Simple Contrastive Learning for Multitask Finetuning

Stanford CS224N Default Project

Gui David, Khaing Mon, Annie Zhu
Department of Computer Science
Stanford University
{gdavid, ksmon, anniezhu}@stanford.edu

Abstract

We explore extensions and methods to improve and fine-tune BERT, a model that uses Bidirectional Encoder Representations from Transformers to develop deep contextual word representations. Since its release, BERT has shown to be the base for state-of-the-art models for a wide range of tasks. In this project, we employ the following extensions for the Vanilla BERT model: Additional pretraining using Simple Contrastive Learning, Gradient Surgery, Multitask Fine-Tuning, Hyperparameter Finetuning, and Model Ensembling to improve performance on three downstream tasks: 1) Sentiment Analysis, 2) Paraphrase Detection, and 3) Semantic Textual Similarity. As a baseline, our model performance using vanilla fine-tuning was 0.526, 0.0442, and -0.041, respectively. After implementing various extensions, our ensembled model yielded the respective accuracies of 0.528, 0.625, and 0.446.

1 Introduction

BERT is a language representation model that utilizes bidirectional transformers to develop contextually rich representations of text and language. This large pre-trained model is often helpful for downstream tasks as it can be fine-tuned to create task-specific models Devlin et al. (2019) that perform at a high level. In this project, we finetune MinBERT, a scaled-down version of BERT, to improve performance on three specific downstream natural language tasks. This is an exciting problem as large language models often attempt to encode universal sentence embeddings but sometimes fail to capture or understand the exact semantics of words in various contexts. For this reason, it may be the case that these large models may do exceptionally well for a subset of downstream tasks but not others.

Pretraining models, however, is one way for models to generate and utilize relevant and meaningful sentence embeddings. When trained on large pieces of data, pre-trained models can learn more meaningful and complex language patterns and be fine-tuned for specific tasks by capturing domain and context-specific meanings from the dataset. BERT is already a pre-trained model, but in this project, we introduce additional pretraining with simple contrastive learning to further finetune the model for our specific tasks. In addition, we introduce refinements such as Multitask Fine-tuning, Gradient Surgery, and Hyperparameter Finetuning to improve the overall model performance.

2 Related Work

SimCSE by Gao et al. (2021), which stands for Simple Contrastive Sentence Embedding framework, is a way to improve sentence embeddings in unsupervised and supervised cases. The general idea behind contrastive learning is to form representative embeddings by looking at sentences in pairs or triples to pull semantically close sentences together and push apart semantically different sentences. One big motivation for better sentence embeddings is that it can massively cut down the time for search and clustering-based problems. For example, finding the most similar pair of sentences in a collection of 10,000 sentences requires about 50 million pair-wise computations with regular BERT,

but then having a method of mapping sentences close to their neighbors dramatically cuts down the pool of candidate pair-wise computations needed (Reimers and Gurevych, 2019).

Unsupervised Contrastive Learning paired with pre-trained models develops these deep sentence embeddings from unlabeled data. It exploits BERT's ability to produce contextual embeddings because it uses the same sentence with differing dropout masks as positive pairs, effectively turning Unsupervised Contrastive Learning into a task that requires no additional supervised data. In extension, Supervised Contrastive Learning builds on this by incorporating annotated sentence pairs and optimizes for the learning objective by comparing labeled positive and negative pairs. This has been shown to significantly outperform existing sentence embedding methods on standard Semantic Textual Similarity (STS) tasks.

