# A Sentence-BERT Extension to the minBERT Model

Stanford CS224N Default Project

**Shruti Sridhar, Lisa Xuejie Yi**
Department of Computer Science
Stanford University
`shrutisr@stanford.edu, xuejieyi@stanford.edu`

## Abstract

In this project, we aim to fine-tune the minBERT model to simultaneously perform well on sentiment analysis, paraphrase detection, and semantic textual similarity (STS) prediction tasks. First, we use pre-trained weights loaded into our minBERT implementation and train only for the sentiment task to obtain baseline performance metrics for all three downstream tasks. Second, we train for all three tasks at once, using multi-task finetuning and gradient surgery to finetune our embeddings. We take an approach inspired by Sentence-BERT (SBERT) to generate embeddings that can be compared via cosine similarity for the STS task, addressing the overhead of computing pairwise similarities with BERT. Overall, our finetuned embeddings outperform our baseline on two out of the three tasks.

## 1 Key Information to include

- Mentor: Tathagat Verma
- External Collaborators (if you have any): None
- Sharing project: No

## 2 Introduction

In this project, we are interested in exploring how to finetune sentence embeddings from BERT to simultaneously perform well across three downstream tasks: sentiment classification, paraphrase detection, and evaluating semantic textual similarity. For STS, the traditional BERT's state-of-the-art performance entails a massive computational burden, since it requires that both sentences are fed into the network. For instance, if we have a collection of $n = 10,000$ sentences, then inferring each of the $49,995,000$ pairwise similarities to identify the most semantically-similar pair takes over 65 hours on a modern GPU. Thus, traditional BERT remains ill-suited to large-scale semantic search tasks, such as finding the most similar existing query to a new query on Quora, as well as unsupervised clustering. Reimers and Gurevych introduced Sentence-BERT, an approach to generating sentence embeddings using siamese BERT-networks that reduces the compute time for identifying the most similar sentence pair from over 65 hours to just 5 seconds with SBERT, while maintaining BERT's accuracy (Reimers and Gurevych, 2019). For our project, we aimed to implement a similar siamese network to generate SBERT embeddings that can be compared via cosine similarity and achieve better performance on the STS prediction task. In addition, we explored whether multi-task finetuning and gradient surgery could help bolster our model's performance on sentiment classification, paraphrase detection, and STS all at once. After implementing our baseline minBERT model and finetuning the embeddings for all three tasks, we found that multi-task finetuning with or without gradient surgery, coupled with the SBERT siamese network approach for evaluating STS via cosine similarity, yielded embeddings that perform fairly well across all three downstream tasks.

## 3 Related Work

Reimers and Gurevych introduced a siamese network structure to learn Sentence-BERT embeddings that can be optimally and efficiently compared via cosine similarity for the STS task (Reimers and Gurevych, 2019). This structure feeds in two sentences into two identically-configured BERT subnetworks, whose parameters are tied such that any updates are reflected in both subnetworks. Thus, the model can learn embeddings that place semantically-similar sentences closer together.

In addition to Sentence-BERT, several sentence embedding approaches have been proposed in recent years, including InferSent, Universal Sentence Encoder (USE), Quick-Thoughts, USE-QA and Whiteningbert. InferSent (Conneau et al., 2017) uses a bi-directional LSTM to encode sentences into fixed-length vectors, while USE (Cer et al., 2018) employs a deep averaging network (DAN) for encoding. Quick-Thoughts (Logeswaran and Lee, 2018) is based on the skip-thoughts model and uses a uni-directional LSTM to encode sentences into fixed-length vectors. Finally, USE-QA (Guo et al., 2021) is a question-answering model that uses the Universal Sentence Encoder to encode questions and answers into fixed-length vectors and has been shown to achieve state-of-the-art results on several benchmark datasets. Whiteningbert (Huang et al., 2021) is a preprocessing technique for BERT-based models that aims to enhance the model's performance while reducing its computational complexity. The technique is based on a mathematical operation called whitening, which transforms the data in a way that makes it easier for the model to learn.

