Evaluating fine-tuning methods for robust multi-task sentence embeddings

Stanford CS224N Default Project

Connor Toups
Department of Computer Science
Stanford University
ctoups22@stanford.edu

Kaleb Tsegay
Department of Computer Science
Stanford University
kbtsegay@stanford.edu

Ammar Alinur
Department of Computer Science
Stanford University
aalinur@stanford.edu

Abstract

In this paper, we evaluate diverse fine-tuning methods to generate high-quality embeddings that generalize well on three NLP tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We evaluate the impact of Gao et al. (2021)'s proposed SimCSE approach of intermediately fine-tuning on an NLI dataset to improve downstream performance and compare it to linear probing and fine-tuning only on the downstream tasks. Surprisingly, we find that intermediate fine-tuning harms performance both when linear probing and when fine-tuning on the target tasks. We offer three reasons for this degraded performance: the use of a learnable output layer diminishes the need for intermediate fine-tuning; multi-task settings have conflicting gradients that can lead to suboptimal convergence; and fine-tuning is sensitive to distributional shift between training stages. We also show that fine-tuning on sentiment analysis harms performance on semantic textual similarity datasets and, vice versa, fine-tuning on semantic textual similarity harms performance on sentiment analysis; however, fine-tuning on paraphrase detection doesn't harm performance on semantic textual similarity nor sentiment analysis datasets.

1 Key Information to include

- Mentor: Default Project
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Using a single model that has been adapted to perform on a variety of downstream tasks has a number of benefits over specialized models that can only perform one task: less total memory is required to store the parameters of a single compared to multiple models; less compute is required to train a single model for multiple tasks relative to models for each task, enabling wider and more democratized deployment; and there can be some degree of positive transfer between tasks, which can be especially beneficial for tasks that have limited training data (Ruder, 2017).

However, we find that much of the existing work on multitask model training would be infeasible for many downstream practitioners to implement – either because it requires retraining a model from scratch or significantly modifying architectures that are typically abstracted away by platforms like HuggingFace.

From an accesibility standpoint, we might hope that fine-tuning strategies could be employed to effectively train a multi-task model. Fine-tuning on top of a pretrained model is a central paradigm of foundational models and enables large models to be adapted to diverse downstream tasks in a computationally efficient and practically accessible way (Bommasani et al., 2021). As models continue to get larger and – in some cases – less open source (see the recent release of GPT-4 as evidence for both points), model practitioners will be even more reliant on fine-tuning if they hope to train models that are in line with SOTA models.

However, fine-tuning on a single dataset has been shown to distort pretrained features and lead to lower out-of-distribution (OOD) performance (Kumar et al., 2022). Given this, we predict fine-tuning on a single dataset will generate poor embeddings for a multi-task model. Fine-tuning on all three tasks might yield better results, but this presumes that the practitioner has access to the true downstream distributions at inference time and is likely susceptible to degraded performance if there's any distribution shift.

Existing work from Gao et al. (2021) found that fine-tuning on an NLI dataset yields more robust embeddings that perform well even on non-NLI distributions – and Phang et al. (2018) found that fine-tuning on an intermediate, most generalized task before fine-tuning on the direct downstream task can yield better multi-task performance than training on the downstream tasks alone. However, Gao et al. (2021) only tested their approach on a variety of semantic textual similarity datasets, so it's unclear if their approach extends to other tasks in NLP.

This approach doesn't require any fundamental changes to the model or for the base model to be retrained; it simply requires one additional pass of fine tuning with a small addition to the output layer for the NLI task. Leveraging these insights, we quantify the performance and robustness of fine-tuning on an NLI dataset before training on our downstream datasets.

3 Related Work

Multi-task learning has been an active area of research in the field of deep learning. However, training these models has proved to have it's own set of challenges. Previous approaches require large amounts of additional pretraining on a wide variety of tasks (Aghajanyan et al., 2021). Other approaches involve fundamental changes to the base architecture of the model (Karimi Mahabadi et al., 2021).

Gao et al. (2021) introduces SimCSE as an alternative option for multi-task learning that only requires a small amount of additional fine-tuning. This makes it an attractive option for our project. SimCSE achieves state-of-the-art performance in sentence embeddings using a contrastive learning framework that includes supervised and unsupervised approaches.