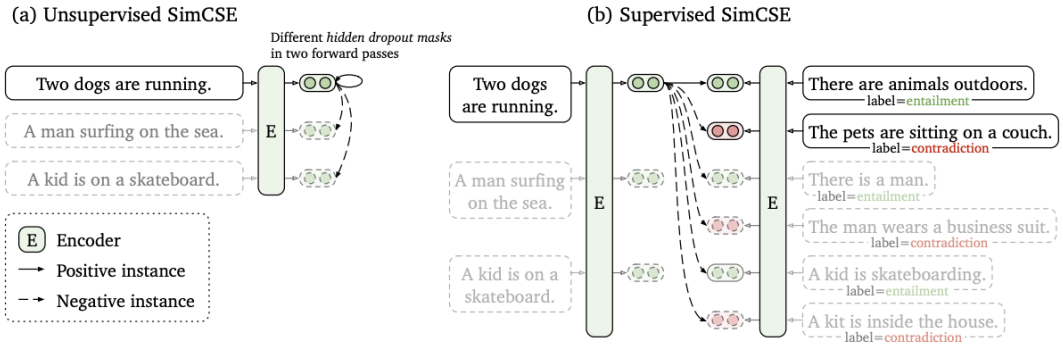

Figure 1: Unsupervised Contrastive Learning Work Flow (Left) and Supervised Contrastive Learning Work Flow(right), taken from Gao et al. (2021)

Other works that we draw from include an implementation of multitask learning from Bi et al. (2022), where the researchers added together the losses of multiple tasks, and an implementation of gradient surgery from Yu et al. (2020) for fixing gradient descent when there are conflicting directions for the different tasks.

3 Approach

3.1 Multitask Fine-tuning Extension

In the baseline, BERT is fine-tuned on a single task, and in this extension, we wanted to fine-tune BERT on multiple tasks using multiple datasets. To do this, we explore three different methods of multitask fine-tuning. First, we executed sequential training, also known as round-robin training, where BERT is fine-tuned on one task and dataset, then on another task and corresponding dataset. Second, we performed concurrent training, where in each epoch, a batch is taken from every dataset, and the loss is computed for every task before being combined, and a step is taken. In this framework, the multiple datasets are iterated concurrently. Lastly, we add gradient surgery to the concurrent setup. This method comes from the idea that the gradient directions of different tasks may conflict, so surgery can be used to project the gradient of a particular task onto the normal plane of a conflicting task's gradient.

3.2 Unsupervised Contrastive Learning (UCL) Extension

At a high level, unsupervised SimCSE takes an input sentence and predicts itself using dropout as noise (see Fig 1). To do this, we feed the same input sentence into the encoder twice. The encoder uses the same standard dropout rate. The two outputs from the encoder are now regarded as "positive" pairs. Every other sentence in the batch is regarded as a "negative" pair to this input sentence. The goal is for the model to predict the "positive" one amongst all the other "negatives" in the batch by

optimizing for this objective function:

$$l_i = -\log \frac{e^{\text{sim}(\mathbf{h}_i^{z_i}, \mathbf{h}_i^{z_i'})/\tau}}{\sum_{j=1}^{N} e^{\text{sim}(\mathbf{h}_i^{z_i}, \mathbf{h}_j^{z_j'})/\tau}}$$

where **z,z'** are the two different embeddings of the same input sentence, and N represents the number of sentences in the batch. In our implementation, we use Unsupervised Contrastive Learning as a Pretraining method for our MinBERT model. We first conduct Unsupervised Contrastive Learning on the SST, STS, and Quora datasets, using input sentences as positive pairs to themselves. The loss function would be the training objective highlighted above. Once done, we save the weights of our pre-trained models, which are loaded in for the official training phase.

3.3 Supervised Contrastive Learning (SCL) Extension

This method extends the unsupervised SimCSE framework by using neutral, positive, and negative embedding sentences. In Supervised Contrastive Learning, semantically dissimilar sentences are treated as negative examples. The objective function for Supervised Contrastive Learning is modified as follows:

$$-\log \frac{e^{\text{sim}(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} \left(e^{\text{sim}(h_i, h_j^+)/\tau} + e^{\text{sim}(h_i, h_j^-)/\tau} \right)} \tag{8}$$

where h_i is the anchor sentence embedding, h_i^+ is a positive sentence embedding that is semantically similar to h_i, h_j^- is a negative sentence embedding that is semantically dissimilar from h_i, N is the number of sentences in the batch, sim is a function that measures the similarity between two sentence embeddings, and τ is a temperature parameter that scales the similarity measure. This objective function encourages the model to distinguish the anchor sentence from a batch of other sentences by bringing the embeddings of similar sentences closer together while pushing dissimilar sentences apart in the embedding space.

