Multitask Finetuning on BERT using Gradient Surgery and Linear Probing before Finetuning

Stanford CS224N Default Project

Arvind Mahankali
Department of Computer Science
Stanford University
amahanka@stanford.edu

Abstract

We study multitask finetuning of BERT to obtain good performance on sentiment analysis (SA), paraphrase detection (PD), and semantic textual similarity (STS). We also study the effect of gradient surgery (PCGrad) and linear probing followed by finetuning (LP-FT) on the dev set performance. We find that PCGrad improves performance on the STS task without significantly worsening performance on the other tasks, and conjecture that it makes better use of the shared structure between PD and STS. We also find that LP-FT worsens performance on the STS task, which we hypothesize is due to the linear probing phase becoming stuck in a region with low gradient norm.

1 Key Information

- Mentor: Gabriel Poesia

2 Introduction

The goal of this project is to understand the effectiveness of various optimization techniques when doing multitask learning with transformers on natural language tasks. Specifically, we aim to train a single transformer with a BERT backbone (Devlin et al. (2019)) to achieve good performance on (1) sentiment analysis (SA), (2) paraphrase detection (PD), and (3) semantic textual similarity (STS). In sentiment analysis, the input is a sentence, and the output is an integer between 1 and 5 that signifies whether the sentence conveys a positive emotion or a negative emotion. In paraphrase detection, the input is a pair of sentences, and the goal is to determine whether the two sentences are paraphrases of each other. Finally, in semantic textual similarity, the goal is, given a pair of sentences, to determine how closely related the two sentences are (represented by a scalar).

Multitask learning can present optimization challenges. The work of Yu et al. (2020) demonstrates in multi-task supervised and reinforcement learning tasks that optimization can fail to make progress when the gradients from the different tasks conflict (i.e. the dot products are negative) under certain additional conditions which they refer to as the "tragic triad." To address this issue, they propose the PCGrad algorithm, which (roughly speaking) modifies the gradients from the different tasks by making them orthogonal. The first question this project seeks to answer is whether PCGrad can improve test accuracy on the three tasks mentioned above, compared to ordinary multitask training.

Additionally, we will investigate a complementary approach known as LP-FT (studied by Kumar et al. (2022) in the context of out-of-distribution generalization). Kumar et al. (2022) find that, for pre-trained models with task-specific linear heads attached, training the linear head prior to performing full fine-tuning will lead to improved performance when the model is evaluated on data which is different from the finetuning data distribution. Similarly, we may expect that finetuning the BERT backbone with respect to one task, say sentiment analysis, can distort the embeddings of the paraphrase detection examples, if the paraphrase detection examples come from a different

distribution than the sentiment analysis examples. Thus, we also investigate whether LP-FT can help improve the training process and improve test error.

3 Related Work

One of the approaches we apply, LP-FT, is extensively studied theoretically and empirically in Kumar et al. (2022), which shows that LP-FT achieves much better out-of-distribution performance than regular finetuning, without sacrificing in-distribution performance. While we are not concerned with out-of-distribution performance in this project, this technique is still potentially relevant to us if the data for the different tasks comes from different distributions, as discussed above. In addition, a natural question is whether training the linear heads first, before finetuning, can provide a better initialization compared to a random initialization. We note that the later work by Lee et al. (2022) studies the effect of fine-tuning a more general subset of layers on out-of-distribution generalization — it is an interesting question for later work whether such a technique could be applied to multitask training. The other approach that we apply, PCGrad, was initially proposed by Yu et al. (2020) — they empirically investigate it in the context of multitask supervised learning (on computer vision tasks) and reinforcement learning. Here we will determine whether it can lead to performance benefits in NLP tasks as well.

4 Approach

Basic Multitask Approach First, we describe our basic multitask learning architecture/algorithm. To each sentence $x \in \mathbb{R}^{T \times |V|}$ (where T is the length of the sentence and $|V|$ is the vocabulary size), we prepend a [CLS] token — the context-dependent BERT embedding $f_\theta(x) \in \mathbb{R}^d$ of this [CLS] token (where θ represents the parameters of the BERT model) is the pretrained representation to which the linear heads for each of the tasks is applied.

