How to Get Past Sesame Street: Sentence-Level Pretraining Beyond Language Modeling

Alex Wang,[*1] **Jan Hula,**[5] **Patrick Xia,**[5] **Raghavendra Pappagari,**[5]
R. Thomas McCoy,[5], **Roma Patel,**[2] **Najoung Kim,**[5] **Ian Tenney,**[3] **Yinghui Huang,**[6]
Katherin Yu,[4] **Shuning Jin,**[7] **Berlin Chen**[8] **Benjamin Van Durme,**[5] **Edouard Grave,**[4]
Ellie Pavlick,[2,3] **Samuel R. Bowman**[1]
[1]New York University, [2]Brown University, [3]Google AI Language, [4]Facebook,
[5]Johns Hopkins University, [6]IBM, [7]University of Minnesota Duluth, [8]Swarthmore College

Abstract

Natural language understanding has recently seen a surge of progress with the use of sentence encoders like ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019) which are pretrained on variants of language modeling. We conduct the first large-scale systematic study of candidate pretraining tasks, comparing 19 different tasks both as alternatives and complements to language modeling. Our primary results support the use language modeling, especially when combined with pretraining on additional labeled-data tasks. However, our results are mixed across pretraining tasks and show some concerning trends: In ELMo's pretrain-then-freeze paradigm, random baselines are worryingly strong and results vary strikingly across target tasks. In addition, fine-tuning BERT on an intermediate task often negatively impacts downstream transfer. We also see modest gains from multitask training, suggesting the development of more sophisticated multitask and transfer learning techniques as an avenue for further research.

1 Introduction

State-of-the-art models in natural language processing (NLP) often incorporate encoder functions which generate a sequence of vectors intended to represent the in-context meaning of each word in an input text. These encoders are typically trained directly on the target task at hand, which can be effective for data-rich tasks and yields human performance on some narrowly-defined benchmarks (Rajpurkar et al., 2018; Hassan et al., 2018), but is tenable only for the few tasks with millions of training data examples. This limitation has prompted interest in *pretraining* for these encoders: The encoders are first trained on outside data, and then plugged into a target task model.

Howard and Ruder (2018), Peters et al. (2018a), Radford et al. (2018), and Devlin et al. (2019) establish that encoders pretrained on *language modeling* can be reused to yield strong performance on downstream NLP tasks. Subsequent work has homed in on language modeling (LM) pretraining, finding that such models can be productively fine-tuned on *intermediate* tasks like natural language

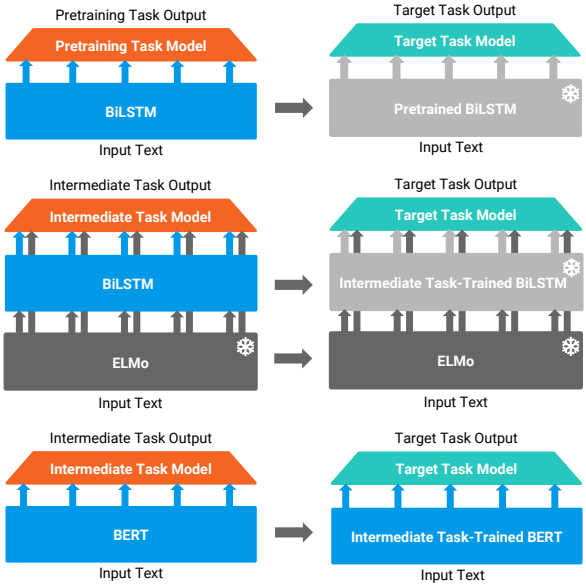

Figure 1: Learning settings that we consider. Model components with frozen parameters are shown in gray and decorated with snowflakes. **Top** (pretraining): We pretrain a BiLSTM on a task (left), and learn a target task model on top of the representations it produces (right). **Middle** (intermediate ELMo training): We train a BiLSTM on top of ELMo for an intermediate task (left). We then train a target task model on top of the intermediate task LSTM and ELMo (right). **Bottom** (intermediate BERT training): We fine-tune BERT on an intermediate task (left), and then fine-tune the resulting model again on a target task (right).

This paper supercedes "Looking for ELMo's Friends: Sentence-Level Pretraining Beyond Language Modeling", an earlier version of this work by the same authors. Correspondence to alexwang@nyu.edu

inference before transferring to downstream tasks (Phang et al., 2018). However, we identify two open questions: (1) How effective are tasks beyond language modeling in training reusable sentence encoders (2) Given the recent successes of LMs with intermediate-task training, which pretraining tasks can be effectively combined.

The main contribution of this paper is a large-scale, systematic study of these two questions. For the first question, we train reusable sentence encoders on 19 different pretraining tasks and task combinations and several simple baselines, using a standardized model architecture and procedure for pretraining. For the second question, we conduct additional pretraining on ELMo (Peters et al., 2018b) and BERT (Devlin et al., 2019) with 17 different intermediate tasks and task combinations. We evaluate each of these encoders on the nine target language-understanding tasks in the GLUE benchmark (Wang et al., 2019), yielding a total of 53 sentence encoders and 477 total trained models. We measure correlation in performance across target tasks and plot learning curves to show the effect of data volume on both pretraining and target task training.

We find that language modeling is the most effective pretraining task that we study. Multitask pretraining or intermediate task training offers modest further gains. However, we see several worrying trends: (i) The margins between substantially different pretraining tasks can be extremely small in this transfer learning regimen, and many pretraining tasks struggle to outperform trivial baselines. (ii) Many of the tasks used for intermediate task training *adversely* impact the transfer ability of LM pretraining, with some signs of catastrophic forgetting. (iii) Different target tasks differ dramatically in what kinds of pretraining they benefit most from, but naïve multitask pretraining seems unable to leverage the relative strengths of disparate pretraining tasks. These observations suggest that while scaling up LM pretraining (as in Radford et al., 2019) is likely the most straightforward path to further gains, our current methods for multitask and transfer learning may be substantially limiting our results.

2 Related Work

Work on reusable sentence encoders can be traced back at least as far as the multitask model of Collobert et al. (2011). Several works focused on learning reusable *sentence-to-vector* encodings, where the pretrained encoder produces a fixed-size representation for each input sentence (Dai and Le, 2015; Kiros et al., 2015; Hill et al., 2016; Conneau et al., 2017). More recent reusable sentence encoders such as CoVe (McCann et al., 2017) and GPT (Radford et al., 2018) instead represent sentences as *sequences* of vectors. These methods work well, but most use distinct pretraining objectives, and none offers a substantial investigation of the choice of objective like we conduct here.

We build on two methods for pretraining sentence encoders on language modeling: ELMo and BERT. ELMo consists of a forward and backward LSTM (Hochreiter and Schmidhuber, 1997), the hidden states of which are used to produce a contextual vector representation for each token in the inputted sequence. ELMo is adapted to target tasks by freezing the model weights and only learning a set of task-specific scalar weights that are used to compute a linear combination of the LSTM layers. BERT consists of a pretrained Transformer (Vaswani et al., 2017), and is adapted to downstream tasks by fine-tuning the entire model. Follow-up work has explored parameter-efficient fine-tuning (Stickland and Murray, 2019; Houlsby et al., 2019) and better target task performance during fine-tuning (Phang et al., 2018; Liu et al., 2019), but work in this area is nascent.

