To Tune or Not to Tune?
Adapting Pretrained Representations to Diverse Tasks

Matthew Peters[1]*, Sebastian Ruder[2,3]†*, and Noah A. Smith[1,4]

[1]Allen Institute for Artificial Intelligence, Seattle, USA
[2]Insight Research Centre, National University of Ireland, Galway, Ireland
[3]Aylien Ltd., Dublin, Ireland
[4]Paul G. Allen School of CSE, University of Washington, Seattle, USA
{matthewp,noah}@allenai.org, sebastian@ruder.io

Abstract

While most previous work has focused on different pretraining objectives and architectures for transfer learning, we ask how to best adapt the pretrained model to a given target task. We focus on the two most common forms of adaptation, feature extraction (where the pretrained weights are frozen), and directly fine-tuning the pretrained model. Our empirical results across diverse NLP tasks with two state-of-the-art models show that the relative performance of fine-tuning vs. feature extraction depends on the similarity of the pretraining and target tasks. We explore possible explanations for this finding and provide a set of adaptation guidelines for the NLP practitioner.

1 Introduction

Sequential inductive transfer learning (Pan and Yang, 2010) consists of two stages: *pretraining*, in which the model learns a general-purpose representation of inputs, and *adaptation*, in which the representation is transferred to a new task. Most previous work in NLP has focused on different pretraining objectives for learning word or sentence representations (Mikolov et al., 2013; Kiros et al., 2015).

Few works, however, have focused on the adaptation phase. There are two main paradigms for adaptation: *feature extraction* and *fine-tuning*. In feature extraction (❄) the model's weights are 'frozen' and the pretrained representations are used in a downstream model similar to classic feature-based approaches (Koehn et al., 2003). Alternatively, a pretrained model's parameters can be unfrozen and fine-tuned (🔥) on a new task (Dai and Le, 2015). Both have benefits: ❄ enables use of task-specific model architectures and may be

---

*The first two authors contributed equally.
†Sebastian is now at DeepMind.

Conditions			Guidelines
Pretrain	Adapt.	Task	
Any	❄	Any	Add many task parameters
Any	🔥	Any	Add minimal task parameters ⚠ Hyper-parameters
Any	Any	Seq. / clas.	❄ and 🔥 have similar performance
ELMo	Any	Sent. pair	use ❄
BERT	Any	Sent. pair	use 🔥

Table 1: This paper's guidelines for using feature extraction (❄) and fine-tuning (🔥) with ELMo and BERT. Seq.: sequence labeling. Clas.: classification. Sent. pair: sentence pair tasks.

computationally cheaper as features only need to be computed once. On the other hand, 🔥 is convenient as it may allow us to adapt a general-purpose representation to many different tasks.

Gaining a better understanding of the adaptation phase is key in making the most use out of pretrained representations. To this end, we compare two state-of-the-art pretrained models, ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) using both ❄ and 🔥 across seven diverse tasks including named entity recognition, natural language inference (NLI), and paraphrase detection. We seek to characterize the conditions under which one approach substantially outperforms the other, and whether it is dependent on the pretraining objective or target task. We find that ❄ and 🔥 have comparable performance in most cases, except when the source and target tasks are either highly similar or highly dissimilar. We furthermore shed light on the practical challenges of adaptation and provide a set of guidelines to the NLP practitioner, as summarized in Table 1.

2 Pretraining and Adaptation

While pretraining tasks have been designed with particular downstream tasks in mind (Felbo et al., 2017), we focus on pretraining tasks that seek to

induce *universal* representations suitable for any downstream task.

Word representations Pretrained word vectors (Turian et al., 2010; Pennington et al., 2014) have been an essential component in state-of-the-art NLP systems. Word representations are often fixed and fed into a task specific model (❄️), although 🔥 can provide improvements (Kim, 2014). Recently, contextual word representations learned supervisedly (e.g., through machine translation; McCann et al., 2017) or unsupervisedly (typically through language modeling; Peters et al., 2018) have significantly improved over noncontextual vectors.