In our implementation, we also used Supervised Contrastive Learning as a pretraining method for our MinBERT model. To do so, we modified the Quora dataset by first filtering for neutral and positive pairs and then adding a third column with shuffled sentences from the positive pairs, ensuring that no sentence ends up in the same position. We computed the loss via the aforementioned objective function and saved the weights of our pre-trained models, which are then loaded in for the official trained phrase.

4 Experiments

4.1 Data, Evaluation Methods, and Experimental Details

This project is evaluated on the 3 pre-defined default tasks using the 3 pre-defined default datasets described in the project handout (i.e., the Stanford Sentiment Treebank, the Quora Question Pairs dataset, and the STS Benchmark dataset). As for the experimental setup, we tried out various epoch counts, learning rates, dropout rates, and combinations of pretraining and finetuning methods described in detail in the following sections.

4.2 Results

We present the summary of the best performance within each extension and the fusion of the extensions in the following table, and details about each part are in the following subsections.

Task	Baseline	Multitask	UCL	SCL	Ensemble
SST Accuracy	0.526	0.502	0.514	0.487	0.528
Paraphrase Accuracy	0.442	0.625	0.625	0.625	0.627
STS Pearson Correlation	-0.041	0.126	0.183	0.196	0.447
Overall Dev Score	0.483	0.571	0.577	0.570	0.626

Table 1: Main Results

4.2.1 Multitask Fine-tuning Extension

The baseline model uses just single-task finetuning so, to improve on that, we implement multitask finetuning on all three datasets for all three downstream tasks. The results are in the table below.

Task	Baseline	Sequential	Concurrent	Gradient Surgery
SST Accuracy	0.526	0.300	0.502	0.493
Paraphrase Accuracy	0.442	0.581	0.625	0.625
STS Pearson Correlation	-0.041	0.172	0.126	0.184
Overall Dev Score	0.483	0.499	**0.571**	0.570

Table 2: Multitask Finetuning with 3 Datasets

As expected, we see improvements with multitask finetuning because the model now has more data to learn from and is being molded to our specific tasks. Next, we can observe that concurrent training performs the best while the gradient surgery harms the performance. One explanation for this dip in accuracy is that Gradient Surgery works best if optimized for one task in particular because it computes the projection of the gradient of tasks into the plane of a conflicting task's gradient. Still, since we care about all three tasks equally, this context is not a good application of Gradient Surgery.

4.2.2 Unsupervised Contrastive Learning Extension

The first approach uses only the SST dataset, whereas the second uses both the SST and STS datasets for the Unsupervised Contrastive Learning objectives. (In both cases, contrastive learning is used first to pretrain the model, and then vanilla single-task training is used to finetune the model.) The results are as follows:

Task	Baseline	One Datasets	Two Datasets
SST Accuracy	0.526	0.520	0.528
Paraphrase Accuracy	0.442	0.384	0.570
STS Pearson Correlation	-0.041	0.068	0.050
Overall Dev Score	0.483	0.479	**0.541**

Table 3: Unsupervised Contrastive Learning with 2 Datasets

Unsurprisingly, Unsupervised Contrastive Learning done with both datasets yielded the best results. This may be because our model has developed more complex sentence embeddings through both the STS and the SST datasets. There are two interesting observations to highlight: 1) Despite pre-training the model using Unsupervised Contrastive Learning on the SST dataset, both models see decreased scores in the two models, and 2) Unsupervised Contrastive Learning Pretraining was only conducted on the SST and STS datasets. However, paraphrase accuracy seemed to go up significantly after pretraining.

Next, we combined our work from the multitask finetuning with Unsupervised Contrastive Learning. In the following table, all predictions were created by running contrastive learning and finetuning the multitask concurrent learning function (on all three downstream tasks) instead of the single-task learning function. Under this framework, we compare how the model works when Unsupervised Contrastive Learning is done on two datasets sequentially vs. when Unsupervised Contrastive Learning is done on three datasets concurrently.

Task	Best No Multitask	Two Datasets + Multi	Three Datasets + Multi
SST Accuracy	0.528	0.514	0.488
Paraphrase Accuracy	0.570	0.625	0.625
STS Pearson Correlation	0.050	0.183	0.187
Overall Dev Score	0.541	**0.577**	0.569

Table 4: Unsupervised Contrastive Learning with Multitask

We are able to achieve a 3.6 percentage point increase (from 54.1 to 57.7) in the overall dev score just by incorporating multitasking learning techniques into the Unsupervised Contrastive Learning model. Once again, the improvements mainly go towards the Paraphrase Detection task.