The successes of sentence encoder pretraining have sparked a line of work analyzing these models (Zhang and Bowman, 2018; Peters et al., 2018b; Tenney et al., 2019, i.a.). Our work also attempts to better understand what is learned by pretrained encoders, but we study this question entirely through the lens of pretraining and fine-tuning tasks, rather than architectures or specific linguistic capabilities.

Multitask representation learning in NLP is well studied, and again can be traced back at least as far as Collobert et al. (2011). Luong et al. (2016) show promising results combining translation and parsing, Subramanian et al. (2018) benefit from multitask learning in sentence-to-vector encoding, and Bingel and Søgaard (2017) and Changpinyo et al. (2018) offer studies of when multitask learning is helpful for lower-level NLP tasks.

3 Transfer Paradigms

We consider two recent paradigms for transfer learning: pretraining and intermediate training.

See Figure 1 for a graphical depiction.

Pretraining Our first set of experiments is designed to systematically investigate the effectiveness of a broad range of tasks in pretraining sentence encoders. For each task, we first train a randomly initialized model to convergence on that *pretraining task*, and then train a model for a *target task* on top of the trained encoder. For these experiments, we largely follow the procedure and architecture used by ELMo rather than BERT, but we expect similar trends with BERT-style models.

Intermediate Training Given the robust success of LM pretraining, we explore methods of further improving on such sentence encoders. In particular, we take inspiration from Phang et al. (2018), who show gains in first fine-tuning BERT on an *intermediate task*, and then fine-tuning again on a target task. Our second set of experiments investigates which tasks can be used for intermediate training to augment LM pretraining. We design experiments using both pretrained ELMo and BERT as the base encoder. When using ELMo, we follow standard procedure and train a task-specific LSTM and output component (e.g. MLP for classification, decoder for sequence generation, etc.) on top of the representations produced by ELMo. During this stage, the pretrained ELMo weights are frozen except for a set of layer mixing weights. When using BERT, we follow standard procedure and train a small task-specific output component using the `[CLS]` output vector while also fine-tuning the weights of the full BERT model.

Target Task Evaluation For our pretraining and intermediate ELMo experiments, to evaluate on a target task, we learn a target task model on top of the representations produced by the encoder, which is again frozen throughout target task training except for a set of target-task-specific layer mixing weights. For our intermediate BERT experiments, we follow the same procedure as in intermediate training: we learn a target-task model using the `[CLS]` representation and fine-tune the encoder throughout target task training.

We use the nine target tasks in GLUE (Wang et al., 2019) to evaluate each of the encoders we train. GLUE is an open-ended shared task competition and evaluation toolkit for reusable sentence encoders, built around a set of nine sentence and sentence pairs tasks spanning a range of dataset sizes, paired with private test data and an online leaderboard. We evaluate each model on each of the nine tasks, and report the resulting scores and the GLUE score, a macro-average over tasks.

4 Tasks

Our experiments compare encoders pretrained or fine-tuned on a large number of tasks and task combinations, where a *task* is a dataset–objective function pair. We select these tasks either to serve as baselines or because they have shown promise in prior work, especially in sentence-to-vector encoding. See Appendix A for details and tasks we experimented with but which did not show strong enough performance to warrant a full evaluation.

Random Encoder A number of recent works have noted that randomly initialized, untrained LSTMs can obtain surprisingly strong downstream task performance (Zhang and Bowman, 2018; Wieting and Kiela, 2019; Tenney et al., 2019). Our pretraining and ELMo experiments include a baseline of a randomly initialized BiLSTM with no further training. This baseline is especially strong because our ELMo-style models use a skip connection from the input of the encoder to the output, allowing the task-specific component to see the input representations, yielding a model similar to Iyyer et al. (2015).

GLUE Tasks We use the nine tasks included with GLUE as pretraining and intermediate tasks: 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), Quora Question Pairs[1] (**QQP**), and STS-Benchmark (**STS**; Cer et al., 2017); and textual entailment with the Multi-Genre NLI Corpus (**MNLI** Williams et al., 2018), RTE 1, 2, 3, and 5 (**RTE**; Dagan et al., 2006, et seq.), and data from SQuAD (**QNLI**[2], Rajpurkar et al., 2016) and the Winograd Schema Challenge (**WNLI**, Levesque et al., 2011) recast as entailment in the style of White et al. (2017). MNLI and QQP have previously been shown to be effective for pretraining in other settings (Conneau et al., 2017; Subramanian et al., 2018; Phang et al., 2018). Other tasks are included to represent a broad sample of labeling schemes commonly used in NLP.

[1] data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[2] QNLI has been re-released with updated splits since the original release. We use the original splits.

Outside Tasks We train **language models** on two datasets: WikiText-103 (WP, Merity et al., 2017) and Billion Word Language Model Benchmark (BWB, Chelba et al., 2013). Because representations from ELMo and BERT capture left and right context, they cannot be used in conjunction with unidirectional language modeling, so we exclude this task from intermediate training experiments. We train **machine translation** (MT) models on WMT14 English-German (Bojar et al., 2014) and WMT17 English-Russian (Bojar et al., 2017). We train **SkipThought**-style sequence-to-sequence (seq2seq) models to read a sentence from WP and predict the following sentence (Kiros et al., 2015; Tang et al., 2017). We train **DisSent** models to read two clauses from WP that are connected by a discourse marker such as *and*, *but*, or *so* and predict the the discourse marker (Jernite et al., 2017; Nie et al., 2017). Finally, we train seq2seq models to predict the response to a given comment from **Reddit** (`reddit.com`), using a dataset of 18M comment–response pairs from 2008-2011 that was collected by Yang et al. (2018) to train sentence encoders.

Multitask Learning We consider three sets of these tasks for multitask pretraining and intermediate training: all GLUE tasks, all non-GLUE (outside) tasks, and all tasks.

5 Models and Experimental Details

We implement our models using AllenNLP (Gardner et al., 2017) and a public implementation of BERT[3]. Appendix A presents additional details.

Encoder Architecture For both the pretraining and intermediate ELMo experiments, we process words using a pretrained character-level convolutional neural network (CNN) from ELMo. We use this pretrained word encoder for pretraining experiments to avoid potentially difficult issues with unknown word handling in transfer learning.

For the pretraining experiments, these input representations are fed to a two-layer 1024D bidirectional LSTM from which we take the sequence of hidden states from the top layer as the contextual representation. A task-specific model sees both the top-layer hidden states of this model and, through a skip connection, the input token representations. For the intermediate ELMo experiments, we compute contextual representations using the entire pretrained ELMo model, which are passed to a similar LSTM that is then trained on the intermediate task. We also include a skip connection from the ELMo representations to the task specific model. Our experiments with BERT use the base, cased architecture.

Task-Specific Components We design task-specific components to be as close to standard models for each task as possible. Though different components may have varying parameter counts, architectures, etc., we believe that results between tasks are still comparable and informative.

For most BERT experiments we use the standard preprocessing and pass the representation of the special `[CLS]` representation to a logistic regression classifier. For seq2seq tasks (MT, SkipThought, Reddit) we replace the classifier with a single-layer LSTM word-level decoder initialized with the `[CLS]` representation.

