importance of ensuring that these models preserve lexical content information.

In earlier work less closely tied to the unsupervised pretraining setup studied here, Bingel and Søgaard (2017) and Kerinec et al. (2018) investigate the conditions under which task combinations can be productively combined in multitask learning. They show that multitask learning is more likely to work when the target task quickly plateaus and the auxiliary task keeps improving. They also report that gains are lowest when the Jensen-Shannon Divergence between the unigram distributions of tasks is highest, i.e when auxiliary and target tasks have different vocabulary.

In word representations, this work shares moti-

[2] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

vations with work on embedding space *retrofitting* (Faruqui et al., 2015) wherein a labeled dataset like WordNet is used to refine representations learned by an unsupervised embedding learning algorithm before those representations are used for a target task.

3 Methods

Pretrained Sentence Encoders We primarily study the impact of STILTs on three sentence encoders: BERT (Devlin et al., 2018), GPT (Radford et al., 2018) and ELMo (Peters et al., 2018a). These models are distributed with pretrained weights from their respective authors, and are the best performing sentence encoders as measured by GLUE benchmark performance at time of writing. All three models are pretrained with large amounts of unlabeled text. ELMo uses a BiLSTM architecture whereas BERT and GPT use the Transformer architecture (Vaswani et al., 2017). These models are also trained with different objectives and corpora. BERT is a bidirectional Transformer trained on BooksCorpus (Zhu et al., 2015b) and English Wikipedia, with a masked-language model and next sentence prediction objective. GPT is uni-directional masked Transformer trained only on BooksCorpus with a standard language modeling objective. ELMo is trained on the 1B Word Benchmark (Chelba et al., 2013) with a standard language modeling objective.

For all three pretrained models, we follow BERT and GPT in using an inductive approach to transfer learning, in which the model parameters learned during pretraining are used to initialize a target task model, but are not fixed and do not constrain the solution learned for the target task. This stands in contrast to the approach originally used for ELMo (Peters et al., 2018b) and for earlier methods like McCann et al. (2017) and Subramanian et al. (2018), in which a sentence encoder component is pretrained and then attached to a target task model as a non-trainable input layer.

To implement intermediate-task and target-task training for GPT and ELMo, we use the public `jiant` transfer learning toolkit,[3] which is built on AllenNLP (Gardner et al., 2017) and PyTorch (Paszke et al., 2017). For BERT, we use the publicly available implementation of BERT released by Devlin et al. (2018), ported into PyTorch(Paszke et al., 2017) by HuggingFace[4].

Target Tasks and Evaluation We evaluate on the nine target tasks in the GLUE benchmark (Wang et al., 2018). These include MNLI, QQP, and seven others: acceptability classification with CoLA (Warstadt et al., 2018); binary sentiment classification with SST (Socher et al., 2013); semantic similarity with the MSR Paraphrase Corpus (MRPC; Dolan and Brockett, 2005) and STS-Benchmark (STS; Cer et al., 2017); and textual entailment with a subset of the RTE challenge corpora (Dagan et al., 2006, et seq.), and data from SQuAD (QNLI, Rajpurkar et al., 2016)[5] and the Winograd Schema Challenge (WNLI, Levesque et al., 2011) converted to entailment format as in White et al. (2017). Because of the adversarial nature of WNLI, our models do not generally perform better than chance, and we follow the recipe of Devlin et al. (2018) by predicting the most frequent label for all examples.

Most of our experiments—including all of our experiments using downsampled training sets for our target tasks—are evaluated on the *development set* of GLUE. Based on the results on the development set, we choose the best intermediate-task training scheme for each task and submit the best-per-task model for evaluation on the test set on the public leaderboard.

Intermediate Task Training Our experiments follow the standard pretrain-then-fine-tune approach, except that we add a supplementary training phase on an intermediate task before target-task fine-tuning. We call this approach *BERT on STILTs*, *GPT on STILTs* and *ELMo on STILTS* for the respective models. We evaluate a sample of four intermediate tasks, which were chosen to represent readily available data-rich sentence-level tasks similar to those in GLUE: (i) textual entailment with MNLI; (ii) textual entailment with SNLI; (iii) paraphrase detection with QQP; and (iv) a custom fake-sentence-detection task.