4.2.3 Supervised Contrastive Learning Extension

We evaluate varying degrees of model pretraining in our Supervised Contrastive Learning Extension. The first setting, Supervised Only, involved pretraining a model with Supervised Contrastive Learning. The second, Unsupervised First, began with the pretraining of the model entirely Unsupervised Contrastive Learning, after which the model weights were loaded into the supervised training loop. The third model, Supervised First, was similarly pretrained with Supervised Contrastive Learning and then passed into the unsupervised training loop. As a last step, we loaded the weights of all those pre-trained models in our regular multitask finetune training loop. All unsupervised and supervised contrastive pretraining were conducted for 3 epochs, while the last multitask training loop was trained for 10 epochs.

Task	Supervised Only	Unsupervised First	Supervised First
SST Accuracy	0.481	0.482	**0.487**
Paraphrase Accuracy	0.625	0.625	**0.625**
STS Pearson Correlation	**0.202**	0.155	0.196
Overall Dev Score	0.569	0.562	**0.570**

Table 5: Unsupervised Contrastive Learning with 3 Datasets

4.3 Model Ensembling Extension

For the ensembling part, we combined the results of different models to ensure our system's overall performance on each separate task is maximized. We selected three of the best-performing models: one trained with Unsupervised Contrastive Learning using two datasets, another trained with Supervised Contrastive Learning that was pre-trained for three epochs, and the third pre-trained with Supervised Contrastive Learning for ten epochs (without using a prediction head for the similarity and paraphrase prediction functions; See Section 5.4 for more details). The results are as follows:

Task	Unsup. 2 Dataset	Sup. 3 Epoch	Sup. 10 Epoch-No Head	Ensemble
SST Acc	**0.528**	0.480	0.493	0.528
Paraphrase Acc	0.570	**0.627**	0.434	0.627
STS Pearson Corr	0.050	0.122	**0.447**	0.447
Overall Dev Score	0.541	0.556	0.550	**0.626**

Table 6: Model Ensembling for Best Performance on Each Task

4.3.1 Hyperparameter Training

We begin our hyperparameter training by testing out various numbers of epochs used for the pretraining portion. In the table below, the first two columns from the left represent pretraining that happens with Unsupervised Contrastive Learning, while the third column represents Supervised Contrastive

Learning. A key finding from this section is that for contrastive learning, the best-performing model comes from decreasing the number of epochs. This result suggests that it is easy to overfit with contrastive learning, which bolsters our hypothesis that supervised contrastive learning is not as beneficial as expected.

Task	Two Datasets + Multi	Three Datasets + Multi	Supervised
10 Epochs	0.564	0.564	0.550
2 Epochs	**0.577**	0.565	0.564
3 Epochs	0.569	0.569	0.569

Table 7: Number of Epochs for Pretraining Affects on Overall Dev Score

Next, we test different learning and dropout rates using the Unsupervised Contrastive Learning model run on two datasets as pretraining and multitask finetuning. In the table below, the Default represents the output when the learning rate is 1e-5, and the dropout rate is 0.3, which are the default values from MinBert. In each of the other columns, if the learning rate is being changed, then the dropout rate is the default value, and vice versa. In this set of tests, we find that the default learning rate and dropout rate work the best.

Task	Default	lr = 1.5e-5	lr = 2e-5	dr = 0.4
SST Accuracy	0.514	0.472	0.471	0.500
Paraphrase Accuracy	0.625	0.625	0.625	0.625
STS Pearson Correlation	0.183	0.183	0.240	0.137
Overall Dev Score	**0.577**	0.563	0.572	0.565

Table 8: Different Learning Rates for Pretraining UCL

5 Analysis

5.1 Multitask Fine-tuning Extension

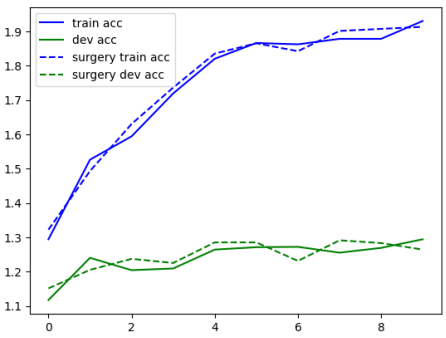

Figure 2: Accuracy over epochs during training

We plot the train and dev accuracy over the 10 epochs of multitask training to analyze how the model is learning.