Sentence embedding methods Such methods learn sentence representations via different pretraining objectives such as previous/next sentence prediction (Kiros et al., 2015; Logeswaran and Lee, 2018), NLI (Conneau et al., 2017), or a combination of objectives (Subramanian et al., 2018). During the adaptation phase, the sentence representation is typically provided as input to a linear classifier (❄️). LM pretraining with 🔥 has also been successfully applied to sentence level tasks. Howard and Ruder (2018, ULMFiT) propose techniques for fine-tuning a LM, including triangular learning rate schedules and discriminative fine-tuning, which uses lower learning rates for lower layers. Radford et al. (2018) extend LM-🔥 to additional sentence and sentence-pair tasks.

Masked LM and next-sentence prediction BERT (Devlin et al., 2018) combines both word and sentence representations (via masked LM and next sentence prediction objectives) in a single very large pretrained transformer (Vaswani et al., 2017). It is adapted to both word and sentence level tasks by 🔥 with task-specific layers.

3 Experimental Setup

We compare ELMo and BERT as representatives of the two best-performing pretraining settings. This section provides an overview of our methods; see the supplement for full details.

3.1 Target Tasks and Datasets

We evaluate on a diverse set of target tasks: named entity recognition (NER), sentiment analysis (SA), and three sentence pair tasks, natural language inference (NLI), paraphrase detection (PD), and semantic textual similarity (STS).

NER We use the CoNLL 2003 dataset (Sang and Meulder, 2003), which provides token level annotations of newswire across four different entity types (PER, LOC, ORG, MISC).

SA We use the binary version of the Stanford Sentiment Treebank (SST-2; Socher et al., 2013), providing sentiment labels (negative or positive) for phrases and sentences of movie reviews.

NLI We use both the broad-domain MultiNLI dataset (Williams et al., 2018) and Sentences Involving Compositional Knowledge (SICK-E; Marelli et al., 2014).

PD For paraphrase detection (i.e., decide whether two sentences are semantically equivalent), we use the Microsoft Research Paraphrase Corpus (MRPC; Dolan and Brockett, 2005).

STS We employ the Semantic Textual Similarity Benchmark (STS-B; Cer et al., 2017) and SICK-R (Marelli et al., 2014). Both datasets, provide a human judged similarity value from 1 to 5 for each sentence pair.

3.2 Adaptation

We now describe how we adapt ELMo and BERT to these tasks. For ❄️ we require a task-specific architecture, while for 🔥 we need a task-specific output layer. For fair comparison, we conduct an extensive hyper-parameter search for each task.

Feature extraction (❄️) For both ELMo and BERT, we extract contextual representations of the words from all layers. During adaptation, we learn a linear weighted combination of the layers (Peters et al., 2018) which is used as input to a task-specific model. When extracting features, it is important to expose the internal layers as they typically encode the most transferable representations. For SA, we employ a bi-attentive classification network (McCann et al., 2017). For the sentence pair tasks, we use the ESIM model (Chen et al., 2017). For NER, we use a BiLSTM with a CRF layer (Lafferty et al., 2001; Lample et al., 2016).

Fine-tuning (🔥): ELMo We max-pool over the LM states and add a softmax layer for text classification. For the sentence pair tasks, we compute cross-sentence bi-attention between the LM states (Chen et al., 2017), apply a pooling operation, then add a softmax layer. For NER, we add a CRF layer on top of the LSTM states.

Pretraining	Adaptation	NER CoNLL 2003	SA SST-2	Nat. lang. inference		Semantic textual similarity		
				MNLI	SICK-E	SICK-R	MRPC	STS-B
Skip-thoughts	❄	-	81.8	62.9	-	**86.6**	75.8	71.8
ELMo	❄	91.7	**91.8**	**79.6**	**86.3**	**86.1**	**76.0**	**75.9**
	🔥	**91.9**	91.2	76.4	83.3	83.3	74.7	75.5
	Δ=🔥-❄	0.2	-0.6	-3.2	-3.3	-2.8	-1.3	-0.4
BERT-base	❄	92.2	93.0	**84.6**	84.8	86.4	78.1	82.9
	🔥	**92.4**	**93.5**	**84.6**	**85.8**	**88.7**	**84.8**	**87.1**
	Δ=🔥-❄	0.2	0.5	0.0	1.0	2.3	6.7	4.2