Choi et al. (2021) highlights the potential of using modified versions of BERT, such as Sentence-ALBERT (SALBERT) and SBERT, in improving sentence representation for various NLP tasks. In a more recent work, Seo et al. (2022) introduced Token Attention Sentence-BERT, which incorporates token-level attention mechanisms, which allow the model to assign different weights to each token in a sentence based on its importance.

Out of all these methodologies, our work chooses to replicate the SBERT architecture since it is most relevant for the STS prediction task. Our work thus serves as a useful next step in evaluating how SBERT, in combination with other multi-task prediction model architectures, might perform on STS as well as other downstream tasks, namely sentiment classification and paraphrase detection.

## 4 Approach

After implementing our minBERT model and AdamW optimizer (refer to project handout for details), we used this as a baseline model to perform sentiment classification. For this task, we obtained the pooled representation of the final BERT embedding output, namely the `CLS` token hidden state. Then, we projected that embedding using a `linear` layer to generate logits for each of the five possible sentiment labels (Figure 1). We also developed baseline versions of the paraphrase detection and semantic textual similarity classifier heads. In particular, for both heads, we applied `dropout` to the pooled BERT embeddings for both input sentences, applied a `linear` layer to project both of them to a single dimension, and computed the cosine similarity, producing a logit in range $[-1,\ 1]$.

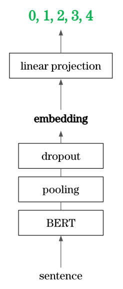

Figure 1: Diagram of model architecture for the sentiment classification subtask head

For our extensions to the baseline minBERT model, we experimented with two different architectures for the paraphrase detection subtask (Figure 2). First, we applied `dropout` to the pooled BERT embeddings of both sentences and concatenated the results. We then applied a `linear` layer to down-project the combined embedding to a single dimension, generating a single logit. This architecture was paired with a binary cross-entropy (BCE) loss function, where for input probability $x_n$ and target label $y_n$, $\ell_n = (y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n))$ . To be compatible with BCE loss, we normalized the logit to the range $[0,\ 1]$ via the sigmoid function. As a second approach, we applied `dropout` to the pooled BERT embeddings, and then a `linear` layer that preserved the embedding dimensions. This layer was included primarily to provide the model with additional learnable parameters. Finally, we computed the cosine similarity between the resulting pair of embeddings, again normalizing the logit to $[0,\ 1]$ for using the BCE loss function.

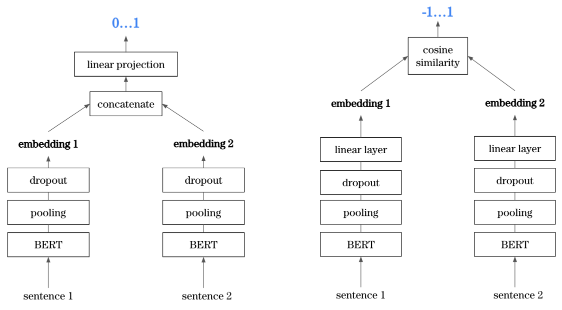

Figure 2: Two alternative architectures for the paraphrase detection subtask head