For sentiment analysis, we use a linear head $W_1 \in \mathbb{R}^{d \times 5}$, which we directly apply to $f_\theta(x)$. We finetune W_1 and $f_\theta(x)$ together on the sentiment analysis task using the cross-entropy loss function. For paraphrase detection, given two sentences $x \in \mathbb{R}^{T_1 \times |V|}$ and $y \in \mathbb{R}^{T_2 \times |V|}$, we apply a linear head W_2 to both $f_\theta(x)$ and $f_\theta(y)$. With this linear head, we obtain an unnormalized logit L, which is converted to a number between 0 and 1 using a sigmoid function — we train on paraphrase detection using the binary cross entropy loss. To obtain the logit L, we first take the cosine similarity of $W_2 f_\theta(x)$ and $W_2 f_\theta(y)$ (where for two vectors u and v, their cosine similarity is defined as $\frac{u \cdot v}{\|u\|_2 \|v\|_2}$). We then apply ReLU (the effect being that sentences with similarities less than 0 are treated the same). Then, we subtract 0.5 and multiply by 5, so that $\sigma(L)$ can represent a wide range of probabilities.

For semantic textual similarity, we apply a similar technique to obtain a scalar between 0 and 5 as the output. If $W_3 \in \mathbb{R}^{d \times d}$ is our linear head for semantic textual similarity and the input sentence pair is (x, y), then we first take the cosine similarity of $W_3 f_\theta(x)$ and $W_3 f_\theta(y)$. We subtract 0.5 from this cosine similarity, then multiply by 5 and apply the sigmoid function. Finally, we multiply the result by 5. Here we simply use the mean squared error loss.

Batches for Multitask Approach We interleaved the minibatches from the SA, PD and STS datasets as follows. Note that the PD dataset had the most examples. We simultaneously iterated through the batches from the SA, PD and STS datasets in order, until we reached the end of the PD `DataLoader`. Whenever we reached the end of the SA/STS `DataLoaders`, we re-started from the beginning. At each iteration, we drew one batch from the SA `DataLoader` and did an AdamW step using the cross-entropy loss, then took one batch from the PD `DataLoader` and did an AdamW step using the binary cross-entropy loss, and finally took one batch from the STS `DataLoader` and did an AdamW step using the mean squared error loss. We interleave the batches when applying PCGrad (described below).

PCGrad (Yu et al. (2020)) We implement PCGrad in a manner similar to the algorithm described in Algorithm 1 of Yu et al. (2020). First, we draw batches from the three tasks as described above. For each of the batches, we compute the gradients for the parameters due to the respective loss functions — we denote these gradients by g_{SA}, g_{PD} and g_{STS} respectively. Then, we define g_{SA}^{PC} as follows. (1) Initially, g_{SA}^{PC} is equal to g_{SA}. (2) Then, if g_{SA}^{PC} conflicts with g_{PD}, we subtract from g_{SA}^{PC} the

orthogonal projection of g_{SA}^{PC} onto g_{PD}. (3) If g_{SA}^{PC} still conflicts with g_{STS}, then we subtract from g_{SA}^{PC} its orthogonal projection onto g_{STS}. Note that we actually perform steps (2) and (3) in random order as recommended in Yu et al. (2020). In a similar manner, we obtain g_{PD}^{PC} and g_{STS}^{PC}. The final gradient for this batch with respect to the parameters is $g_{SA}^{PC} + g_{PD}^{PC} + g_{STS}^{PC}$, which is used as the gradient that is given as input to AdamW.

LP-FT First, we train the linear heads on their respective tasks for several epochs. Then, we fully finetune the model as in the basic multitask approach.

Separate-task Finetuning Baseline As a baseline, we also separately finetuned BERT on the three tasks, without any shared parameters. For each task, we used linear heads in the same manner as described in the basic multitask approach.

5 Experiments

5.1 Data

For sentiment analysis, we used the Stanford Sentiment Treebank (SST) dataset Socher et al. (2013). In this dataset, the inputs are sentences, and the outputs are scores which are integers between 1 and 5, signifying whether the emotion of the sentence is positive or negative.