First, we notice that train accuracy steadily rises throughout the procedure while the dev accuracy plateaus a bit in comparison. This difference in improvement in the later epochs suggests that the model might be overfitting as the benefits of the training are no longer apparent in the new unseen data. Second, we see that the gradient surgery lines hug the non-gradient surgery lines, which explains why adding gradient surgery leads to no improvement in the scores.

Another interesting qualitative aspect of the different methods is the tradeoff between more training and computational resources required. Sequential training took well over 5 hours, at over 30 minutes per epoch, whereas concurrent training took under 1 hour, making concurrent training the preferable method.

5.2 Unsupervised Contrastive Learning Extension

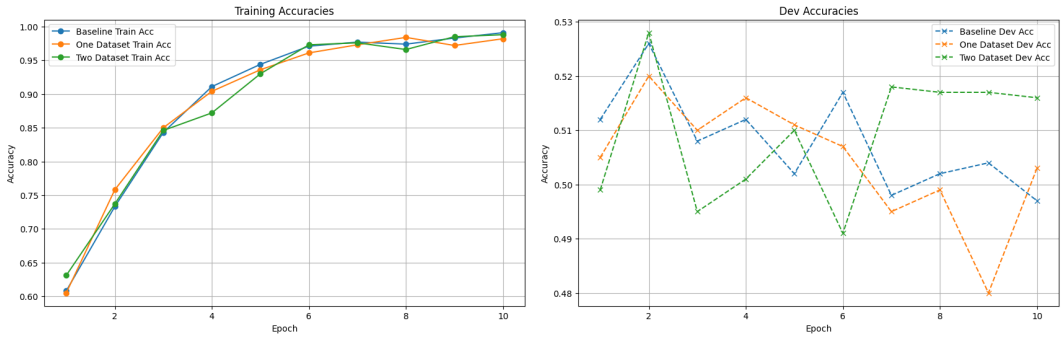

Figure 3: Training Accuracy (left) and Dev Accuracy (right) over Epochs for Unsupervised Contrastive Learning Pretrained Model and Vanilla Multitask

The above plots show the training and dev accuracy for the baseline vanilla-trained model and an Unsupervised Contrastive Learning pre-trained model with either 1 or 2 datasets. As one can see, both 1 and 2 dataset versions improved the dev accuracy of the model, indicating that the model was adjusting moderately better to the training data. Despite this, it was surprising to observe that the baseline model performs better than the 1 dataset model. This is interesting, as we had expected that pretraining on even one dataset would result in deeper embeddings that would help increase performance across the board.

Using two datasets for analysis yielded the best results for us. The graph shows that this model also produces the best training and dev accuracies. For both the baseline and the 1 dataset model, we can see that the dev accuracies reach a peak around the 2nd or 3rd epoch. These dev accuracies continue to fall as the number of epochs increases, suggesting that overfitting may have occurred. The 2 dataset model was the only one that yielded high scores across the board.