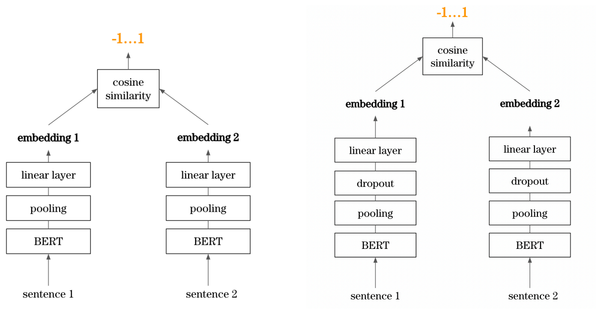

Figure 3: Two alternative architectures for the STS prediction subtask head

We similarly experimented with two model architectures for the STS prediction head (Figure 3). In the first variant, we took inspiration from the SBERT siamese network approach: we applied a `linear` layer to both sentences' pooled embeddings, preserving their dimensions, and then computed cosine similarity, generating a logit in the range $[-1,\ 1]$. Our second architecture was identical, except that we applied `dropout` prior to the `linear` layer on both embeddings. By comparing these two alternatives, we wanted to gauge whether `dropout` would help the embeddings better generalize to unseen examples for the STS prediction subtask. For both architectures, we used a mean squared error (MSE) loss function, which measures the element-wise MSE for an input-target pair $(x_n,\ y_n)$: $\ell_n = (x_n - y_n)^2$. To be consistent with this loss, we rescaled the targets to be in the $[-1,\ 1]$ range.

In the training process, we experimented with multi-task finetuning, where the total loss is computed as the sum of the subtask losses (Bi et al., 2022): $\mathcal{L}_{total} = \mathcal{L}_{sent} + \mathcal{L}_{para} + \mathcal{L}_{sts}$. In addition, we applied gradient surgery, which detects if gradient directions of different subtasks conflict, and projects the gradient of the $i$-th task $\mathbf{g_i}$ onto the normal plane of a conflicting task's gradient $\mathbf{g_j}$:

$$\mathbf{g_i} = \mathbf{g_i} - \frac{\mathbf{g_i} \cdot \mathbf{g_j}}{||\mathbf{g_j}||^2} \cdot \mathbf{g_j}$$

We incorporated this technique so that the learned embeddings would not be biased towards minimizing loss on one task at the expense of the other two subtasks, due to diverging gradients. We applied the PyTorch `pcgrad` implementation to our AdamW optimizer (Yu et al., 2020; Tseng, 2020).

## 5 Experiments

### 5.1 Data

For the sentiment analysis classification task, we used the SST dataset[1], containing $11,855$ sentences from movie reviews, annotated with one of five sentiment labels: 0 (negative), 1 (somewhat negative), 2 (neutral), 3 (somewhat positive), or 4 (positive). We also used the CFIMDB dataset of 2434 highly-polar movie reviews, labeled as negative or positive. For the paraphrase detection

[1]https://nlp.stanford.edu/sentiment/treebank.html

task, we used the Quora dataset[2], containing $400,000$ question pairs labeled by $0$ or $1$, indicating whether they are paraphrases of each other. For the STS prediction task, we used the SemEval STS Benchmark dataset[3], containing $8628$ sentence pairs labeled by degree of similarity from $0$ to $5$.

### 5.2 Evaluation method

For the sentiment classification and paraphrase detection tasks, we evaluated the accuracy of our predicted labels against the gold labels on each of the train/dev/test datasets. For STS performance, we measured the Pearson correlation coefficient between the predicted cosine similarity values and the actual similarity labels (rescaled from $[0, 5]$ to $[-1, 1]$ to align with predicted cosine similarities).

### 5.3 Experimental details

We first used our minBERT model's pooled embeddings to perform sentiment classification. We ran the classifier in pretrain mode (LR = 1e-3) and finetune mode (LR = 1e-5) each for 10 epochs. In pretrain mode, the classifier loads in the pre-trained BERT weights and only updates parameters in these last two additional layers. In finetune mode, the classifier updates BERT's parameters in addition to these added layers. We also ran the multi-task classifier for 10 epochs in pretrain and finetune mode, including just SST data in the training loop with a batch size of 64. We evaluated all three tasks using the baseline paraphrase and STS heads, obtaining baseline accuracies for all tasks.

We then conducted several experiments to evaluate the impact of various extensions on the model's performance on all three downstream tasks. In Experiment 1, we performed the multi-task finetuning with gradient surgery, BCE loss for the paraphrase detection task, MSE loss for the STS task. In Experiment 2, we followed the same routine, but without gradient surgery. In Experiment 3, we used the same routine as Experiment 1, but tried the alternative architecture for the paraphrase detection head, which performs cosine similarity instead of concatenating the sentence embeddings. Finally, for Experiment 4, we used the same routine as Experiment 1, but with the alternative architecture for the STS prediction head, which includes an additional `dropout` layer before the `linear` layer.