Our use of MNLI is motivated by prior successes with MNLI pretraining by Conneau et al. (2018) and Subramanian et al. (2018). We include the single-genre captions-based SNLI in ad-

[3] https://github.com/jsalt18-sentence-repl/jiant

[4] https://github.com/huggingface/pytorch-pretrained-BERT

[5] A newer version of QNLI was recently released by the maintainers of GLUE benchmark. All reported numbers in this work, including the aggregated GLUE score, reflect evaluation on the older version of QNLI (QNLIv1).

	Avg	**A.Ex**	**CoLA**	**SST**	**MRPC**	**QQP**	**STS**	**MNLI**	**QNLI**	**RTE**	**WNLI**
Training Set Size			*8.5k*	*67k*	*3.7k*	*364k*	*7k*	*393k*	*108k*	*2.5k*	*634*
Development Set Scores											
BERT	80.8	78.4	**62.1**	92.5	89.0/92.3	**91.5/88.5**	90.3/90.1	**86.2**	89.4	70.0	56.3
BERT→QQP	80.9	78.5	56.8	93.1	88.7/92.0	~~91.5/88.5~~	90.9/90.7	86.1	89.5	74.7	56.3
BERT→MNLI	82.4	80.5	59.8	**93.2**	**89.5/92.3**	91.4/88.4	**91.0/90.8**	~~86.2~~	**90.5**	**83.4**	56.3
BERT→SNLI	81.4	79.2	57.0	92.7	88.5/91.7	91.4/88.4	90.7/90.6	86.1	89.8	80.1	56.3
BERT→Real/Fake	77.4	74.3	52.4	92.1	82.8/88.5	90.8/87.5	88.7/88.6	84.5	88.0	59.6	56.3
BERT, Best of Each	**82.6**	**80.8**	**62.1**	**93.2**	**89.5/92.3**	**91.5/88.5**	**91.0/90.8**	**86.2**	**90.5**	**83.4**	56.3
GPT	75.4	72.4	**50.2**	**93.2**	80.1/85.9	89.4/85.9	86.4/86.5	**81.2**	82.4	58.1	56.3
GPT→QQP	76.0	73.1	48.3	93.1	83.1/88.0	~~89.4/85.9~~	87.0/86.9	80.7	82.6	62.8	56.3
GPT→MNLI	76.7	74.2	45.7	92.2	**87.3/90.8**	89.2/85.3	88.1/88.0	~~81.2~~	**82.6**	**67.9**	56.3
GPT→SNLI	76.0	73.1	41.5	91.9	86.0/89.9	89.9/86.6	**88.7/88.6**	81.1	82.2	65.7	56.3
GPT→Real/Fake	76.6	73.9	49.5	91.4	83.6/88.6	**90.1/86.9**	87.9/87.8	81.0	82.5	66.1	56.3
GPT, Best of Each	**77.5**	**75.9**	**50.2**	**93.2**	**87.3/90.8**	**90.1/86.9**	**88.7/88.6**	**81.2**	**82.6**	**67.9**	56.3
ELMo	63.8	59.4	15.6	84.9	69.9/80.6	86.4/82.2	64.5/64.4	69.4	73.0	50.9	56.3
ELMo→QQP	64.8	61.7	16.6	87.0	73.5/82.4	~~86.4/82.2~~	71.6/72.0	63.9	73.4	52.0	56.3
ELMo→MNLI	66.4	62.8	16.4	87.6	73.5/83.0	87.2/83.1	**75.2/75.8**	~~69.4~~	72.4	**56.3**	56.3
ELMo→SNLI	66.4	62.7	14.8	**88.4**	**74.0/82.5**	**87.3/83.1**	74.1/75.0	69.7	**74.0**	56.0	56.3
ELMo→Real/Fake	66.9	63.3	**27.3**	87.8	72.3/81.3	87.1/83.1	70.3/70.6	**70.3**	73.7	54.5	56.3
ELMo, Best of Each	**68.0**	**64.8**	**27.3**	**88.4**	**74.0/82.5**	**87.3/83.1**	**75.2/75.8**	**70.3**	**74.0**	**56.3**	56.3
Test Set Scores											
BERT	80.4	79.4	60.5	**94.9**	85.4/89.3	**89.3/72.1**	87.6/86.5	**86.3**	**91.1**	70.1	65.1
BERT on STILTs	**81.8**	**81.4**	**62.1**	94.3	**89.8/86.7**	89.4/71.9	**88.7/88.3**	86.0	**91.1**	**80.1**	65.1
GPT	74.1	71.9	45.4	91.3	82.3/75.7	**88.5/70.3**	82.0/80.0	**81.8**	88.1	56.0	65.1
GPT on STILTs	**76.9**	**75.9**	**47.2**	**93.1**	**87.7/83.7**	88.1/70.1	**85.3/84.8**	80.7	**87.2**	**69.1**	65.1
ELMo	62.2	59.0	16.2	**87.1**	79.7/69.1	84.9/63.9	64.3/63.9	69.0	57.1	52.3	65.1
ELMo on STILTs	**65.9**	**63.8**	**30.3**	86.5	**82.0/73.9**	**85.2/64.4**	**71.8/71.4**	**69.7**	**62.6**	**54.4**	65.1