For paraphrase detection, we use a dataset provided by Quora Iyer et al.. In this dataset, the inputs are pairs of sentences, and the outputs are binary labels (i.e. 0 or 1) that signify whether the sentences are paraphrases of each other.

Finally, in the semantic textual similarity task Agirre et al. (2013), the inputs are pairs of sentences, and the output is a scalar between 0 and 5 which signals how closely related the sentences are.

5.2 Evaluation method

Our evaluation metrics for SA and PD are simply classification accuracy. For the STS dataset, our evaluation metric is the Pearson correlation between the model outputs and the true labels. Specifically, we apply the model to each of the examples in the dev set — then, we take the covariance between the model outputs and the true labels, and then divide by the standard deviation of the model logits and the standard deviation of the correct labels.

5.3 Experimental details

The experimental configurations are as follows. The learning rate was $1e-5$ for finetuning, and when applying LP-FT, we used a learning rate of $1e-3$ for the initial linear probing phase. We used 5 epochs when finetuning jointly on all the datasets. For linear probing, 10 epochs were used for SA/STS, and 5 epochs were used for PD. For the basic multitask approach, we used a batch size of 64. For PCGrad, we used a batch size of 32 due to memory constraints. We additionally run the basic multitask approach with a batch size of 32 to compare with PCGrad.

While finetuning, at the end of every epoch, we took the average of the dev set accuracies on the three tasks, and saved the model which had the highest average accuracy. Here, one epoch is defined as making one iteration through all of the batches of PD (while simultaneously drawing one batch for SA/STS per each batch of PD). The dev set accuracy for the STS task is defined as follows — we take the logit output by the model and round it to the nearest integer. If the rounded logit is equal to the correct label, then we say the model is correct, and otherwise we say it is incorrect.

5.4 Results

First, we show the results on the test sets in Table 1. We only included the predictions of these three models, since they were the best performing models on the dev set. We also include the results for the dev sets in Table 2.

In summary, the basic multitask approach performs well, and PCGrad leads to a significant advantage over the basic multitask approach on the STS task without suffering significantly on the other tasks —

Model	SST Accuracy	PD Accuracy	STS Correlation
Basic Multitask (Batch Size 64)	0.514	0.827	0.813
Basic Multitask (Batch Size 32)	0.512	0.844	0.812
PCGrad (Batch Size 32)	0.512	0.835	0.822

Table 1: Results on test set.

Model	SST Accuracy	PD Accuracy	STS Correlation
Basic Multitask (Batch Size 64)	0.503	0.829	0.821
PCgrad (Batch Size 32)	0.507	0.838	0.837
Basic Multitask (Batch Size 32)	0.504	0.844	0.813
PCgrad with LPFT (Batch Size 32)	0.496	0.804	0.620
Basic Multitask with LP-FT	0.507	0.810	0.665
PD-only finetuning (5 epochs)	N/A	0.845	N/A
SST-only finetuning (10 epochs)	0.525	N/A	N/A
STS-only finetuning (10 epochs)	N/A	N/A	0.770

Table 2: Results on dev set for all models. By default, methods use batch size 64 unless otherwise noted.

this is the case on both the test and dev sets. We note that the LP-FT models performed quite poorly on the STS task — we study this further in the next section.

Note that on the STS task, the multitask models (aside from the LP-FT models) achieved significantly higher performance then the baseline model that was finetuned only on STS. It is quite possible that this is largely due to a difference in training time, since the multitask models are trained for the equivalent of many epochs on the STS task (every epoch of the multitask training loop corresponds to several epochs of STS, since the PD dataset is much larger than the STS dataset). However, there is still a possibility that the improved performance is due to shared structure between the tasks, since after a certain number of epochs, the dev set accuracy of the baseline model finetuned only on STS did not improve (see Figure 1), suggesting that it might not simply be due to the multitask models having more running time.

6 Analysis

PCGrad v.s. Basic Multitask Approach In this section, we attempt to analyze the PCGrad approach and the basic multitask approach, and understand the improved performance of the PCGrad approach on the STS task.