For our training loop, we utilized a round-robin approach: on every epoch, we trained on consecutive batches of the SST, Quora, and SemEval datsets. Since the Quora datset is significantly larger than the other two datasets, we first tried iterating through each of the datasets just once over. With this method, we performed gradient surgery with the available subtasks (whose datasets hadn't yet run out) at that given iteration: either all three, just sentiment and paraphrase, or just paraphrase. We also explored repeatedly iterating through the SST and SemEval datasets until we completed one full pass over the Quora dataset. In this case, we always applied gradient surgery to all three tasks.

For all experiments, we trained for 15 epochs in finetune mode (LR = 1e-5), with a batch size of 32 across all three datasets. Therefore, we did not need to re-weight any of the multi-task loss terms.

### 5.4 Results

#### 5.4.1 Baseline minBERT Implementation Results

| **Dataset** | **Pretrain** | **Finetune** |
|---|---|---|
| SST | 0.390 | **0.523** |
| CFIMDB | 0.780 | **0.976** |

Table 1: Sentiment classification dev accuracies

For the SST and CFIMDB datasets, the sentiment classifier performed better in finetune mode compared to pretrain mode. This result was expected, as finetune mode allows for gradient updates to be pushed back to BERT's parameters themselves. When trained only on SST data and evaluated using the baseline paraphrase and SST heads, the multitask classifier achieved high dev performance on the sentiment task, but lower performance for the other two tasks. These results were to be expected,

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

[3]http://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark

| **Task** | **Pretrain** | **Finetune** |
|---|---|---|
| Sentiment Classification | 0.401 | **0.534** |
| Paraphrase Detection | 0.376 | **0.392** |
| STS Evaluation | 0.019 | **0.312** |

Table 2: Baseline multi-task dev performance

since we didn't yet train for these downstream tasks. Once again, we observed that finetune mode yielded better performing embeddings. Thus, for our extensions, we only trained in finetune mode.

#### 5.4.2 Model Extension Experiment Results

| **Task** | **Exp 1** | **Exp 2** | **Exp 3** | **Exp 4** |
|---|---|---|---|---|
| Sentiment Classification | **1.000** | 1.000 | 0.998 | 1.000 |
| Paraphrase Detection | 0.990 | **0.991** | 0.969 | 0.988 |
| STS Evaluation | **0.991** | 0.988 | 0.990 | 0.976 |

Table 3: Multi-task finetuning train performance

| **Task** | **Exp 1** | **Exp 2** | **Exp 3** | **Exp 4** |
|---|---|---|---|---|
| Sentiment Classification | 0.490 | **0.492** | 0.489 | 0.490 |
| Paraphrase Detection | **0.777** | 0.772 | 0.443 | 0.770 |
| STS Evaluation | **0.592** | 0.584 | 0.590 | 0.460 |

Table 4: Multi-task finetuning dev performance

| **Task** | **Exp 1** |
|---|---|
| Sentiment Classification | 0.508 |
| Paraphrase Detection | 0.785 |
| STS Evaluation | 0.569 |

Table 5: Multi-task finetuning test performance (best model only)

These results were obtained from repeatedly iterating through the SST and SemEval datasets until we completed one full pass over the Quora dataset on each epoch. We didn't observe any performance changes when we used the other training approach, where we completed just one full pass of each dataset. The repeated iteration approach allowed for faster convergence, but the training time per epoch was considerably longer (on Colab GPU, 40+ minutes vs. 15 minutes with only one full pass).