Table 2: Test set performance of feature extraction (❄) and fine-tuning (🔥) approaches for ELMo and BERT-base compared to two sentence embedding methods. Settings that are good for 🔥 are colored in red (Δ=🔥-❄ > 1.0); settings good for ❄ are colored in blue (Δ=🔥-❄ < -1.0). Numbers for baseline methods are from respective papers, except for SST-2, MNLI, and STS-B results, which are from Wang et al. (2018). BERT fine-tuning results (except on SICK) are from Devlin et al. (2018). The metric varies across tasks (higher is always better): accuracy for SST-2, SICK-E, and MRPC; matched accuracy for MultiNLI; Pearson correlation for STS-B and SICK-R; and span F_1 for CoNLL 2003. For CoNLL 2003, we report the mean with five seeds; standard deviation is about 0.2%.

Fine-tuning (🔥): BERT We feed the sentence representation into a softmax layer for text classification and sentence pair tasks following Devlin et al. (2018). For NER, we extract the representation of the first word piece for each token and add a softmax layer.

4 Results

We show results in Table 2 comparing ELMo and BERT for both ❄ and 🔥 approaches across the seven tasks with one sentence embedding method, Skip-thoughts (Kiros et al., 2015), that employs a next-sentence prediction objective similar to BERT.

Both ELMo and BERT outperform the sentence embedding method significantly, except on the semantic textual similarity tasks (STS) where Skip-thoughts is similar to ELMo. The overall performance of ❄ and 🔥 varies from task to task, with small differences except for a few notable cases. For ELMo, we find the largest differences for sentence pair tasks where ❄ consistently outperforms 🔥. For BERT, we obtain nearly the opposite result: 🔥 significantly outperforms ❄ on all STS tasks, with much smaller differences for the others.

Discussion Past work in NLP (Mou et al., 2016) showed that similar pretraining tasks transfer better.[1] In computer vision (CV), Yosinski et al. (2014) similarly found that the transferability of

features decreases as the distance between the pre-training and target task increases. In this vein, Skip-thoughts—and Quick-thoughts (Logeswaran and Lee, 2018), which has similar performance—which use a next-sentence prediction objective similar to BERT, perform particularly well on STS tasks, indicating a close alignment between the pretraining and target task. This strong alignment also seems to be the reason for BERT's strong relative performance on these tasks.

In CV, 🔥 generally outperforms ❄ when transferring from ImageNet supervised classification pretraining to other classification tasks (Kornblith et al., 2018). Recent results suggest 🔥 is less useful for more distant target tasks such as semantic segmentation (He et al., 2018). This is in line with our results, which show strong performance with 🔥 between closely aligned tasks (next-sentence prediction in BERT and STS tasks) and poor performance for more distant tasks (LM in ELMo and sentence pair tasks). A confounding factor may be the suitability of the inductive bias of the model architecture for sentence pair tasks, which we will analyze next.

5 Analyses

Modelling pairwise interactions LSTMs consider each token sequentially, while Transformers can relate each token to every other in each layer (Vaswani et al., 2017). This might facilitate 🔥 with Transformers on sentence pair tasks, on which ELMo-🔥 performs comparatively poorly.

---

[1]Mou et al. (2016), however, only investigate transfer between classification tasks (NLI → SICK-E/MRPC).

	SICK-E	SICK-R	STS-B	MRPC
ELMo-🔥 +bi-attn.	83.8	84.0	80.2	77.0
w/o bi-attn.	70.9	51.8	38.5	72.3

Table 3: Comparison of ELMO-🔥 cross-sentence embedding methods on dev. sets of sentence pair tasks.

	SICK-E	SICK-R	STS-B	MRPC
BERT-❄️, joint enc.	**85.5**	86.4	**88.1**	**83.3**
separate encoding	81.2	**86.8**	86.8	81.4

Table 4: Comparison of BERT-❄️ cross-sentence embedding methods on dev. sets of sentence pair tasks.