For ELMo-style models, we use several model types. For **single-sentence classification tasks**, we train a linear projection over the output states of the encoder, max-pool those projected states, and feed the result to an MLP. When pretraining on any **sentence-pair task** or training target-task models for MRPC, RTE, WNLI, and DisSent, we perform these steps on both sentences and use the heuristic feature vector $[h_1; h_2; h_1 \cdot h_2; h_1 - h_2]$ in the MLP, following Mou et al. (2016). When training target-task models on the remaining sentence-pair tasks, we use a cross-sentence attention mechanism similar to BiDAF (Seo et al., 2017). We do not use this mechanism in other cases as early results indicated that it hurt transfer performance. For **seq2seq tasks** (MT, SkipThought, Reddit), we use a single-layer LSTM decoder initialized with the pooled representation of the input. For **language modeling**, we follow ELMo by concatenating forward and backward language models to prevent information leakage across directions and learning layer mixing weights.

To use GLUE tasks for pretraining or intermediate training in a way that is more comparable to outside tasks, after pretraining we discard the learned GLUE classifier, and initialize a new classifier from scratch for target-task training.

Optimization For BERT experiments, we train our models with the same optimizer and learning rate schedule as the original work. For all

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

Pretr.	Avg	CoLA	SST	MRPC	QQP	STS	MNLI	QNLI	RTE	WNLI
					Baselines					
Random	68.2	16.9	84.3	77.7/85.6	83.0/80.6	81.7/82.6	73.9	**79.6**	57.0	31.0*
Single-Task	69.1	21.3	89.0	77.2/84.7	84.7/81.9	81.4/82.2	74.8	78.8	56.0	11.3*
					GLUE Tasks as Pretraining Tasks					
CoLA	68.2	21.3	85.7	75.0/83.7	85.7/82.4	79.0/80.3	72.7	78.4	56.3	15.5*
SST	68.6	16.4	89.0	76.0/84.2	84.4/81.6	80.6/81.4	73.9	78.5	58.8	19.7*
MRPC	68.2	16.4	85.6	77.2/84.7	84.4/81.8	81.2/82.2	73.6	79.3	56.7	22.5*
QQP	68.0	14.7	86.1	77.2/84.5	84.7/81.9	81.1/82.0	73.7	78.2	57.0	45.1*
STS	67.7	14.1	84.6	77.9/85.3	81.7/79.2	81.4/82.2	73.6	79.3	57.4	43.7*
MNLI	69.1	16.7	88.2	78.9/85.2	84.5/81.5	81.8/82.6	74.8	**79.6**	58.8	36.6*
QNLI	67.9	15.6	84.2	76.5/84.2	84.3/81.4	80.6/81.8	73.4	78.8	58.8	**56.3**
RTE	68.1	18.1	83.9	77.5/85.4	83.9/81.2	81.2/82.2	74.1	79.1	56.0	39.4*
WNLI	68.0	16.3	84.3	76.5/84.6	83.0/80.5	81.6/82.5	73.6	78.8	58.1	11.3*
					Non-GLUE Pretraining Tasks					
DisSent WP	68.6	18.3	86.6	79.9/86.0	85.3/82.0	79.5/80.5	73.4	79.1	56.7	42.3*
LM WP	70.1	30.8	85.7	76.2/84.2	86.2/82.9	79.2/80.2	74.0	79.4	60.3	25.4*
LM BWB	**70.4**	30.7	86.8	79.9/86.2	**86.3/83.2**	80.7/81.4	74.2	79.0	57.4	47.9*
MT En-De	68.1	16.7	85.4	77.9/84.9	83.8/80.5	82.4/82.9	73.5	**79.6**	55.6	22.5*
MT En-Ru	68.4	16.8	85.1	79.4/86.2	84.1/81.2	82.7/83.2	74.1	79.1	56.0	26.8*
Reddit	66.9	15.3	82.3	76.5/84.6	81.9/79.2	81.5/81.9	72.7	76.8	55.6	53.5*
SkipThought	68.7	16.0	84.9	77.5/85.0	83.5/80.7	81.1/81.5	73.3	79.1	**63.9**	49.3*
					Multitask Pretraining					
MTL GLUE	68.9	15.4	**89.9**	78.9/86.3	82.6/79.9	**82.9/83.5**	**74.9**	78.9	57.8	38.0*
MTL Non-GLUE	69.9	30.6	87.0	**81.1/87.6**	86.0/82.2	79.9/80.6	72.8	78.9	54.9	22.5*
MTL All	**70.4**	**33.2**	88.2	78.9/85.9	85.5/81.8	79.7/80.0	73.9	78.7	57.4	33.8*
					Test Set Results					
LM BWB	66.5	29.1	86.9	75.0/82.1	82.7/63.3	74.0/73.1	73.4	68.0	51.3	65.1
MTL All	68.5	36.3	88.9	77.7/84.8	82.7/63.6	77.8/76.7	75.3	66.2	53.2	65.1

Table 1: Results for **pretraining** experiments on development sets except where noted. **Bold** denotes best result overall. Underlining denotes an average score surpassing the *Random* baseline. See Section 6 for discussion of WNLI results (*).

other models, we train our models with AMSGrad (Reddi et al., 2018). We do early stopping using dev set performance of the task we are training on. Typical experiments (pretraining or intermediate training of an encoder and training nine associated target-task models) take 1–5 days to complete on an NVIDIA P100 GPU.

Hyperparameters Appendix B lists the hyperparameter values used. As our experiments require more than 150 GPU-days on NVIDIA P100 GPUs to run—not counting debugging or learning curves—we do not have the the resources for extensive tuning. Instead, we fix most hyperparameters to commonly used values. The lack of tuning limits our ability to diagnose the causes of poor performance when it occurs, and we invite readers to further refine our models using the public code.

6 Results

Tables 1 and 2 respectively show results for our pretraining and intermediate training experiments. The *Single-Task* baselines train and evaluate a model on only the corresponding GLUE task. To comply with GLUE's limits on test set access, we only evaluate the top few pretrained encoders. For roughly comparable results in prior work, see Wang et al. (2019) or `www.gluebenchmark.com`; we omit them here in the interest of space. As of writing, the best test result using a comparable frozen pretrained encoder is 70.0 from Wang et al. (2019) for a model similar to our GLUE^E, and the best overall result is 82.9 from Liu et al. (2019) using a model similar to our GLUE^B (below), but substantially larger.