Table 1: GLUE results with and without STILTs, fine-tuning on full training data of each target task. **Bold** marks the best within each section. ~~Strikethrough~~ indicates cases where the intermediate task is the same as the target task—we substitute the baseline result for that cell. *A.Ex* is the average excluding MNLI and QQP because of the overlap with intermediate tasks. See text for discussion of WNLI results. Test results *on STILTs* uses the supplementary training regime for each task based on the performance on the development set, corresponding to the numbers shown in *Best of Each*. The aggregated GLUE scores differ from the public leaderboard because we report performance on QNLIv1.

dition to the multi-genre MNLI to disambiguate between the benefits of domain shift and task shift from supplementary training on natural language inference. QQP is included as we believed it could improve performance on sentence similarity tasks such as MRPC and STS. Lastly, we construct a fake-sentence-detection task based on the BooksCorpus dataset in the style of Warstadt et al.. Importantly, because both GPT and BERT are pretrained on BooksCorpus, the fake-sentence-detection enables us to isolate the impact of task shift from domain shift from the pretaining corpus. We construct this task by sampling sentences from BooksCorpus, and fake sentences are generated by randomly swapping 2–4 pairs of words in the sentence. We generate a dataset of 600,000 sentences with a 50/50 real/fake split for this intermediate task.

Training Details Unless otherwise stated, for replications and both stages of our STILTs experiments, we follow the model formulation and training regime of BERT and the GPT specified in Devlin et al. and (Radford et al., 2018) respectively. Specifically, for both models we use a three-epoch training limit for both supplementary training and target-task fine-tuning. We use a fresh optimizer for each phase of training. For each task, we add only a single task-specific, randomly initialized output layer to the pretrained Transformer model, following the setup laid out by each respective work. For our baseline, we do not fine-tune on any intermediate task: Other than the batch size, this is equivalent to the formulation presented in the papers introducing BERT and GPT respectively and serves as our attempt to replicate their results.