Yu et al. (2020) identifies the challenges of multi-task optimization and proposes a simple yet general approach to avoid detrimental gradient interference between tasks. The approach involves projecting a task's gradient onto the normal plane of the gradient of any other task that has a conflicting gradient. This approach leads to substantial gains in efficiency and performance in multi-task supervised and multi-task RL problems. Gradient surgery could be another approach to our problem, but we didn't use it for two reasons: (1) it requires access to gradients from all tasks within the same batch, and we thought this might be unrealistic for model practitioners and (2) it seemed like many other groups wanted to use it and we wanted to try something different.

Finally, Kumar et al. (2022) highlights the tradeoff between in-distribution (ID) and out-of-distribution (OOD) accuracy in fine-tuning pretrained models for downstream tasks. It suggests that full fine-tuning can lead to worse accuracy OOD than linear probing when the pretrained features are good and the distribution shift is large. The paper recommends a two-step strategy of linear probing followed by full fine-tuning as an alternative approach.

4 Approach

Below, we describe our approach. To reiterate, our experiments and approach emphasize computational efficiency (e.g. avoiding re-training from scratch) and feasibility of implementation (e.g.

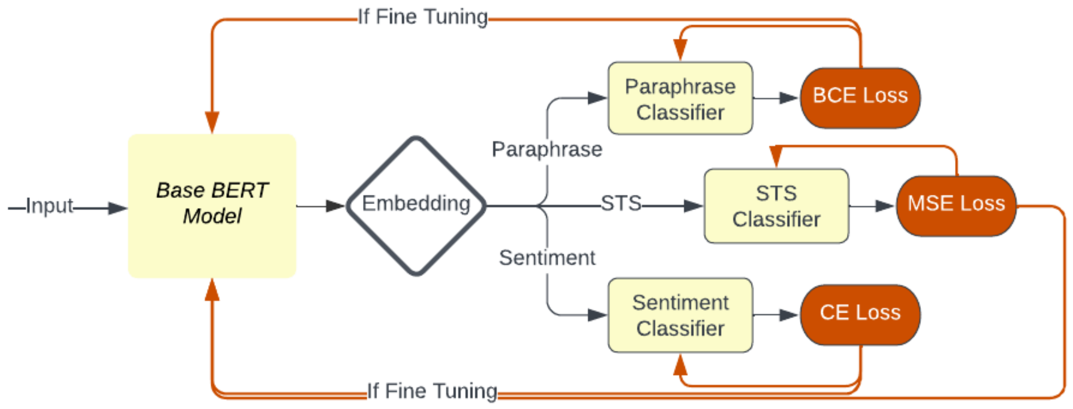

Figure 1: Simplified diagram of our general model architecture (without intermediate fine-tuning). Red arrows indicate backpropogation of loss.

avoiding making fundamental changes to parts of the model typically abstracted away from model pracitioners).

4.1 Overall Model

Our overall model is built on top of the original pretrained BERT model introduced by Devlin et al. (2018). The original BERT model with pretrained weights (or fine-tuned weights in certain experiments) is used to generate embeddings for the input sentence or sentences.

For our BERT embedding, we use the hidden-state of the last layer of the first token of the sequence which is then passed through a fully connected and Tanh layer (pooler output).

We define distinct classifiers – described in further detail below – for each of our 3 tasks, and we pass embedding(s) to these 3 distinct classifiers depending on the task of the input instance x_i.

For the reason discussed earlier, we avoid making significant changes to the model and instead rely primarily on fine-tuning and simple changes that be applied on top of a pre-trained model in order to generate robust embeddings.

4.2 Output Layers

We define a separate output layer for each of three tasks; the task label – which we know at train and test time – determines which output layer we pass the embedding(s) from the base BERT model to.

Each of these output layers generates a prediction which – at train time – thereby generates a loss. The loss function is different for each classifier. The loss is then backpropogated only to the parameters of the output layer that made the prediction (e.g. the loss incurred on a paraphrase example doesn't affect the gradient of the sentiment classifier). If we are fine-tuning the models, then the loss is also used to update the parameters of the base BERT model.

4.2.1 Sentiment Analysis Classifier

In sentiment analysis, our input is a single token sequence that we retrieve a BERT embedding for. Our desired output is the class prediction for the sentiment of the input sentence; there are 5 sentiment classes.

We feed the embedding from BERT into a single fully connected layer which outputs a (1 x N) logit where N is the number of sentiment classes. We then pass this (1 x 5) logit into a sigmoid activation function to get the probability distribution over the sentiment classes; we take the argmax of this distribution to determine our prediction y_i.

4.2.2 Semantic Textual Similarity Regressor

In semantic textual similarity, our input is two token sequences that we retrieve BERT embeddings for.

We test two approaches to designing our decoder layer to translate these two BERT embeddings into a single scalar prediction.