While not feasible to run each setting multiple times, we estimate the variance of the GLUE score by re-running three experiments five times each with different random seeds. We observe

Intermediate Task	Avg	CoLA	SST	MRPC	QQP	STS	MNLI	QNLI	RTE	WNLI
ELMo with Intermediate Task Training										
Random^E	70.5	38.5	87.7	79.9/86.5	86.7/83.4	80.8/82.1	75.6	79.6	**61.7**	33.8*
Single-Task^E	71.2	39.4	**90.6**	77.5/84.4	86.4/82.4	79.9/80.6	75.6	78.0	55.6	11.3*
CoLA^E	71.1	39.4	87.3	77.5/85.2	86.5/83.0	78.8/80.2	74.2	78.2	59.2	33.8*
SST^E	71.2	38.8	**90.6**	80.4/86.8	87.0/83.5	79.4/81.0	74.3	77.8	53.8	43.7*
MRPC^E	71.3	40.0	88.4	77.5/84.4	86.4/82.7	79.5/80.6	74.9	78.4	58.1	**54.9***
QQP^E	70.8	34.3	88.6	79.4/85.7	86.4/82.4	81.1/82.1	74.3	78.1	56.7	38.0*
STS^E	71.6	39.9	88.4	79.9/86.4	86.7/83.3	79.9/80.6	74.3	78.6	58.5	26.8*
MNLI^E	72.1	38.9	89.0	80.9/86.9	86.1/82.7	81.3/82.5	75.6	79.7	58.8	16.9*
QNLI^E	71.2	37.2	88.3	81.1/86.9	85.5/81.7	78.9/80.1	74.7	78.0	58.8	22.5*
RTE^E	71.2	38.5	87.7	81.1/87.3	86.6/83.2	80.1/81.1	74.6	78.0	55.6	32.4*
WNLI^E	70.9	38.4	88.6	78.4/85.9	86.3/82.8	79.1/80.0	73.9	77.9	57.0	11.3*
DisSent WP^E	71.9	39.9	87.6	**81.9/87.2**	85.8/82.3	79.0/80.7	74.6	79.1	61.4	23.9*
MT En-De^E	72.1	40.1	87.8	79.9/86.6	86.4/83.2	81.8/82.4	75.9	79.4	58.8	31.0*
MT En-Ru^E	70.4	**41.0**	86.8	76.5/85.0	82.5/76.3	81.4/81.5	70.1	77.3	60.3	45.1*
Reddit^E	71.0	38.5	87.7	77.2/85.0	85.4/82.1	80.9/81.7	74.2	79.3	56.7	21.1*
SkipThought^E	71.7	40.6	87.7	79.7/86.5	85.2/82.1	81.0/81.7	75.0	79.1	58.1	52.1*
MTL GLUE^E	72.1	33.8	90.5	81.1/87.4	86.6/83.0	82.1/83.3	**76.2**	79.2	61.4	42.3*
MTL Non-GLUE^E	**72.4**	39.4	88.8	80.6/86.8	**87.1/84.1**	**83.2/83.9**	75.9	**80.9**	57.8	22.5*
MTL All^E	72.2	37.9	89.6	79.2/86.4	86.0/82.8	81.6/82.5	76.1	80.2	60.3	31.0*
BERT with Intermediate Task Training										
Single-Task^B	78.8	56.6	90.9	88.5/91.8	89.9/86.4	86.1/86.0	83.5	**87.9**	69.7	**56.3**
CoLA^B	78.3	**61.3**	91.1	87.7/91.4	89.7/86.3	85.0/85.0	83.3	85.9	64.3	43.7*
SST^B	78.4	57.4	**92.2**	86.3/90.0	89.6/86.1	85.3/85.1	83.2	87.4	67.5	43.7*
MRPC^B	78.3	60.3	90.8	87.0/91.1	89.7/86.3	86.6/86.4	**83.8**	83.9	66.4	**56.3**
QQP^B	79.1	56.8	91.3	88.5/91.7	**90.5/87.3**	88.1/87.8	83.4	87.2	69.7	**56.3**
STS^B	79.4	61.1	92.3	88.0/91.5	89.3/85.5	86.2/86.0	82.9	87.0	71.5	50.7*
MNLI^B	**79.6**	56.0	91.3	88.0/91.3	90.0/86.7	87.8/87.7	82.9	87.0	**76.9**	**56.3**
QNLI^B	78.4	55.4	91.2	**88.7/92.1**	89.9/86.4	86.5/86.3	82.9	86.8	68.2	**56.3**
RTE^B	77.7	59.3	91.2	86.0/90.4	89.2/85.9	85.9/85.7	82.0	83.3	65.3	**56.3**
WNLI^B	76.2	53.2	92.1	85.5/90.0	89.1/85.5	85.6/85.4	82.4	82.5	58.5	**56.3**
DisSent WP^B	78.1	58.1	91.9	87.7/91.2	89.2/85.9	84.2/84.1	82.5	85.5	67.5	43.7*
MT En-De^B	73.9	47.0	90.5	75.0/83.4	89.6/86.1	84.1/83.9	81.8	83.8	54.9	**56.3**
MT En-Ru^B	74.3	52.4	89.9	71.8/81.3	89.4/85.6	82.8/82.8	81.5	83.1	58.5	43.7*
Reddit^B	75.6	49.5	91.7	84.6/89.2	89.4/85.8	83.8/83.6	81.8	84.4	58.1	**56.3**
SkipThought^B	75.2	53.9	90.8	78.7/85.2	89.7/86.3	81.2/81.5	82.2	84.6	57.4	43.7*
MTL GLUE^B	**79.6**	56.8	91.3	88.0/91.4	90.3/86.9	**89.2/89.0**	83.0	86.8	74.7	43.7*
MTL Non-GLUE^B	76.7	54.8	91.1	83.6/88.7	89.2/85.6	83.2/83.2	82.4	84.4	64.3	43.7*
MTL All^B	79.3	53.1	91.7	88.0/91.3	90.4/87.0	88.1/87.9	83.5	87.6	75.1	45.1*
Test Set Results										
Non-GLUE^E	69.7	34.5	89.5	78.2/84.8	83.6/64.3	77.5/76.0	75.4	74.8	55.6	65.1
MNLI^B	77.1	49.6	93.2	88.5/84.7	70.6/88.3	86.0/85.5	82.7	78.7	72.6	65.1
GLUE^B	77.3	49.0	93.5	89.0/85.3	70.6/88.6	85.8/84.9	82.9	81.0	71.7	34.9
BERT Base	78.4	52.1	93.5	88.9/84.8	71.2/89.2	87.1/85.8	84.0	91.1	66.4	65.1

Table 2: Results for **intermediate training** experiments on development sets except where noted. ^{E} and ^{B} respectively denote ELMo and BERT experiments. **Bold** denotes best scores by section. Underlining denotes average scores better than the single-task baseline. See Section 6 for discussion of WNLI results (*). BERT Base numbers are from Devlin et al. (2019).

$\sigma = 0.4$ for the random encoder with no pretraining, $\sigma = 0.2$ for ELMo with intermediate MNLI training, and $\sigma = 0.5$ for BERT without intermediate training. This variation is substantial but many of our results surpass a standard deviation of our baselines.

The WNLI dataset is adversarial: the same hypotheses can be paired with different premises and opposite labels in the train and dev sets, so models that overfit the train set will have low dev performance. As a result, few of our models reached even the *most frequent class* performance (56.3). When evaluating models that do worse than this, we replace their predictions with the most frequent label to simulate the performance achieved by not modeling the task.