For BERT, we use a batch size of 24 and a learn-

	Avg	**A.Ex**	**CoLA**	**SST**	**MRPC**	**QQP**	**STS**	**MNLI**	**QNLI**	**RTE**	**WNLI**
Training Set Size			*8.5k*	*67k*	*3.7k*	*364k*	*7k*	*393k*	*108k*	*2.5k*	*634*
At Most 5k Training Examples for Target Tasks											
BERT	78.3	78.1	**60.6**	**93.5**	87.3/91.0	83.1/78.6	90.2/89.8	77.1	82.8	74.0	56.3
BERT→QQP	77.6	77.3	55.3	92.0	88.0/91.4	~~83.1/78.6~~	90.7/90.5	75.9	81.6	76.5	56.3
BERT→MNLI	79.5	79.7	59.6	92.4	**89.5/92.5**	**83.7/78.1**	**91.1/90.6**	~~77.1~~	**83.9**	**83.4**	56.3
BERT→SNLI	78.8	78.2	56.6	91.5	88.2/91.6	83.0/77.9	90.8/90.6	**80.6**	82.7	80.5	56.3
BERT→Real/Fake	71.0	71.7	53.6	88.9	82.6/87.6	81.7/76.1	88.4/88.4	59.1	74.1	54.9	56.3
BERT, Best of Each	**80.1**	**79.9**	**60.6**	**93.5**	**89.5/92.5**	**83.7/78.1**	**91.1/90.6**	**80.6**	**83.9**	**83.4**	56.3
GPT	71.6	71.2	**50.8**	**91.1**	81.4/87.1	79.5/73.8	87.6/87.4	68.8	73.1	56.3	56.3
GPT→QQP	65.2	63.3	0.0	82.0	82.8/87.7	~~79.5/73.8~~	87.4/87.3	65.1	71.6	62.8	56.3
GPT→MNLI	72.3	71.8	35.3	89.4	**86.8/90.8**	**81.6/76.3**	88.8/88.7	~~68.8~~	74.1	**70.4**	56.3
GPT→SNLI	72.3	70.2	29.6	89.2	86.3/90.2	81.6/76.0	**89.5/89.4**	**78.3**	**74.7**	66.4	56.3
GPT→Real/Fake	71.4	69.3	45.1	87.8	78.2/85.2	80.6/75.4	87.8/87.5	77.5	72.2	56.3	56.3
GPT, Best of Each	**75.4**	**74.3**	**50.8**	**91.1**	**86.8/90.8**	**81.6/76.3**	**89.5/89.4**	**78.3**	**74.7**	**70.4**	56.3
At Most 1k Training Examples for Target Tasks											
BERT	74.2	74.5	**54.0**	**91.1**	83.8/88.4	79.9/73.8	88.1/87.9	69.7	77.0	69.0	56.3
BERT→QQP	73.2	73.5	47.5	89.7	82.1/86.9	~~79.9/73.8~~	88.6/88.5	67.5	76.4	71.5	56.3
BERT→MNLI	75.1	75.6	44.0	90.5	**85.5/90.0**	80.3/74.3	**88.7/88.7**	~~69.7~~	**79.0**	**82.7**	56.3
BERT→SNLI	75.5	74.7	47.6	89.3	82.8/87.8	**80.6/74.1**	87.8/88.1	**78.6**	77.6	79.1	56.3
BERT→Real/Fake	63.9	67.5	43.9	72.5	78.9/84.7	74.1/68.4	82.4/83.2	35.3	69.7	61.7	56.3
BERT, Best of Each	**77.3**	**77.1**	**54.0**	**91.1**	**85.5/90.0**	**80.6/74.1**	**88.7/88.7**	**78.6**	**79.0**	**82.7**	56.3
GPT	64.5	64.8	33.4	85.3	70.1/81.3	75.3/67.7	80.8/80.8	55.7	66.7	54.9	56.3
GPT→QQP	64.6	64.6	23.0	**87.0**	74.8/83.2	~~75.3/67.7~~	84.4/84.3	57.8	67.1	55.2	56.3
GPT→MNLI	65.2	65.2	13.3	86.2	79.2/85.8	**78.4/70.5**	86.2/86.1	~~55.7~~	68.6	**63.2**	56.3
GPT→SNLI	67.5	64.9	13.4	85.7	**80.1/86.2**	77.2/70.0	**87.5/87.5**	**76.8**	**70.3**	60.6	56.3
GPT→Real/Fake	65.3	62.5	**36.3**	69.7	69.6/79.6	75.5/69.4	84.7/84.8	74.6	69.1	50.2	56.3
GPT, Best of Each	**70.6**	**68.9**	**36.3**	**87.0**	**80.1/86.2**	**78.4/70.5**	**87.5/87.5**	**76.8**	**70.3**	**63.2**	56.3

Table 2: Results on the GLUE development set based on fine-tuning on only a subset of target-task data, simulating data scarce scenarios. **Bold** indicates the best within each section. ~~Strikethrough~~ indicates cases where the intermediate task is the same as the target task: We substitute the baseline result for that cell. *A.Ex* is the average excluding MNLI and QQP, because of their overlap with the candidate intermediate tasks. See text for discussion of WNLI results.

ing rate of 2e-5. This is within the range of hyperparameters recommended by the authors and initial experiments showed promising results. We use the larger, 24-layer version of BERT, which is the state of the art on the GLUE benchmark. For this model, fine-tuning can be unstable on small data sets—hence, for the tasks with limited data (CoLA, MRPC, STS, RTE), we perform 20 random restarts for each experiment and report the results of the model that performed best on the validation set.

For GPT, we choose the largest batch size out of 8/16/32 that a single GPU can accommodate. We use the version with an auxiliary language modeling objective in fine-tuning, corresponding to the entry on the GLUE leaderboard.[6]

For ELMo, to facilitate a fair comparison with GPT and BERT, we adopt a similar fine-tuning setup where all the weights are fine-tuned. This differs from the original ELMo setup that freezes ELMo weights and trains an additional encoder module when fine-tuning. The details of our ELMo setup are described in Appendix A.

We also run our main experiment on the 12-layer BERT and the non-LM fine-tuned GPT. These results are in Table 4 in the Appendix.