5.3 Supervised Contrastive Learning Extension

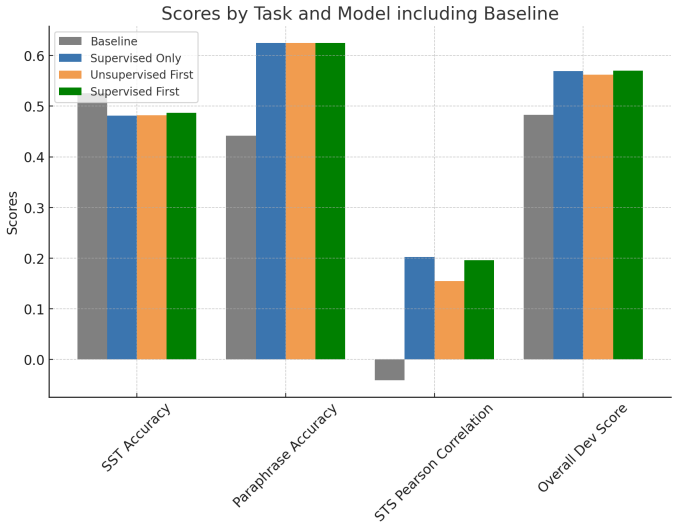

Figure 4: Performance analysis of supervised models on NLP tasks, contrasting baseline and fine-tuned models across various metrics.

The main takeaway from the Supervised Contrastive Learning experiments is the subtle improvement for various tasks. For example, performance on the paraphrase detection task improved slightly, which validates the assumption that regular exposure to semantically similar and dissimilar sentence pairs allows the model to make more nuanced distinctions regarding paraphrasing. This might be explained by the fact that the model learns more nuanced distinctions between sentences that were judged to be neutral, positive, and negative in our modified Quora Question Pairs dataset.

5.4 Underlying Prediction Functions

During our experimentation, we identified our prediction functions as a potential source of weakness, so we changed our implementation. To understand how switching from cosine similarity to using linear layers for our predict similarity and predict paraphrase functions affected performance on each task, we computed the change in metrics for eight different models. (The eight models are Baseline, Concurrent, Grad Surgery, One Dataset UCL, Two Datasets UCL, Two datasets UCL + Multitask, Three datasets UCL + Multitask, and Supervised CL). Then, we take the average change in value for each metric and plot them on the graph below.

From this graph, we can see that our use of prediction heads instead of simple cosine similarity helped improve the Paraphrase accuracy but also harmed STS Pearson Correlation by the same magnitude. This tradeoff we see from our changes suggests that for the STS task, it might have been more beneficial to stick with the cosine similarity method and only use the linear layers/prediction heads method on the Paraphrase task.

This hypothesis turned out to be true. We tested a final version of prediction functions, where we brought back cosine similarity for similarity prediction, and this yielded our highest-performing model without model ensembling.

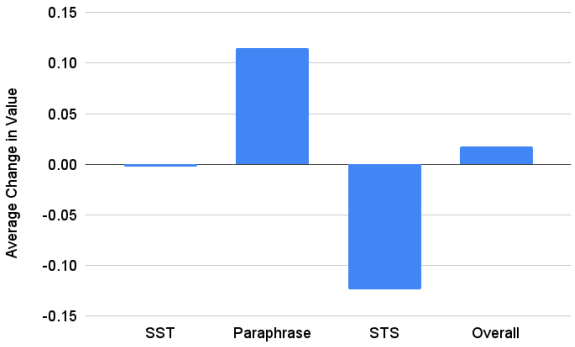

Figure 5: Change in metrics due to prediction functions

Task	Two Datasets Unsupervised	Unsupervised + Supervised CL
Only Linear Layers	0.577	0.562
Linear Layers and Cosine	**0.593**	0.588
Difference	+0.016	+0.025

Table 9: Different Prediction Functions Affect on Overall Dev Scores

6 Conclusion

In summary, Unsupervised Contrastive Learning (UCL) and Supervised Contrastive Learning (SCL) improved performance on all three tasks compared to the baseline without extensions. However, it is surprising that UCL performs better than SCL on its own, and SCL combined with UCL in multiple forms. We suspect that SCL might be overfitting to the Qurora Dataset, or the learned embeddings are not helpful to the downstream tasks because the objective of contrastive learning is not perfectly aligned with the objective of the downstream tasks. Other significant improvements came from implementing concurrent multitask fine-tuning (without gradient surgery), decreasing the number of epochs, and utilizing model ensembling. There was also room for improvement in how we formulated our prediction functions in the downstream tasks. The experimentation between using cosine similarity and linear layers for the prediction functions highlighted that benefits could also be achieved by taking a closer look at the foundational aspects of the model rather than only looking toward new extensions.

References

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. MTRec: Multi-task learning over BERT for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, Dublin, Ireland. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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 *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, pages 5824–5836. Curran Associates, Inc.