Our first architecture concatenates the two vectors together; we refer to this as our 'concatenate decoder'. Our second architecture element-wise multiplies the two vectors together; we refer to this as our 'weighted dot-product' decoder because the weight matrix of the following linear layer can learn a weighted summation over the element wise multiplication.

In both architectures, the resulting vector is passed through a fully connected layer which directly outputs a single scalar prediction for the similarity of the two sequences.

4.2.3 Paraphrase Classifier

In paraphrase classification, our input is two token sequences that we retrieve BERT embeddings for. Similar to semantic textual similarity, we again test best the concatenate and weighted dot product as our decoder structure.

The resulting embedding is passed into a fully connected layer and then into a Sigmoid nonlinear activation layer to determine the probability that the inputs are a paraphrase of each other. We use the simply majority as our threshold.

4.3 Training Approach

4.3.1 Training on all three downstream tasks.

We have two approaches to training on all three downstream tasks: linear probing and fine-tuning. In linear probing, we freeze the BERT parameters and only train the output layers. In fine-tuning, we also update the parameters of the base BERT model.

At train time, each mini-batch only contains examples from a single task; by isolating task examples from each other, we can ensure that the loss from each batch is only applied the relevant classifier. That is to say, the parameters of the sentiment classifier aren't affected by loss incurred on semantic textual similarity examples.

We use Adam as our optimization function, and we maintain separate Adam instances for each output classifier so that momentum and learning rates are calculated only based on gradient updates made to that classifier and not based on updates to any of the other classifiers.

4.3.2 Integrating contradiction/entailment pairs as an intermediate fine-tuning task.

Inspired by the work of Gao et al. (2021), we test the impact of finetuning our BERT model on a natural language inference dataset using entailment pairs as ground truth positives and contradiction pairs as ground truth negatives. We refer to this fine-tuning strategy as SimCSE (Simple Contrastive Sentence Embeddings) – as in the paper.

More concretely, we adopt the contrastive loss function defined in Equation 1. Here, h_i represents the BERT embedding of the premise; h_i^+ refers to the BERT embedding of the entailment; h_i^- refers to the BERT embedding of the contradiction; N is the batch size. Intuitively, this loss function should incentivize the embeddings of entailed sentences to be close together and the embeddings of contradictory sentences to be far part.

$$-log \frac{e^{cos\ sim(h_i, h_i^+)}}{\sum_{j=1}^{N} e^{cos\ sim(h_i, h_i^+)} + e^{cos\ sim(h_i, h_i^-)}} \tag{1}$$

After fine-tuning on these entailment and contradiction pairs, we adopt the same training strategy as described above.

This approach requires no modification to the architecture of the model, is agnostic to the downstream classifier structure, and is easy and computationally efficient to train since it only requires fine-tuning on top of a large pre-trained model.

5 Experiments

5.1 Data

For sentiment analysis, we use the Stanford Sentiment Treebank dataset (11,855 instances with 5 classes).. For paraphrase detection, we use a subset of the Quora dataset (202,152 instances with 2 classes). For semantic textual similarity, we use a subset of the SemEval STS Benchmark dataset (8,628 instances with 6 classes). For NLI data, we use the same subset of the MNLI dataset used by Gao et al. (2021).

5.2 Evaluation method

Since our project is on the impact of fine-tuning methods, we define our baseline as 'linear probing' which involves no fine-tuning and gives us a good idea of what the raw BERT embeddings passed into a fully connected layer can yield. Against this linear probing baseline, we evaluate two styles of fine-tuning: round-robin fine-tuning and intermediate fine-tuning with a contrastive loss before round-robin training.

We evaluate the performance of our model on each task separately using the appropriate evaluation metric: accuracy for SST and paraphrase classifcation, pearson correlation for STS.

SimCSE intermediate fine-tuning. For SimCSE intermediate fine-tuning, we examine how intermediate fine-tuning affects the performance of linear probing and round robin fine-tuning; concretely, this means we intermediately fine-tune and then either train linear probing layers on the target task or fine-tune on the target tasks afterwards and compare this performance against the performance these architectures achieved without integrating SimCSE intermediate fine-tuning.

5.3 Experimental details

For linear probing, we use a learning rate of 1e-3; for fine-tuning on the 3 final tasks we use a learning rate of 1e-3. We use batch sizes of 32 (which is the maximum we could use given the RAM constraints on our instance).

For SimCSE fine-tuning, we attempt to replicate the conditions of the paper by using 3 epochs for train and a learning rate of 5e-5. They use batch sizes of 64, but we are unable to accomplish this given the memory constrains of our cluster and instead use batch sizes of 32.