Multitask Learning Strategies To compare STILTs to alternative multitask learning regimes, we also experiment with the following two approaches: (i) a single phase of fine-tuning simultaneously on both a intermediate task and the target task (ii) fine-tuning simultaneously on a intermediate task and the target task, and then doing an additional phase of fine-tuning on the target task only. In the multitask learning phase, for both approaches, training steps are sampled proportionally to the sizes of the respective training sets and we do not weight the losses.

[6] Radford et al. (2018) introduced two versions of GPT: one which includes an auxiliary language modeling objective when fine-tuning, and one without.

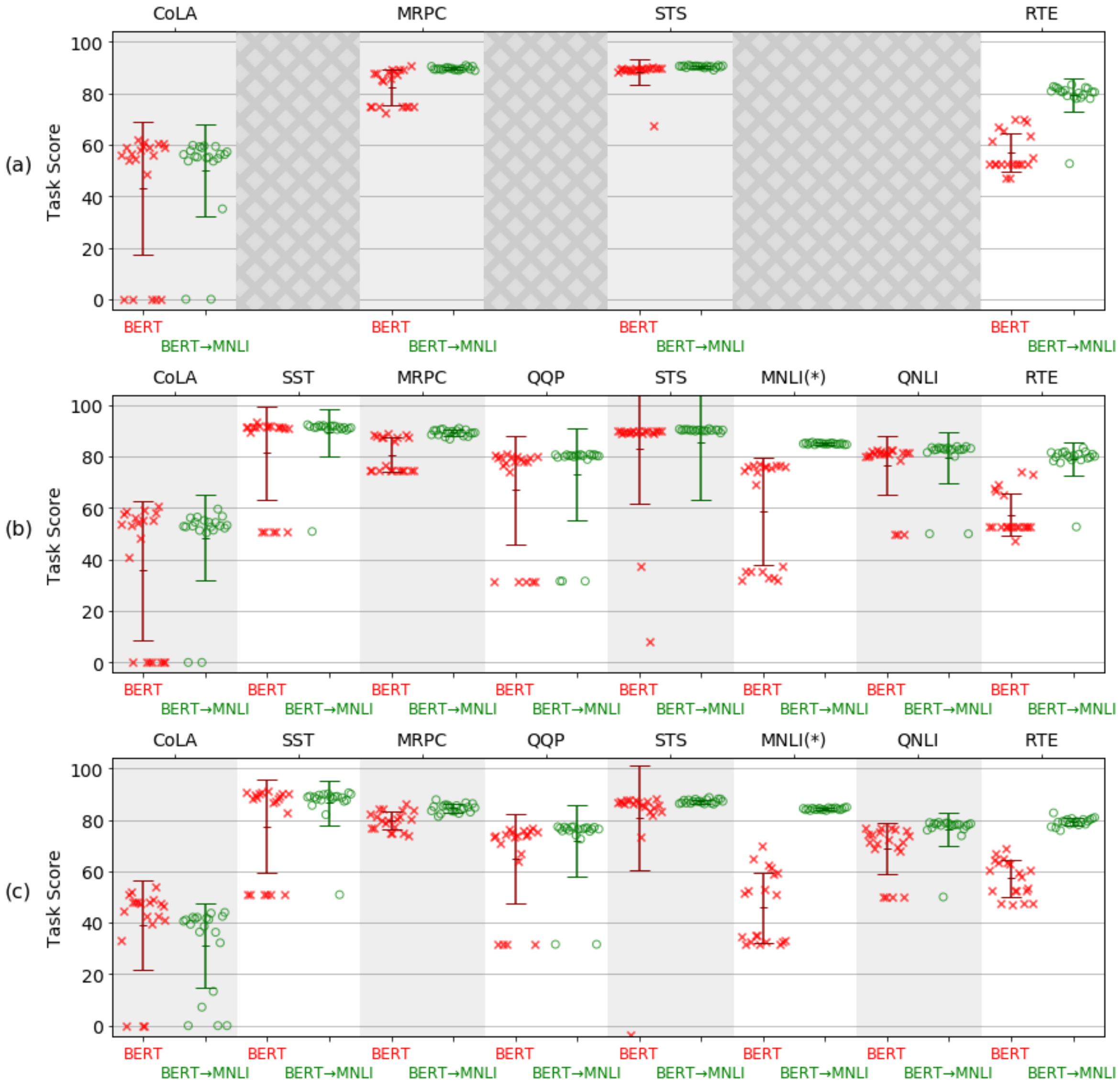

Figure 1: Distribution of task scores across 20 random restarts for BERT, and BERT with intermediary fine-tuning on MNLI. Each cross represents a single run. Error lines show mean±1std. (a) Fine-tuned on all data, for tasks with <10k training examples. (b) Fine-tuned on no more than 5k examples for each task. (c) Fine-tuned on no more than 1k examples for each task. (*) indicates that the intermediate task is the same as the target task.