Model configuration	F_1
❄️ + BiLSTM + CRF	**95.5**
❄️ + CRF	91.9
🔥 + CRF + gradual unfreeze	**95.5**
🔥 + BiLSTM + CRF + gradual unfreeze	95.2
🔥 + CRF	95.1

Table 5: Comparison of CoNLL 2003 NER development set performance (F_1) for ELMo for both feature extraction and fine-tuning. All results averaged over five random seeds.

	TE	GO	TR	FI	SL
BERT-❄️	84.4	86.7	86.1	84.5	80.9
Δ=🔥-❄️	-1.1	-0.2	-0.6	0.4	-0.6
JS div	0.21	0.18	0.14	0.09	0.09

Table 6: Accuracy of feature extraction (❄️) and fine-tuning (🔥) with BERT-base trained on training data of different MNLI domains and evaluated on corresponding dev sets. TE: telephone. FI: fiction. TR: travel. GO: government. SL: slate.

To analyze this further, we compare different ways of encoding the sentence pair with ELMo and BERT. For ELMo, we compare encoding with and without cross-sentence bi-attention in Table 3. When adapting the ELMo LSTM to a sentence pair task, modeling the sentence interactions by fine-tuning through the bi-attention mechanism provides the best performance.[2] This provides further evidence that the LSTM has difficulty modeling the pairwise interactions during sequential processing. This is in contrast to a Transformer LM that can be fine-tuned in this manner (Radford et al., 2018).

For BERT-❄️, we compare joint encoding of the sentence pair with encoding the sentences separately in Table 4. The latter leads to a drop in performance, which shows that the BERT representations encode cross-sentence relationships and are therefore particularly well-suited for sentence pair tasks.

Impact of additional parameters We evaluate whether adding parameters is useful for both adaptation settings on NER. We add a CRF layer (as used in 🔥) and a BiLSTM with a CRF layer (as used in ❄️) to both and show results in Table 5. We find that additional parameters are key for ❄️, but hurt performance with 🔥. In addition, 🔥 requires gradual unfreezing (Howard and Ruder, 2018) to match performance of feature extraction.

ELMo fine-tuning We found fine-tuning the ELMo LSTM to be initially difficult and required careful hyper-parameter tuning. Once tuned for one task, other tasks have similar hyper-parameters. Our best models used slanted trian-

gular learning rates and discriminative fine-tuning (Howard and Ruder, 2018) and in some cases gradual unfreezing.

Impact of target domain Pretrained language model representations are intended to be universal. However, the target domain might still impact the adaptation performance. We calculate the Jensen-Shannon divergence based on term distributions (Ruder and Plank, 2017) between the domains used to train BERT (books and Wikipedia) and each MNLI domain. We show results in Table 6. We find no significant correlation. At least for this task, the distance of the source and target domains does not seem to have a major impact on the adaptation performance.

Representations at different layers In addition, we are interested how the information in the different layers of the models develops over the course of fine-tuning. We measure this information in two ways: a) with diagnostic classifiers (Adi et al., 2017); and b) with mutual information (MI; Noshad et al., 2018). Both methods allow us to associate the hidden activations of our model with a linguistic property. In both cases, we use

---

[2]This is similar to text classification tasks, where we find max-pooling to outperform using the final hidden state, similar to (Howard and Ruder, 2018).

the mean of the hidden activations of BERT-base[3] of each token / word piece of the sequence(s) as the representation.[4]

With diagnostic classifiers, for each example, we extract the pretrained and fine-tuned representation at each layer as features. We use these features as input to train a logistic regression model (linear regression for STS-B, which has real-valued outputs) on the training data of two single sentence (CoLA[5] and SST-2) and two pair sentence tasks (MRPC and STS-B). We show its performance on the corresponding dev sets in Figure 1.
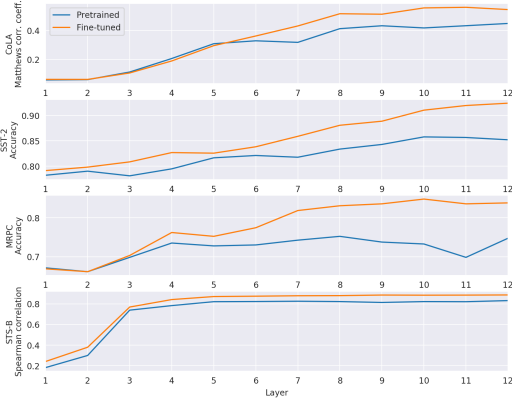

Figure 1: Performance of diagnostic classifiers trained on pretrained and fine-tuned BERT representations at different layers on the dev sets of the corresponding tasks.