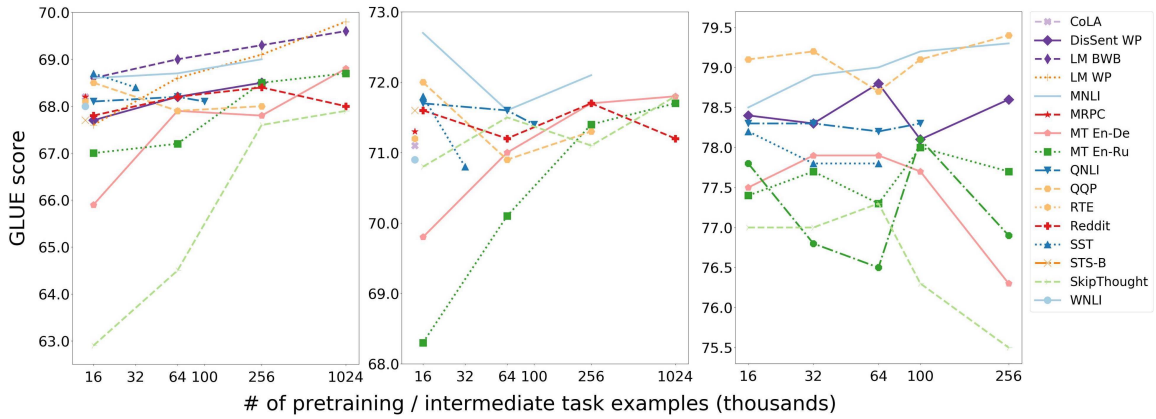

Figure 2: Learning curves (log scale) showing overall GLUE scores for encoders pretrained to convergence with varying amounts of data, shown for pretraining (left) and intermediate ELMo (center) and BERT (right) training.

6.1 Pretraining

From Table 1, we find the grammar-related CoLA task benefits dramatically from LM pretraining: The results achieved with LM pretraining are significantly better than the results achieved without. In contrast, the meaning-oriented STS sees good results with several kinds of pretraining, but does not benefit substantially from LM pretraining.

Among pretraining tasks, language modeling performs best, followed by MNLI. The remaining pretraining tasks yield performance near that of the random baseline. Even our single-task baseline gets less than a one point gain over this simple baseline. The multitask models are tied or outperformed by models trained on one of their constituent tasks, suggesting that our approach to multitask learning does not reliably produce models that productively combine the knowledge taught by each task. However, of the two models that perform best on the development data, the multitask model generalizes better than the single-task model on test data for tasks like STS and MNLI where the test set contains out-of-domain data.

Intermediate Task Training Looking to Table 2, using ELMo uniformly improves over training the encoder from scratch. The ELMo-augmented random baseline is strong, lagging behind the single-task baseline by less than a point. Most intermediate tasks beat the random baseline, but several fail to significantly outperform the single-task baseline. MNLI and English–German translation perform best with ELMo, with SkipThought and DisSent also beating the single-task baseline. Intermediate multitask training on all the non-GLUE tasks produces our best-performing ELMo model.

Using BERT consistently outperforms ELMo and pretraining from scratch. We find that intermediate training on each of MNLI, QQP, and STS leads to improvements over no intermediate training, while intermediate training on the other tasks harms transfer performance. The improvements gained via STS, a small-data task, versus the negative impact of fairly large-data tasks (e.g. QNLI), suggests that the benefit of intermediate training is not solely due to additional training, but that the signal provided by the intermediate task complements the original language modeling objective. Intermediate training on generation tasks such as MT significantly impairs BERT's transfer ability. We speculate that this may be due to catastrophic forgetting in fine-tuning for a generative task rather than a discriminative task like BERT was originally trained on. This phenomenon might be mitigated in our ELMo models via the frozen encoder and skip connection. On the test set, we lag slightly behind the BERT base results from Devlin et al. (2019), likely due in part to our limited hyperparameter tuning.

7 Analysis and Discussion

Target Task Correlations Table 3 presents an alternative view of the results of the pretraining experiment (Table 1): The table shows correlations between pairs of target tasks over the space of pretrained encoders. The correlations reflect the degree to which the performance on one target task with some encoder predicts performance on another target task with the same encoder. See Appendix D for the full table and similar tables for intermediate ELMo and BERT experiments.

Many correlations are low, suggesting that dif-

Task	Avg	CoLA	SST	STS	QQP	MNLI	QNLI
CoLA	0.86	1.00					
SST	0.60	0.25	1.00				
MRPC	0.39	0.21	0.34				
STS	-0.36	-0.60	0.01	1.00			
QQP	0.61	0.61	0.27	-0.58	1.00		
MNLI	0.54	0.16	0.66	0.40	0.08	1.00	
QNLI	0.43	0.13	0.26	0.04	0.27	0.56	1.00
RTE	0.34	0.08	0.16	-0.10	0.04	0.14	0.32
WNLI	-0.21	-0.21	-0.37	0.31	-0.37	-0.07	-0.26

Table 3: Pearson correlations between performances on a subset of all pairs of target tasks, measured over all runs reported in Table 1. The *Avg* column shows the correlation between performance on a target task and the overall GLUE score. For QQP and STS, the correlations are computed respectively using F1 and Pearson correlation. Negative correlations are underlined.

ferent tasks benefit from different forms of pretraining to a substantial degree, and bolstering the observation that no single pretraining task yields good performance on all target tasks. As noted above, the models that tended to perform best overall also overfit the WNLI training set most, leading to a negative correlation between WNLI and overall GLUE score. STS also shows a negative correlation, likely due to the observation that it does not benefit from LM pretraining. In contrast, CoLA shows a strong correlation with the overall GLUE scores, but has weak or negative correlations with many tasks: The use of LM pretraining dramatically improves CoLA performance, but most other forms of pretraining have little effect.

Learning Curves Figure 2 shows performance on the overall GLUE metric for encoders pretrained to convergence on each task with varying amounts of data. Looking at pretraining tasks in isolation (left), most tasks improve slightly as the amount of data increases, with the LM and MT tasks showing the most promising combination of slope and maximum performance. Combining these tasks with ELMo (center) or BERT (right) yields less interpretable results: the relationship between training data volume and performance becomes weaker, and some of the best results reported in this paper are achieved by models that combine ELMo with *restricted-data versions* of intermediate tasks like MNLI and QQP. This effect is amplified with BERT, with training data volume having unclear or negative relationships with performance for many tasks. With large datasets for generation tasks, we see clear evidence of catastrophic forgetting with performance sharply decreasing in amount of training data.

A second set of learning curves in Appendix E focuses on three fully pretrained encoders and measures performance on each GLUE target task separately with varying amounts of target task data. We see that all tasks benefit from increasing data quantities, with no obvious diminishing returns, and that most tasks see a consistent improvement in performance across data volumes from the use of pretraining, either with ELMo or with multitask learning.

Results on the GLUE Diagnostic Set On GLUE's analysis dataset, we find that many of our pretraining tasks help on examples involving lexical-semantic knowledge and logical operations, but less so on examples that highlight world knowledge. See Appendix F for details.

8 Conclusions

We present a systematic comparison of tasks and task combinations for the pretraining and intermediate fine-tuning of sentence-level encoders like those seen in ELMo and BERT. With nearly 60 pretraining tasks and task combinations and nine target tasks, this represents a far more comprehensive study than any seen on this problem to date.

Our primary results are perhaps unsurprising: LM works well as a pretraining task, and no other single task is consistently better. Intermediate training of language models can yield modest further gains. Multitask pretraining can produce results better than any single task can. Target task performance continues to improve with more LM data, even at large scales, suggesting that further work scaling up LM pretraining is warranted.

We also observe several worrying trends. Target tasks differ significantly in the pretraining tasks they benefit from, with correlations between target tasks often low or negative. Multitask pretraining fails to reliably produce models better than their best individual components. When trained on intermediate tasks like MT that are highly different than its original training task, BERT shows signs of catastrophic forgetting. These trends suggest that improving on LM pretraining with current techniques will be challenging.