Models and Code Our pretrained models and code for BERT on STILTs can be found at https://github.com/zphang/pytorch-pretrained-BERT, which is a fork of the Hugging Face implementation. We used the `jiant` framework experiments on GPT and ELMo.

4 Results

Table 1 shows our results on GLUE with and without STILTs. Our addition of supplementary training boosts performance across many of the two-sentence tasks. We also find that most of the gains are on tasks with limited data. On each of our STILTs models, we show improved overall GLUE scores on the development set. Improvements from STILTs tend to be larger for ELMo and GPT and somewhat smaller for BERT. On the other hand, for pairs of pretraining and target tasks that are close, such as MNLI and RTE, we indeed find a marked improvement in performance from STILTs. For the two single-sentence tasks—the syntax-oriented CoLA task and the SST sentiment task—we find somewhat deteriorated performance. For CoLA, this mirrors results reported in Bowman et al. (2019), who show that few pretraining tasks other than language modeling offer any advantage for CoLA. The *Best of Each* score is computed based on taking the best score for each task, including no STILTs.

On the test set, we see similar performance gains across most tasks. Here, we compute the results for each model *on STILTs*, which shows scores from choosing the best corresponding model based on development set scores and evaluating on the test set. These also correspond to the selected models for *Best of Each* above.[7] For both BERT and GPT, we show that using STILTs leads to improvements in test set performance improving on the reported baseline by 1.4 points and setting the state of the art for the GLUE benchmark, while GPT on STILTs achieves a score of 76.9, improving on the baseline by 2.8 points, and significantly closing the gap between GPT and the 12-layer BERT model with a similar number of parameters, which attains a GLUE score of 78.3.

Limited Target-Task Data Table 2 shows the same models fine-tuned on 5k training examples and 1k examples for each task, selected randomly without replacement. Artificially limiting the size of the training set allows us to examine the effect of STILTs in data constrained contexts. For tasks with training sets that are already smaller than these limits, we use the training sets as-is. For BERT, we show the maximum task performance across 20 random restarts for all experiments, and the data subsampling is also random for each restart.

The results show that the benefits of supplementary training are generally more pronounced in these settings, with performance in several tasks showing improvements of more than 10 points. CoLA and SST are again the exceptions: Both tasks deteriorated moderately with supplementary training, and CoLA trained with the auxiliary language modeling objective in particular showed highly unstable results when trained on small amounts of data.

We see one obvious area for potential improvement: In our experiments, we follow the recipe for fine-tuning from the original works as closely as possible, only doing supplementary training and fine-tuning for three epochs each. Particularly in the case of the artificially data-constrained tasks, we expect that performance could be improved with more careful tuning of the training duration and learning rate schedule.

Fine-Tuning Stability In the work that introduced BERT, Devlin et al. highlight that the larger, 24-layer version of BERT is particularly prone to degenerate performance on tasks with small training sets, and that multiple random restarts may be required to obtain a usable model. In Figure 1, we plot the distribution of performance scores for 20 random restarts for each task, using all training data and maximum of 5k or 1k training examples. For conciseness, we only show results for BERT without STILTs, and BERT with intermediate fine-tuning on MNLI. We omit the random restarts for tasks with training sets of more than 10k examples, consistent with our training methodology.

We show that, in addition to improved performance, using STILTs significantly reduces the variance of performance across random restarts. A large part of reduction can be attributed to the far fewer number of degenerate runs—performance outliers that are close to random guessing. This effect is consistent across target tasks, though the magnitude varies from task to task. For instance, although we show above that STILTs with our four intermediate tasks does not improve model performance in CoLA and SST, using STILTs nevertheless reduces the variance across runs as well as the number of degenerate fine-tuning results.

Multitask Learning and STILTs We investigate whether setups that leverage multitask learning are more effective than STILTs. We highlight results from one of the cases with the largest improvement: GPT with intermediary fine-tuning on MNLI with RTE as the target task. To better isolate the impact of multitask learning, we exclude the auxiliary language modeling training objective in this experiment. Table 3 shows all setups improve compared to only fine-tuning, with the STILTs format of consecutive single-task fine-tuning having the largest improvement. Although this does not represent an in-depth inquiry of all the ways to leverage multitask learning and balance multiple training objective, naive multitask learning appears to yield worse performance than STILTs, at potentially greater computational cost.