For all tasks, diagnostic classifier performance generally is higher in higher layers of the model. Fine-tuning improves the performance of the diagnostic classifier at every layer. For the single sentence classification tasks CoLA and SST-2, pretrained performance increases gradually until the last layers. In contrast, for the sentence pair tasks MRPC and STS-B performance is mostly flat after the fourth layer. Relevant information for sentence pair tasks thus does not seem to be concentrated primarily in the upper layers of pretrained representations, which could explain why fine-tuning is particularly useful in these scenarios.

Computing the mutual information with regard to representations of deep neural networks has only become feasible recently with the development of more sophisticated MI estimators. In our experiments, we use the state-of-the-art ensemble dependency graph estimator (EDGE; Noshad et al., 2018) with default hyper-parameter values. As a sanity check, we compute the MI between hidden activations and random labels and random representations and random labels, which yields 0 in every case as we would expect.[6]

We show the mutual information $I(H;Y)$ between the pretrained and fine-tuned mean hidden activations H at each layer of BERT and the output labels Y on the dev sets of CoLA, SST-2, and MRPC in Figure 2.
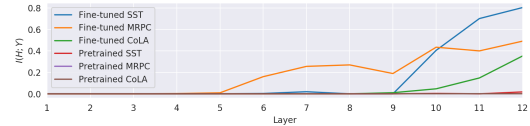

Figure 2: The mutual information between fine-tuned and pretrained mean BERT representations and the labels on the dev set of the corresponding tasks.

The MI between pretrained representations and labels is close to 0 across all tasks and layers, except for SST where the last layer shows a small non-zero value. In contrast, fine-tuned representations display much higher MI values. The MI for fine-tuned representations rises gradually through the intermediate and last layers for the sentence pair task MRPC, while for the single sentence classification tasks, the MI rises sharply in the last layers. Similar to our findings with diagnostic classifiers, knowledge for single sentence classification tasks thus seems mostly concentrated in the last layers, while pair sentence classification tasks gradually build up information in the intermediate and last layers of the model.

6 Conclusion

We have empirically analyzed fine-tuning and feature extraction approaches across diverse datasets, finding that the relative performance depends on the similarity of the pretraining and target tasks. We have explored possible explanations and provided practical recommendations for adapting pretrained representations to NLP practicioners.

---

[3] We show results for BERT as they are more inspectable due to the model having more layers. Trends for ELMo are similar.

[4] We observed similar results when using max-pooling or the representation of the first token.

[5] The Corpus of Linguistic Acceptability (CoLA) consists of examples of expert English sentence acceptability judgments drawn from 22 books and journal articles on linguistic theory. It uses the Matthews correlation coefficient (Matthews, 1975) for evaluation and is available at: nyu-mll.github.io/CoLA

[6] For the same settings, we obtain non-zero values with earlier estimators (Saxe et al., 2018), which seem to be less reliable for higher numbers of dimensions.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. In *Proceedings of ICLR 2017*.

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

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. In *Proceedings of ACL 2017*.

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*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

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

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

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.

Kaiming He, Ross Girshick, and Piotr Dollár. 2018. Rethinking ImageNet Pre-training. *arXiv preprint arXiv:1811.08883*.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of ACL 2018*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Proceedings of NIPS 2015*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Simon Kornblith, Jonathon Shlens, Quoc V Le, and Google Brain. 2018. Do Better ImageNet Models Transfer Better? *arXiv preprint arXiv:1805.08974*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *Proceedings of ICLR 2018*.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In *Advances in Neural Information Processing Systems*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications? *Proceedings of 2016 Conference on Empirical Methods in Natural Language Processing*.