While further work on language modeling seems straightforward and worthwhile, we believe that the future of this line of work will require a better understanding of the settings in which target

task models can effectively utilize outside knowledge and data, and new methods for pretraining and transfer learning to do so.

Acknowledgments

This work was conducted as part of the Fifth Frederick Jelinek Memorial Summer Workshop (JSALT) at Johns Hopkins University, and benefited from support by the JSALT sponsors and a team-specific donation of computing resources from Google. We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan V GPU used at NYU for this research. AW acknowledges support from an NSF Graduate Fellowship.

References

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, and Julia Kreutzer. 2017. Proceedings of the second conference on machine translation. In *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58. Association for Computational Linguistics.

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 *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

Soravit Changpinyo, Hexiang Hu, and Fei Sha. 2018. Multi-task learning for sequence tagging: An empirical study. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2965–2977, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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 1312.3005*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

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, Copenhagen, Denmark, September 9-11, 2017*, pages 681–691.

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.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc.

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, Volume 1 (Long Papers)*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

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. *arXiv preprint 1803.07640*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, Will Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian

Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic Chinese to English news translation.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. *arXiv preprint 1902.00751*.

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 (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Yacine Jernite, Samuel R. Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint 1705.00557*.

Douwe Kiela, Alexis Conneau, Allan Jabri, and Maximilian Nickel. 2018. Learning visually grounded sentence representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 408–418. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

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.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint 1901.11504*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136, Berlin, Germany. Association for Computational Linguistics.

Allen Nie, Erin D Bennett, and Noah D Goodman. 2017. DisSent: Sentence representation learning from explicit discourse relations. *arXiv preprint 1710.04334*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jason Phang, Thibault Fvry, and Samuel R. Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint 1811.01088*.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Towards a unified natural language inference framework to evaluate sentence representations. *arXiv preprint 1804.08207*.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Improving language understanding by generative pre-training. Unpublished manuscript accessible via the OpenAI Blog.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you dont know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

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

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of Adam and beyond. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. *arXiv preprint 1902.02671*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia de Sa. 2017. Rethinking Skip-thought: A neighborhood based approach. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 211–218. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *Proceedings of the International Conference on Learning Representations (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 *Advances in Neural Information Processing Systems*, pages 6000–6010.

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

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 *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 996–1005.

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint 1609.08144*.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-Yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-hsuan

Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 164–174, Melbourne, Australia. Association for Computational Linguistics.

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*.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. *Transactions of the Association of Computational Linguistics*, 5(1):379–395.

	Small	**Medium**	**Large**
Steps btw. validations	100	100	1000
Attention	N	N	Y
Classifier dropout rate	0.4	0.2	0.2
Classifier hidden dim.	128	256	512
Max pool projection dim.	128	256	512

Table 4: Hyperparameter settings for target-task models and target-task training for ELMo-style models. Small-data tasks are RTE and WNLI; medium-data tasks are CoLA, SST, and MRPC; large-data tasks are STS, QQP, MNLI, and QNLI. STS has a relatively small training set, but consistently patterns with the larger tasks in its behavior.

A Additional Pretraining Task Details

DisSent To extract discourse model examples from the WikiText-103 corpus (Merity et al., 2017), we follow the procedure described in Nie et al. (2017) by extracting clause-pairs that follow specific dependency relationships within the corpus (see Figure 4 in Nie et al., 2017). We use the Stanford Parser (Chen and Manning, 2014) distributed in Stanford CoreNLP version 3.9.1 to identify the relevant dependency arcs.

Cross-Sentence Attention For MNLI, QQP, QNLI, and STS with ELMo-style models, we use an attention mechanism between all pairs of words representations, followed by a $512D \times 2$ BiLSTM with max-pooling over time, following the mechanism used in BiDAF (Seo et al., 2017).

Alternative Tasks Any large-scale comparison like the one attempted in this paper is inevitably incomplete. Among the thousands of publicly available NLP datasets, we also performed initial trial experiments on several datasets for which we were not able to reach development-set performance above that of the random encoder baseline in the pretraining or as an intermediate task with ELMo. These include image-caption matching with MSCOCO (Lin et al., 2014), following Kiela et al. (2018); the small-to-medium-data text-understanding tasks collected in NLI format by Poliak et al. (2018); ordinal common sense inference (Zhang et al., 2017); POS tagging on the Penn Treebank (Marcus et al., 1993); supertagging on CCGBank (Hockenmaier and Steedman, 2007); and a variant objective on our Reddit data, inspired by Yang et al. (2018), where the model is trained to select which of two candidate replies to a given comment is correct.

B Hyperparameters and Optimization Details

See Section 5 for general comments on hyperparameter tuning.

Validation We evaluate on the validation set for the current training task or tasks every 1,000 steps, except where noted otherwise for small-data target tasks. During multitask learning, we multiply this interval by the number of tasks, evaluating every 9,000 steps during GLUE multitask training, for example.

Optimizer For BERT, we use the same optimizer and learning rate schedule as Devlin et al. (2019), with an initial learning rate of 1e-5 and training for a maximum of three epochs at each stage (or earlier if we trigger a different early stopping criterion). For all other experiments, we use AMSGrad (Reddi et al., 2018). During pretraining, we use a learning rate of 1e-4 for classification and regression tasks, and 1e-3 for text generation tasks. During target-task training, we use a learning rate of 3e-4 for all tasks.

Learning Rate Decay We multiply the learning rate by 0.5 whenever validation performance fails to improve for more than 4 validation checks. We stop training if the learning rate falls below 1e-6.

Early Stopping We maintain a saved checkpoint reflecting the best validation result seen so far. We stop training if we see no improvement after more than 20 validation checks. After training, we use the last saved checkpoint.

Regularization For BERT models, we follow the original work. For non-BERT models, we apply dropout with a drop rate of 0.2 after the character CNN in pretraining experiments or after ELMo, after each LSTM layer, and after each MLP layer in the task-specific classifier or regressor. For small-data target tasks, we increase MLP dropout to 0.4 during target-task training.

Preprocessing For BERT, we follow (Devlin et al., 2019) and use the WordPiece (Wu et al., 2016) tokenizer. For all other experiments, we use the Moses tokenizer for encoder inputs, and set a maximum sequence length of 40 tokens. There is no input vocabulary, as we use ELMo's character-based input layer.

For English text generation tasks, we use the Moses tokenizer to tokenize our data, but use a word-level output vocabulary of 20,000 types for tasks that require text generation. For translation tasks, we use BPE tokenization with a vocabulary of 20,000 types. For all sequence-to-sequence tasks we train word embeddings on the decoder side.

Target-Task-Specific Parameters For non-BERT models, to ensure that baseline performance for each target task is competitive, we find it necessary to use slightly different models and training regimes for larger and smaller target tasks. We used partially-heuristic tuning to separate GLUE tasks into big-, medium- and small-data groups, giving each group its own heuristically chosen task-specific model specifications. Exact values are shown in Table 4.

Sequence-to-Sequence Models We found attention to be helpful for the SkipThought and Reddit pretraining tasks but not for machine translation, and report results for these configurations. We use the max-pooled output of the encoder to initialize the hidden state of the decoder, and the size of this hidden state is equal to the size of the output of our shared encoder. We reduce the dimension of the output of the decoder by half via a linear projection before the output softmax layer.