5 Discussion

Broadly, we have shown that, across three different sentence encoders with different architectures

[7] For BERT, we run an additional 80 random restarts–100 random restarts in total–for the tasks with limited data, and select the best model based on validation score for test evaluation

Model	RTE accuracy
GPT → RTE	54.2
GPT → MNLI → RTE	**70.4**
GPT → {MNLI, RTE}	68.6
GPT → {MNLI, RTE} → RTE	67.5

Table 3: Comparison of STILTs against multitask learning setups for GPT, with MNLI as the intermediate task, and RTE as the target task. GPT is fine-tuned without the auxiliary language modeling objective in this experiment. Both intermediary and final fine-tuning task(s) are delineated here, in contrast to Table 1 and Table 2 where we omit the name of the target-task.

and pretraining schemes, STILTs can leads to performance gains on many downstream target tasks. However, this benefit is not uniform. We find that sentence pair tasks seem to benefit more from supplementary training than single-sentence ones. We also find that tasks with little training data benefit much more from supplementary training. Indeed, when applied to RTE, supplementary training on the related MNLI task leads to a eight-point increase in test set score for BERT.

Overall, the benefit of STILTs is smaller for BERT than for GPT and ELMo. One possible reason is that BERT is better conditioned for fine-tuning for classification tasks, such as those in the GLUE Benchmark. Indeed, GPT uses the hidden state corresponding to the last token of the sentence as a proxy to encode the whole sentence, but this token is not used for classification during pre-training. On the other hand, BERT has a <CLS> token which is used for classification during pre-training for their additional next-sentence-prediction objective. This token is then used in fine-tuning for classification. When adding STILTs to GPT, we bridge that gap by training the last token with the classification objective of the intermediary task. This might explain why fake-sentence-detection is a broadly beneficial task for GPT and not for BERT: Since fake-sentence-detection uses the same corpus that GPT and BERT are pretrained on, it is likely that the improvements we find for GPT are due to the better conditioning of this sentence-encoding token.

Applying STILTs also comes with little complexity or computational overhead. The same infrastructure used to fine-tune BERT or GPT models can be used to perform supplementary training. The computational cost of the supplementary training phase is another phase of fine-tuning, which is small compared to the cost of training the original model. In addition, in the case of BERT, the smaller number of degenerate runs induced by STILTs will reduce the computational cost of a full training procedure in some settings.

Our results also show where STILTs may be ineffective or counterproductive. In particular, we show that most of our intermediate tasks were actually detrimental to the single-sentence tasks in GLUE. The interaction between the intermediate task, the target task, and the use of the auxiliary language modeling objective is a subject due for further investigation. Moreover, the four intermediary training tasks we chose represent only a small sample of potential tasks, and it is likely that a more expansive survey might yield better performance on different downstream tasks. Therefore, for best target task performance, we recommend experimenting with supplementary training with several closely-related data-rich tasks and use the development set to select the most promising approach for each task, as in the *Best of Each* formulation shown in Table 1.

6 Conclusion

This work represents only an initial investigation into the benefits of supplementary supervised pretraining. More work remains to be done to firmly establish when methods like STILTs can be productively applied and what criteria can be used to predict which combinations of intermediate and target tasks should work well. Nevertheless, in our initial work with four example intermediate training tasks, we showed significant gains from applying STILTs to three sentence encoders, BERT, GPT and ELMo, and set the state of the art on the GLUE benchmark with BERT on STILTs. STILTs also helps to significantly stabilize training in unstable training contexts, such as when using BERT on tasks with little data. Finally, we show that in data-constrained regimes, the benefits of using STILTs are even more pronounced, yielding up to 10 point score improvements on some intermediate/target task pairs.

Acknowledgments

We would like to thank Alex Wang, Ilya Kulikov, Nikita Nangia and Phu Mon Htut for their helpful feedback.

References

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Samuel R. Bowman, Ellie Pavlick, Edouard Grave, Benjamin Van Durme, Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, and Berlin Chen. 2019. Looking for ELMo's friends: Sentence-level pretraining beyond language modeling. *arXiv preprint 1812.10860*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval-2017*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint 1810.04805*.