After training the four model configurations outlined in Section 5.3 in finetune mode, we found that the model in Experiment 1 (multi-task finetuning with gradient surgery) performed the best overall. This model exceeded the baseline performance for both paraphrase detection (0.777 accuracy) and STS prediction (0.592 correlation) on the respective dev datasets. The accuracy for sentiment classification (0.490) was slightly lower than our baseline dev accuracy (0.534). However, this minor decline is reasonable, as the baseline results were obtained after just training on the SST dataset. After training on all three datasets, the learned sentence embeddings might have been adjusted so as to perform slightly worse on the sentiment task, but to minimize the multi-task loss holistically. Overall, these accuracies were generally aligned with our expectations for multi-task finetuning.

We observed a very minimal difference between the performance in Experiment 1 and Experiment 2, where we omitted gradient surgery. This was surprising, as we anticipated that gradient surgery would resolve any conflicting gradients during each update, thus providing a nontrivial improvement over standard multi-task finetuning. One potential reason for this is that gradient surgery was not able to resolve three conflicting gradients at once. Interestingly though, we did not observe a difference in results when we applied it to just 2 out of 3 tasks. This suggests that regardless of the number of tasks involved, gradient surgery might not have been able to meaningfully resolve these conflicts.

For Experiment 3, where we used the alternative paraphrase head architecture (with cosine similarity instead of concatenation), we observed a significant decline in the paraphrase detection accuracy (0.443 dev) compared to the best performance from Experiment 1 (0.777 dev). This suggests that applying `linear` layers to both embeddings independently (like in our STS prediction head) is less suitable for the paraphrase task than applying a single `linear` down-projection layer to the concatenated embeddings. This might be because the paraphrase detection task is much simpler than STS prediction: it's a binary classification task, whereas the STS categories are more graded.

For Experiment 4, where we inserted a `dropout` layer before the `linear` layer for both embeddings in our STS prediction head, we observed a slight decline in STS correlation (0.460 dev) compared to the best result from Experiment 1 (0.592 dev). The final train correlation was also slightly lower (0.976) compared to Experiment 1 (0.991). This result was unexpected, as we thought that Dropout would help our embeddings generalize and achieve higher dev set performance. However, this might have been because 15 epochs wasn't enough time for the network to truly converge with Dropout. Dropout usually leads to poorer performance in early training, but better loss after convergence.

Since it had the best average performance for all three tasks, we submitted our Experiment 1 model to the test leaderboard and achieved comparable performance to the dev scores. The sentiment and paraphrase test scores were slightly higher, and the STS test score was slightly lower than the dev.

## 6 Analysis

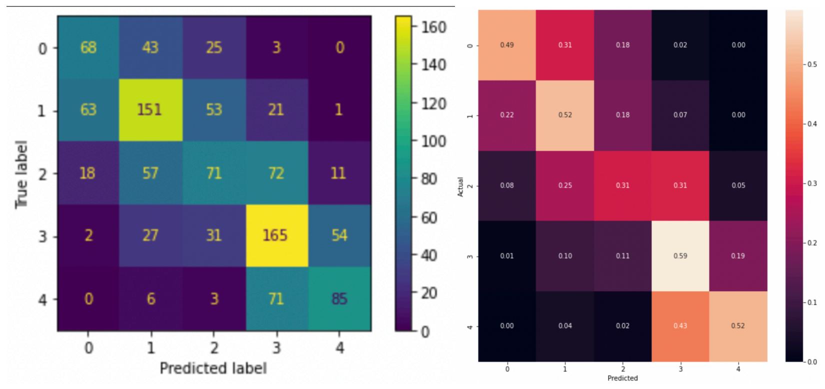

Figure 4: raw and normalized confusion matrix for sentiment classification (SST dev; Exp 1 model)