C Multitask Learning Methods

Our general approach to multitask learning is as follows: At each update when training on multiple tasks, we randomly sample a single task with probability proportional to its training data size raised to the power of 0.75. This sampling rate is meant to balance the risks of overfitting small-data tasks and underfitting large ones, and performed best in early experiments. More extensive experiments with methods like this are shown in Appendix C. We perform early stopping based on an average of the tasks' validation metrics.

Our multitask learning experiments have three somewhat distinctive properties: (i) We mix tasks with very different amounts of training data—at the extreme, under 1,000 examples for WNLI, and over 1,000,000,000 examples from LM BWB. (ii) Our goal is to optimize the quality of the shared encoder, not the performance of any one of the tasks in the multitask mix. (iii) We mix a relatively large number of tasks, up to eighteen at once in some conditions. These conditions make it challenging but important to avoid overfitting or un-

	Pretraining Tasks			
Sampling	**GLUE**	**S1**	**S2**	**S3**
Uniform	69.1	53.7	82.1	31.7
Proportional	**69.8**	52.0	83.1	36.6
Log Proportional	68.8	54.3	82.9	31.2
Power 0.75	69.3	51.1	82.7	**37.9**
Power 0.66	69.0	53.4	82.8	35.5
Power 0.5	69.1	**55.6**	**83.3**	35.9

Table 5: Comparison of sampling methods on four subsets of GLUE using uniform loss scaling. The reported scores are averages of the development set results achieved for each task after early stopping. Results in **bold** are the best within each set.

	Loss Scaling		
Sampling	**Uniform**	**Proportional**	**Power 0.75**
Uniform	69.1	**69.7**	**69.8**
Proportional	**69.8**	69.4	69.6
Log Proportional	68.8	68.9	68.9
Power 0.75	69.3	69.1	69.0

Table 6: Combinations of sampling and loss scaling methods on GLUE tasks. Results in **bold** are tied for best overall GLUE score.

derfitting any of our tasks.

Relatively little work has been done on this problem, so we conduct a small experiment here. All our experiments use the basic paradigm of randomly sampling a new task to train on at each step, and we experiment with two hyperparameters that can be used to control over- and underfitting: The probability with which we sample each task and the weight with which we scale the loss for each task. Our experiments follow the setup in Appendix B, and do not use the ELMo BiLSTM. For validation metrics like perplexity that decrease from high starting values during training, we include the transformed metric $1 - \frac{metric}{250}$ in our average, where the constant 250 was tuned in early experiments.

Task Sampling We consider several approaches to determine the probability with which to sample a task during training, generally making this probability a function of the amount of data available for the task. For task i with training set size N_i, the probability is $p_i = f(N_i)/\sum_j f(N_j)$, where $f(N_i) = 1$ (Uniform), N_i (Proportional), $log(N_i)$ (Log Proportional), or N_i^a (Power a) where a is a constant.

Loss Scaling At each update, we scale the loss of a task with weight $w_i = f(N_i)/max_j f(N_j)$, where $f(N_i) = 1$ (Uniform), N_j (Proportional), or N_j^a (Power a).

Experiments For task sampling, we run experiments with multitask learning on the full set of nine GLUE tasks, as well as three subsets: single sentence tasks (S1: SST, CoLA), similarity and paraphrase tasks (S2: MRPC, STS, QQP), and inference tasks (S3: WNLI, QNLI, MNLI, RTE). The results are shown in Table 5.

We also experiment with several *combinations* of task sampling and loss scaling methods, using only the full set of GLUE tasks. The results are shown in Table 6.

While no combination of methods consistently offers dramatically better performance than any other, we observe that it is generally better to apply only one of non-uniform sampling and non-uniform loss scaling at a time rather than apply both simultaneously, as they provide roughly the same effect. Following encouraging results from earlier pilot experiments, we use power 0.75 task sampling and uniform loss scaling in the multitask learning experiments shown in Table 1.

D Additional Target Task Correlations

Tables 7, 8, and 9 respectively show the full target task correlations computed over pretraining, intermediate ELMo, and intermediate BERT experiments.

See Section 7 for a discussion about correlations for the pretraining experiments. The general trends in correlation vary significantly between the three experimental settings, which we take to roughly indicate the different types of knowledge encoded in ELMo and BERT. The exception is that WNLI is consistently negatively correlated with the other target tasks and often the overall GLUE score.

For intermediate ELMo experiments, correlations are generally low, with the exception of MNLI with other tasks. CoLA is negatively correlated with most other tasks, while QQP and SST are positively correlated with most tasks.

For intermediate BERT experiments, correlations with the GLUE score are quite high, as we found that intermediate training often negatively impacted GLUE score. QQP is highly negatively correlated with most other tasks, while the smaller tasks like MRPC and RTE are most highly correlated with overall GLUE score.

E Additional Learning Curves

Figure 3 shows learning curves reflecting the amount of target-task data required to train a model on each GLUE task, starting from three selected encoders. See Section 7 for discussion.

F Diagnostic Set Results

Tables 10 and 11 show results on the four coarse-grained categories of the GLUE diagnostic set for all our pretraining experiments. This set consists of about 1000 expert-constructed examples in NLI format meant to isolate a range of relevant phenomena. Results use the target task classifier trained on the MNLI training set.

No model achieves performance anywhere close to human-level performance, suggesting that *either* none of our pretrained models extract features that are suitable for robust reasoning over text, or that the MNLI training set and the MNLI target-task model are not able to exploit any such features that exist. See Section 7 for further discussion.

While no model achieves near-human performance, the use of ELMo and appears to be helpful on examples that highlight world knowledge and lexical-semantic knowledge, and less so on examples that highlight complex logical reasoning patterns or alternations in sentence structure. This relative weakness on sentence structure is somewhat surprising given the finding in Zhang and Bowman (2018) that language model pretraining is helpful for tasks involving sentence structure.

Using BERT helps significantly with understanding sentence structure, lexical knowledge, and logical reasoning, but does not seem to help on world knowledge over using ELMo. Encouragingly, we find that intermediate training of BERT on all of our pretraining tasks outperforms intermediate training on one or no tasks in two of the four categories.

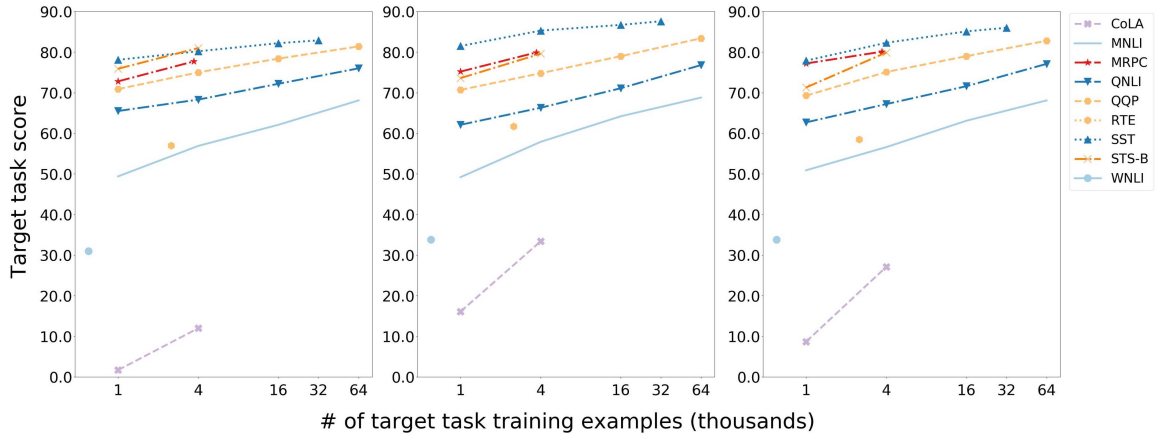

Figure 3: Target-task training learning curves for each GLUE task with three encoders: the random encoder without ELMo (left), random with ELMo (center), and MTL Non-GLUE pretraining (right).