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

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Emma Kerinec, Chloé Braud, and Anders Søgaard. 2018. When does deep multi-task learning work for loosely related document classification tasks? In *Proc. EMNLP Workshop BlackboxNLP*.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In *Aaai spring symposium: Logical formalizations of commonsense reasoning*, volume 46, page 47.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *NAACL*.

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *EMNLP*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Unpublished manuscript accessible via the OpenAI Blog.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

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 *EMNLP*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *ICLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint 1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. 2017. Inference is everything: Recasting semantic resources into a unified evaluation framework. In *Proc. Eighth International Joint Conference on Natural Language Processing*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Kelly Zhang and Samuel R. Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv preprint 1809.10040*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015a. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015b. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*.

A ELMo on STILTs

Experiment setup We use the same architecture as Peters et al. (2018a) for the non-task-specific parameters. For task-specific parameters, we use the layer weights and the task weights described in the paper, as well as a classifier composed of max-pooling with projection and a logistic regression classifier. In contrast to the GLUE baselines and to Bowman et al. (2019), we refrain from adding many non-LM pretrained parameters by not using pair attention nor an additional encoding layer. The whole model, including ELMo parameters, is trained during both supplementary training on the intermediate task and target-task tuning. For two-sentence tasks, we follow the model design of Wang et al. (2018) rather than that of Radford et al. (2018), since early experiments showed better performance with the former. Consequently, we run the shared encoder on the two sentences u and u' independently and then use $[u'; v'; |u' - v'|; u' * v']$ for our task-specific classifier. We use the default optimizer and learning rate schedule from `jiant`.

	Avg	AvgEx	CoLA	SST	MRPC	QQP	STS	MNLI	QNLI	RTE	WNLI
Training Set Size			*8.5k*	*67k*	*3.7k*	*364k*	*7k*	*393k*	*108k*	*2.5k*	*634*
						Development Set Scores					
BERT	79.2	76.7	55.2	92.5	86.8/90.9	**90.8/87.7**	88.9/88.5	84.4	88.8	68.6	56.3
BERT→QQP	78.6	76.0	49.7	91.5	84.3/89.0	~~90.8/87.7~~	89.7/89.5	83.7	87.7	72.6	56.3
BERT→MNLI	81.1	79.2	**59.0**	**92.7**	**88.5/91.9**	90.8/87.5	**90.3/90.2**	~~84.4~~	**89.0**	**79.1**	56.3
BERT→SNLI	79.9	77.5	52.9	**92.7**	87.0/90.7	90.9/87.6	89.9/89.8	**84.8**	88.4	76.5	56.3
BERT→Real/Fake	77.8	75.0	53.1	92.0	82.6/88.4	90.5/87.3	89.3/88.8	83.4	87.5	64.3	56.3
BERT, Best of Each	**81.2**	**79.3**	**59.0**	**92.7**	**88.5/91.9**	**90.8/87.7**	**90.3/90.2**	**84.8**	**89.0**	**79.1**	56.3
GPT	75.3	72.7	**52.8**	**92.3**	80.6/86.4	88.2/84.6	87.5/87.2	79.6	81.5	57.8	56.3
GPT→QQP	73.1	69.7	29.8	91.4	82.8/87.7	~~88.2/84.6~~	87.4/87.3	80.1	78.9	62.8	56.3
GPT→MNLI	76.2	74.1	41.5	91.9	**86.8/90.8**	88.8/81.3	89.2/89.0	~~79.6~~	**83.1**	**70.4**	56.3
GPT→SNLI	75.4	72.5	35.3	90.9	86.3/90.2	**89.0/85.4**	**90.1/89.8**	**81.2**	82.9	66.4	56.3
GPT→Real/Fake	74.9	71.9	50.3	92.1	78.2/85.2	88.4/84.7	88.3/88.1	**81.2**	81.8	56.3	56.3
GPT, Best of Each	**78.0**	**75.9**	**52.8**	**92.3**	**86.8/90.8**	**89.0/85.4**	**90.1/89.8**	**81.2**	**83.1**	**70.4**	56.3

Table 4: Results on the GLUE development set with and without STILTs, fine-tuning on full training data of each target task. BERT results are based on the 12-layer model, while GPT results are without an auxiliary language modeling objective. **Bold** indicates the best within each section. ~~Strikethrough~~ indicates cases where the intermediate task is the same as the target task–we substitute the baseline result for that cell. *A.Ex* is the average excluding MNLI and QQP because of the overlap with intermediate tasks. See text for discussion of WNLI results.