It is possible that the improved performance of PCGrad on the dev set is because PCGrad makes better use of shared structure between the STS and PD tasks. As shown in the first two images of Figure 2, near the beginning of the process, the STS and PD gradients require much larger adjustments than the

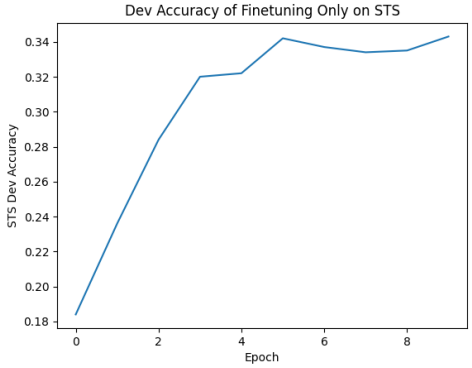

Figure 1: Dev set accuracy of baseline model that is finetuned only on STS

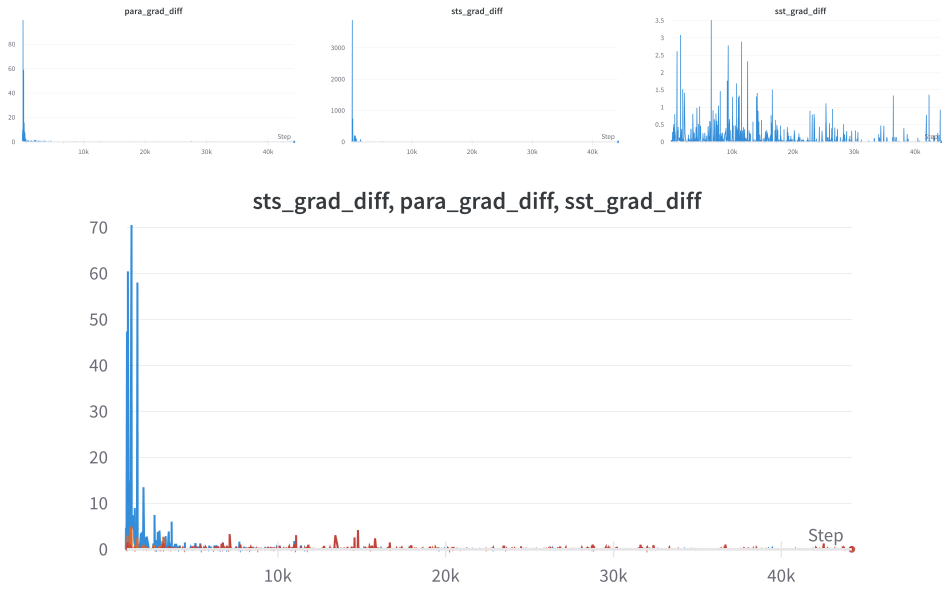

Figure 2: The first image shows the difference between the unmodified PD gradient g_{PD} and the modified PD gradient g_{PD}^{PC} throughout the PCGrad training run. The second image shows the difference between the unmodified STS gradient g_{STS} and the modified STS gradient g_{STS}^{PC} throughout the process. The third image shows the difference between g_{SST} and g_{SST}^{PC} throughout the process. The last graph shows all three plotted on the same graph (after several time steps have passed) — the blue curve represents the adjustment to the STS gradient, the orange (barely visible) representing the adjustment to the PD gradient, and the red representing the adjustment to the SA gradient. Observe that in the last image, after the first few thousand steps, the red curve (i.e. the adjustment to the SA gradient) is significantly larger than the adjustments to the other tasks' gradients.

	PD Dataset	STS Dataset
Basic Multitask Model	0.964	0.959
PCGrad Model	0.972	0.966

Table 3: Correlation between cosine similarity obtained from PD head and STS head. Evaluated for the PCGrad model and the basic multitask model, using sentence pairs from PD dataset and STS dataset.

SST gradient — the most likely conclusion from this is that the STS and PD gradients conflicted with each other. Afterwards, as shown in the last image of Figure 2, the adjustments to the STS and PD gradients are relatively small. In summary, Figure 2 suggests that throughout the process, PCGrad only allows updates in directions which are conducive to both STS and PD.