Task	Avg	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI
CoLA	0.86	1.00								
SST	0.60	0.25	1.00							
MRPC	0.39	0.21	0.34	1.00						
STS	-0.36	-0.60	0.01	0.29	1.00					
QQP	0.61	0.61	0.27	-0.17	-0.58	1.00				
MNLI	0.54	0.16	0.66	0.56	0.40	0.08	1.00			
QNLI	0.43	0.13	0.26	0.32	0.04	0.27	0.56	1.00		
RTE	0.34	0.08	0.16	-0.09	-0.10	0.04	0.14	0.32	1.00	
WNLI	-0.21	-0.21	-0.37	0.31	0.31	-0.37	-0.07	-0.26	0.12	1.00

Table 7: Pearson correlations between performances on different target tasks, measured over all pretraining runs reported in Table 1.

Task	Avg	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI
CoLA	0.07	1.00								
SST	0.32	-0.48	1.00							
MRPC	0.42	-0.20	0.29	1.00						
STS	0.41	-0.40	0.26	0.21	1.00					
QQP	0.02	0.08	0.26	0.18	0.15	1.00				
MNLI	0.60	-0.21	0.33	0.38	0.72	0.21	1.00			
QNLI	0.50	0.10	0.03	0.12	0.63	-0.01	0.72	1.00		
RTE	0.39	-0.13	-0.15	0.21	0.27	-0.04	0.60	0.59	1.00	
WNLI	-0.14	0.02	0.23	-0.29	-0.02	0.15	0.02	-0.25	-0.22	1.00

Table 8: Pearson correlations between performances on different target tasks, measured over all ELMo runs reported in Table 2. Negative correlations are underlined.

Task	Avg	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI
CoLA	0.71	1.00								
SST	0.41	0.32	1.00							
MRPC	0.83	0.67	0.62	1.00						
STS	0.82	0.34	0.21	0.60	1.00					
QQP	-0.41	0.01	0.04	-0.05	-0.64	1.00				
MNLI	0.73	0.31	0.10	0.42	0.69	-0.68	1.00			
QNLI	0.73	0.38	0.29	0.56	0.43	-0.11	0.62	1.00		
RTE	0.88	0.47	0.22	0.56	0.87	-0.70	0.68	0.55	1.00	
WNLI	0.45	-0.10	-0.03	0.20	0.79	-0.89	0.65	0.11	0.69	1.00

Table 9: Pearson correlations between performances on different target tasks, measured over all BERT runs reported in Table 2. Negative correlations ad underlined.

Pretr.	Knowledge	Lexical Semantics	Logic	Predicate/Argument Str.
		Baselines		
Random	17.6	19.6	12.5	26.9
		GLUE Tasks as Pretraining Tasks		
CoLA	15.3	24.2	14.9	**31.7**
SST	16.1	24.8	16.5	28.7
MRPC	16.0	**25.2**	12.6	26.4
QQP	12.8	22.5	12.9	30.8
STS	16.5	20.2	13.0	27.1
MNLI	16.4	20.4	**17.7**	29.9
QNLI	13.6	21.3	12.2	28.0
RTE	16.3	23.1	14.5	28.8
WNLI	**18.8**	19.5	13.9	29.1
		Non-GLUE Pretraining Tasks		
DisSent WP	18.5	24.2	15.4	27.8
LM WP	14.9	16.6	9.4	23.0
LM BWB	15.8	19.4	9.1	23.9
MT En-De	13.4	24.6	14.8	30.1
MT En-Ru	13.4	24.6	14.8	30.1
Reddit	13.9	20.4	14.1	26.0
SkipThought	15.1	22.0	13.7	27.9
		Multitask Pretraining		
MTL All	16.3	21.4	11.2	28.0
MTL GLUE	12.5	21.4	15.0	30.1
MTL Outside	14.5	19.7	13.1	26.2

Table 10: GLUE diagnostic set results, reported as R_3 correlation coefficients ($\times 100$), which standardizes the score of random guessing by an uninformed model at roughly 0. Human performance on the overall diagnostic set is roughly 80. Results in **bold** are the best overall.

Pretr.	Knowledge	Lexical Semantics	Logic	Predicate/Argument Str.
ELMo with Intermediate Task Training				
Random^E	19.2	22.9	9.8	25.5
CoLA^E	17.2	21.6	9.2	27.3
SST^E	19.4	20.5	9.7	28.5
MRPC^E	11.8	20.5	12.1	27.4
QQP^E	17.5	16.0	9.9	**30.5**
STS^E	18.0	18.4	9.1	25.5
MNLI^E	17.0	23.2	14.4	23.9
QNLI^E	17.4	**24.1**	10.7	30.2
RTE^E	18.0	20.2	8.7	28.0
WNLI^E	16.5	19.8	7.3	25.2
DisSent WP^E	16.3	23.0	11.6	26.5
MT En-De^E	19.2	21.0	13.5	29.7
MT En-Ru^E	20.0	20.1	11.9	21.4
Reddit^E	14.7	22.3	**15.0**	29.0
SkipThought^E	20.5	18.5	10.4	26.8
MTL GLUE^E	**20.6**	22.1	14.7	25.3
MTL Non-GLUE^E	15.7	23.7	12.6	29.0
MTL All^E	13.8	18.4	10.8	26.7
BERT with Intermediate Task Training				
Single-Task^B	20.3	36.3	21.7	40.4
CoLA^B	18.5	34.0	23.5	40.1
SST^B	19.8	36.0	23.2	39.1
MRPC^B	20.6	33.3	20.9	37.8
QQP^B	17.4	35.7	23.8	40.5
STS^B	**21.3**	34.7	24.0	40.7
MNLI^B	19.1	34.0	23.3	**41.7**
QNLI^B	20.3	38.4	24.4	41.5
RTE^B	15.4	32.6	20.2	38.5
WNLI^B	20.8	35.8	23.1	39.3
DisSent WP^B	17.9	34.0	23.7	39.1
MT En-De^B	18.6	33.8	20.7	37.4
MT En-Ru^B	14.2	30.2	20.3	36.5
Reddit^B	16.5	29.9	22.7	37.1
SkipThought^B	15.8	35.0	20.9	38.3
MTL GLUE^B	17.0	35.2	24.3	39.6
MTL Non-GLUE^B	18.7	37.0	21.8	40.6
MTL All^B	17.8	**40.3**	**27.5**	41.0

Table 11: GLUE diagnostic set results, reported as R_3 correlation coefficients ($\times 100$), which standardizes the score of random guessing by an uninformed model at roughly 0. Human performance on the overall diagnostic set is roughly 80. Results in **bold** are the best by section.