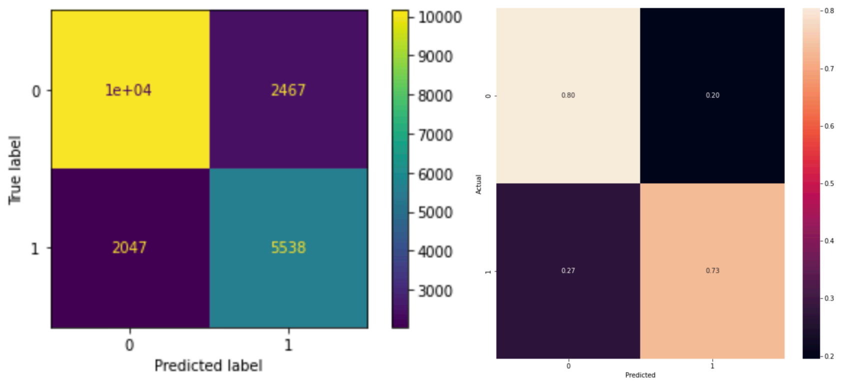

Figure 5: raw and normalized confusion matrix for paraphrase detection (Quora dev; Exp 1 model)

In Figure 4, we see common error patterns of our model on the sentiment classification task. In particular, our current model frequently misclassifies nearby labels. For instance, it has trouble distinguishing "negative" (label 0) from "somewhat negative" (label 1) examples in the dev dataset.

Over 30% of true "negative" examples are predicted as "somewhat negative", and over 20% of true "somewhat negative" examples are predicted as "negative". There is a similar pattern for "positive" (label 4) vs. "somewhat positive" (label 3) examples. For true "positive" examples, the model predicts a label of "somewhat positive" over 40% of the time. Additionally, for true "neutral" examples (label 2), the model frequently misclassifies them as "somewhat positive" or "somewhat negative". These errors show that the SST dataset and current training routine might have been insufficient to learn all the fine-grained distinctions among these classes and generalize beyond the training set.

In Figure 5, we see common error patterns for the paraphrase detection task. Overall, the false positive rate (20%) and false negative rate (27%) are comparable. The larger false negative rate might be due to an imbalance of examples in the Quora dev dataset: as seen in the raw confusion matrix, there are many more true negatives (non-paraphrases) than there are positives (paraphrases).

## 7 Conclusion

In this project, we have explored several techniques to enable our minBERT embeddings to simultaneously perform well on three downstream sentence classification or regression tasks: sentiment classification, paraphrase detection, and STS evaluation. With multi-task finetuning, gradient surgery, and the SBERT siamese network and cosine similarity methodology, we were able to achieve higher performance than our baselines on the paraphrase detection and STS tasks. Our sentiment accuracy, while slightly lower, remained comparable to our baseline model's, which was only trained for the sentiment task. We found that the inclusion of gradient surgery didn't impact performance, and adding additional complexity to the subtask prediction head architectures didn't always yield performance improvements, especially if the subtask was quite simple. Overall, our learned embeddings are able to perform fairly well across all three tasks when evaluated on the dev and test datasets. The primary limitation of our current model is the STS correlation. For future steps to augment the performance on this subtask, we might consider additional pretraining with STS data and a Masked Language Model learning objective. Additionally, we might explore whether a triplet loss function or multiple-negatives ranking loss function yields better results than just MSE loss.

## 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.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Hyunjin Choi, Judong Kim, Seongho Joe, and Youngjune Gwon. 2021. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In *2020 25th International conference on pattern recognition (ICPR)*, pages 5482–5487. IEEE.

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*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Mandy Guo, Yinfei Yang, Daniel Cer, Qinlan Shen, and Noah Constant. 2021. Multireqa: A cross-domain evaluation forretrieval question answering models. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 94–104.

Junjie Huang, Duyu Tang, Wanjun Zhong, Shuai Lu, Linjun Shou, Ming Gong, Daxin Jiang, and Nan Duan. 2021. Whiteningbert: An easy unsupervised sentence embedding approach. *arXiv preprint arXiv:2104.01767*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *ArXiv*, abs/1803.02893.

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.

Jaejin Seo, Sangwon Lee, Ling Liu, and Wonik Choi. 2022. Ta-sbert: Token attention sentence-bert for improving sentence representation. *IEEE Access*, 10:39119–39128.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

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*.