5.4 Results

5.4.1 Impact of SimCSE

In Table 1, we report the accuracies on our three downstream tasks using our baseline round-robin style training and using the SimCSE contrastive loss intermediate fine-tuning objective. Surprisingly, we find that intermediate fine-tuning with a contrastive loss term lowers performance compared to round-robin training both when fine-tuning and when only linear probing. It harms performance most on the semantic textual similarity task and slightly helps performance on the sentiment analysis task in the linear probing architecture; this is especially surprising since Gao et al. (2021) evaluate the SimCSE methodology on STS tasks.

In section 6, we discuss possible explanations for SimCSE's poor performance on our tasks.

5.4.2 Leave one out fine-tuning testing as a generalizability baseline

In Table 2, we show the impact of fine-tuning on two tasks but not on a third, left-out task. All tasks, including the left-out one still have their classifier layers trained, but the left-out task doesn't have its loss propagated to the layers of the $BERT_{base}$ model. We use this test to demonstrate how

	Linear Probing		Finetuning	
	Round Robin	w/ SimCSE	Round Robin	w/ SimCSE
Sentiment Analysis	.386	.390	**.494** (.480)	.487
Paraphrase Classification	.694	.632	**.823** (.822)	.775
Semantic Textual Similarity	.304	.195	**.370** (.355)	.346
Average	.461	.406	**.562** (.552)	.536

Table 1: Impact of intermediate fine-tuning on an NLI dataset with a contrastive learning objective relative to round-robin training. We find that, contrary to our expectation, intermediate fine-tuning harms performance. All performance figures are on dev set except for our best model which we evaluate on the test set and report performance in parentheses.

	Left Out Dataset During Finetuning		
	Sentiment	Paraphrase	STS
Sentiment Analysis Accuracy	0.278 (-.092)	0.495 (+.001)	0.501 (+.007)
Paraphrase Classification Accuracy	0.813 (-.010)	0.628 (-.066)	0.825 (+.002)
Semantic Textual Similarity Correlation	0.395 (+.025)	0.357 (-.013)	-0.064 (-.434)

Table 2: Impact of leaving out a single dataset during fine-tuning. Figures in parentheses are the difference in metric score from the baseline. Along the diagonal, the relevant baseline is the performance of that task in the linear probing setting (since we don't fine-tune); everywhere else, the relevant baseline is that task's fine-tuning performance.

fine-tuning can degrade the robustness of embeddings on unseen tasks – consistent with the findings of Kumar et al. (2022) – and to understand, partially, which tasks potentially degrade the performance of the other tasks.

We find that leaving out a task during fine-tuning significantly degrades performance on that task (even though we trained a linear classifier on the task's data) and this degradation of performance is largest for the STS task and smallest for the paraphrase classification task. For a left-out task, we define degradation in performance relative to the performance of that task in the 'linear probing' experiment in Table 1 (which didn't fine-tune on any tasks); this is the fair comparison since we don't fine-tune and only train output layers for the left-out task. For an included task, we define change in performance relative to the performance of that task in the fine-tuning experiment; this is appropriate since we fine-tune on this task.

We also find that leaving out sentiment data improved the performance on the STS task and, vice versa, leaving out STS data improved performance on the sentiment analysis task – potentially signalling that the STS and sentiment tasks have conflicting gradients and are dissimilar enough that the same model's embedding struggles to perform well on both.

Contrastingly, we find that leaving out paraphrase data during the fine-tuning stage doesn't improve performance on the other tasks. It's possible that including paraphrase data doesn't conflict as much with either of the two other tasks, so leaving it out of the fine-tuning stage doesn't improve performance on the other tasks.

5.4.3 Concatenate vs weighted dot product decoder for STS and paraphrase tasks

We evaluate two decoder schemes for the output layers of the STS and Paraphrase tasks. Our first approach simply concatenates the two embedding vectors x_i and x_j together and passes them through a fully connected layer. Our second approach takes the elementwise product of vectors x_i and x_j and passes them through a fully connected layer. In paraphrase detection, the logits are then passed through a Sigmoid layer.

We find that weighted dot product yields better results and converges faster; we use weighted dot product for our SimCSE experiment and don't test the concatenate decoder.