Morteza Noshad, Yu Zeng, and Alfred O. Hero III. 2018. Scalable Mutual Information Estimation using Dependence Graphs. *arXiv preprint arXiv:1801.09125*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with Bayesian Optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. 2018. On the Information Bottleneck Theory of Deep Learning. In *Proceedings of ICLR 2018*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y 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 *Proceedings of ICLR 2018*.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz 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.

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

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *NIPS*.

A Experimental details

For fair comparison, all experiments include extensive hyper-parameter tuning. We tuned the learning rate, dropout ratio, weight decay and number of training epochs. In addition, the fine-tuning experiments also examined the impact of triangular learning rate schedules, gradual unfreezing, and discriminative learning rates. Hyperparameters were tuned on the development sets and the best setting evaluated on the test sets.

All models were optimized with the Adam optimizer (Kingma and Ba, 2015) with weight decay fix (Loshchilov and Hutter, 2017).

We used the publicly available pretrained ELMo[7] and BERT[8] models in all experiments. For ELMo, we used the original two layer bidirectional LM. In the case of BERT, we used the BERT-base model, a 12 layer bidirectional transformer. We used the English uncased model for all tasks except for NER which used the English cased model.

A.1 Feature extraction

To isolate the effects of fine-tuning contextual word representations, all feature based models only include one type of word representation (ELMo or BERT) and do not include any other pretrained word representations.

For all tasks, all layers of pretrained representations were weighted together with learned scalar parameters following Peters et al. (2018).

NER For the NER task, we use a two layer bidirectional LSTM in all experiments. For ELMo, the output layer is a CRF, similar to a state-of-the-art NER system (Lample et al., 2016). Feature extraction for ELMo treated each sentence independently.

In the case of BERT, the output layer is a softmax to be consistent with the fine-tuned experiments presented in Devlin et al. (2018). In addition, as in Devlin et al. (2018), we used document context to extract word piece representations. When composing multiple word pieces into a single word representation, we found it beneficial to run the biLSTM layers over all word pieces before taking the LSTM states of the first word piece in each word. We experimented with other pooling operations to combine word pieces into a

---

[7]https://allennlp.org/elmo
[8]https://github.com/google-research/bert

single word representation but they did not provide additional gains.

SA   We used the implementation of the bi-attentive classification network in AllenNLP (Gardner et al., 2017) with default hyper-parameters, except for tuning those noted above. As in the fine-tuning experiments for SST-2, we used all available annotations during training, including those of sub-trees. Evaluation on the development and test sets used full sentences.

Sentence pair tasks   When extracting features from ELMo, each sentence was handled separately. For BERT, we extracted features for both sentences jointly to be consistent with the pretraining procedure. As reported in Section 5 this improved performance over extracting features for each sentence separately.

Our model is the ESIM model (Chen et al., 2017), modified as needed to support regression tasks in addition to classification. We used default hyper-parameters except for those described above.

A.2   Fine-tuning

When fine-tuning ELMo, we found it beneficial to use discriminative learning rates (Howard and Ruder, 2018) where the learning rate decreased by $0.4\times$ in each layer (so that the learning rate for the second to last layer is $0.4\times$ the learning rate in the top layer). In addition, for SST-2 and NER, we also found it beneficial to gradually unfreeze the weights starting with the top layer. In this setting, in each epoch one additional layer of weights is unfrozen until all weights are training. These settings were chosen by tuning development set performance.

For fine-tuning BERT, we used the default learning rate schedule (Devlin et al., 2018) that is similar to the schedule used by Howard and Ruder (2018).

SA   We considered several pooling operations for composing the ELMo LSTM states into a vector for prediction including max pooling, average pooling and taking the first/last states. Max pooling performed slightly better than average pooling on the development set.

Sentence pair tasks   Our bi-attentive fine-tuning mechanism is similar to the the attention mechanism in the feature based ESIM model. To apply it, we first computed the bi-attention between all words in both sentences, then applied the same "enhanced" pooling operation as in (Chen et al., 2017) before predicting with a softmax. Note that this attention mechanism and pooling operation does not add any additional parameters to the network.