	Linear Probing		Finetuning	
	Concatenate	Weighted Dot Product	Concatenate	Weighted Dot Product
Paraphrase Classification	.681	.694	.760	**.823**
Semantic Textual Similarity	.277	.304	.345	**.370**

Table 3: Performance of concatenating vs. weighted dot product of sentence embeddings on the paraphrase classification and semantic textual similarity. We find that the dot product passed into a fully connected layer yields significantly better performance.

6 Understanding why SimCSE harms performance

Gao et al. (2021) find that training with a contrastive loss objective (defined in Equation 1) using entailment pairs and contradiction pairs as negatives improves the performance of BERT embeddings on a variety of STS tasks.

However, when we adopt this training strategy for our task, we find that it hurts performance relative to the performance we achieve when we don't intermediately fine-tune using this contrastive loss.

We offer three possible explanations for this degraded performance: (1)Gao et al. (2021) do not learn a fully connected layer on top of their BERT embeddings for their downstream tasks while we do; (2) the multi-task setting we are in result in conflicting gradients which harms performance; (3) fine-tuning performance is sensitive to the particular data distribution being used in pre-training, fine-tuning, and at evaluation.

6.1 Difference in output classifiers

Expounding on the first point, Gao et al. (2021) use the cosine similarity of the outputted embeddings from $BERT_{base}$ directly as their input to their activation function. We, contrastingly, learn a fully connected layer on top of our outputted embeddings before passing to the activation function. This fully connected layer potentially adds expressive power to the model that mitigates the need for the intermediate step of tine-tuning the embeddings using the SimCSE objective.

6.2 Multitask conflicting gradients

Regarding the second point, we are in a multi-task regime with 3 distinct tasks and datasets. This introduces a variety of factors that can alter performance: Yu et al. (2020) find that training on multiple tasks can result in conflicting gradients – concretely, gradients that point in the opposite direction and therefore have opposite cosine similarities – that ultimately result in the model failing to efficiently converge to a global optima. With the introduction of SimCSE, we implicitly introduce another task whose gradients potentially conflict with the gradients of our other tasks, resulting in a further suboptimal convergence point. In Table 2, we generally find that fine-tuning on two tasks but only linear probing on the other results in performance gains for the two fine-tuned tasks at the cost of performance for the non-fine-tuned task. We suggest this is evidence of conflicting gradients in our setting.

6.3 Sensitivity to data distributions and distribution shift

Expounding on the final reason, Kumar et al. (2022) has found that fine-tuning can degrade performance on OOD examples, and Peters et al. (2019) has found that the success of fine-tuning depends on the similarity of the pretrain and fine-tune tasks: at a high level, we take this as a sign that fine-tuning is sensitive to data distributions – and, by extension, distributional shift – involved in the pre-training and fine-tuning stages of training. Indeed, Gao et al. (2021) find performance of their SimCSE method strongly depended on the evaluation dataset and the choice of fine-tune dataset; we chose the dataset they found performed best as the fine-tune dataset, but it's possible that this choice of dataset performs specifically well for their choice of evaluation datasets and performs poorly for ours.

7 Conclusion

In this paper, we explored the efficacy of various fine-tuning methods to generate high-quality embeddings that generalize well on three natural language processing tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We evaluated the impact of intermediately fine-tuning on an NLI dataset using Gao et al. (2021)'s proposed SimCSE approach to improve downstream performance and compared it to linear probing and fine-tuning only on the target tasks tasks.

Surprisingly, we find that intermediate fine-tuning with a contrastive loss term harms performance compared to round-robin training, both when fine-tuning and when only linear probing. We suggest three possible explanations for this degraded performance: the use of a learnable output layer, conflicting gradients in multi-task settings that lead to suboptimal convergence, and the sensitivity of fine-tuning to distributional shift between training stages. Our results demonstrate that fine-tuning, while often effective in improving performance, is sensitive to the datasets used in each stage of training and the target distribution used at evaluation.

Additionally – inspired by recent work from Kumar et al. (2022) – we evaluate the impact of fine-tuning on two the three datasets on the left-out dataset to estimate the loss of generalizability incurred by fine-tuning. We find that leaving out sentiment analysis data during the fine-tune stage improved performance on the STS task, and vice versa, leaving out STS data improved performance on the sentiment analysis task. Conversely, leaving out paraphrase data during the fine-tuning stage did not improve performance on the other tasks. Our work suggests certain tasks may directly conflict with each other (sentiment analysis and STS) while other tasks may be more compatible with others (paraphrase detection). We encourage further research into this phenomenon – potentially by measuring the degree of cosine similarity in the gradients of different tasks.

References

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

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the opportunities and risks of foundation models.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th*

International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 565–576, Online. Association for Computational Linguistics.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*.