The above observations are intuitively reasonable because STS and PD are similar tasks — the goal is to determine whether two sentences are similar. Furthermore, for both tasks, we compute the logits in a similar manner, using cosine similarity which is then scaled appropriately. In order to support the conjecture that the PCGrad performs better due to making better use of shared structure between the STS and PD tasks, we perform the following experiment with the goal of showing that the PD head and the STS head of the PCGrad model behave similarly. For every sentence pair in the STS dev set, we apply the PCGrad model's PD linear head to the sentence pair and then take the cosine similarities of the resulting embeddings. We then take the cosine similarities using the STS linear head as usual. We then compute the Pearson correlation between the cosine similarities computed using the PD head, and the STS head. We then repeat this for the basic multitask model and for the PD dataset.

The results are in Table 3 and Fig 3. As seen in Figure 3, for both the PCGrad and the basic multitask model, there is a clear linear relationship between the outputs of the PD head and the STS head. As seen in Table 3, this relationship is slightly stronger for the PCGrad model.

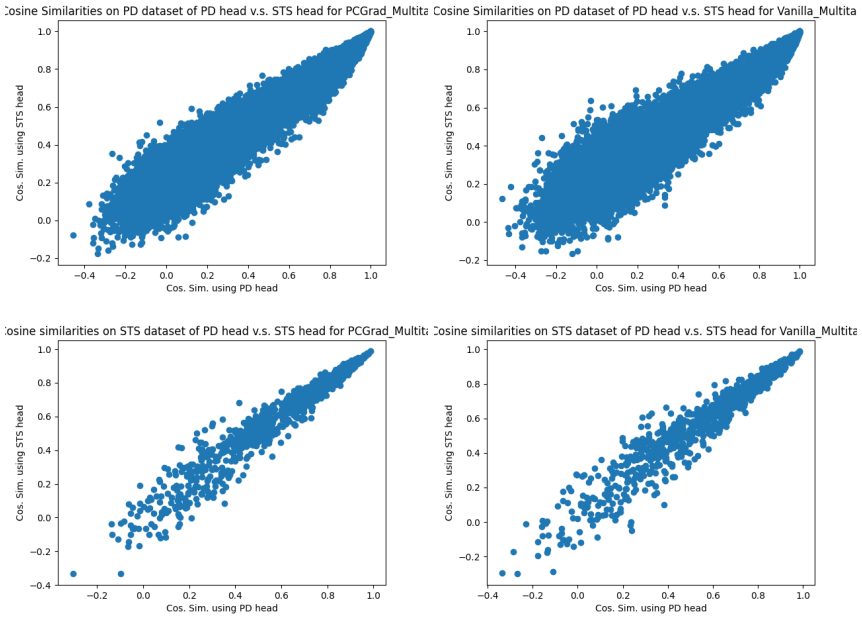

Figure 3: (Upper Left) Scatterplot where the x-axis is the cosine similarity when the PCGrad model's PD head is applied to a pair in the PD dataset, and the y-axis is the cosine similarity when the PCGrad model's STS head is applied to the same pair from the PD dataset. (Upper Right) Scatterplot where the x-axis is the cosine similarity when the basic multitask model's PD head is applied to a pair in the PD dataset, and the y-axis is the cosine similarity when the PCGrad model's STS head is applied to the same pair from the PD dataset. (Lower Left) Scatterplot where the x-axis is the cosine similarity when the PCGrad model's PD head is applied to a pair in the STS dataset, and the y-axis is the cosine similarity when the PCGrad model's STS head is applied to the same pair from the STS dataset. (Lower Right) Scatterplot where the x-axis is the cosine similarity when the basic multitask model's PD head is applied to a pair in the STS dataset, and the y-axis is the cosine similarity when the PCGrad model's STS head is applied to the same pair from the STS dataset.

Effect of Using LP-FT Surprisingly, LP-FT seems to hinder dev set performance according to Table 2, particularly on STS. The following experiments suggest that this is due to LP-FT moving the parameters to a region where the gradients have low norm, making it difficult for the later finetuning process to recover. First, observe in Figure 4 that the train and dev accuracy of the linear probe remains at roughly 0.2, which is the performance achievable by guessing uniformly at random, suggesting that the linear probe does not learn useful information.

Next, to further support this conjecture, we plot the outputs, on the STS train and dev datasets, of the model trained on STS with linear probing — see Figure 5. Compared to the distribution of labels for the STS train and dev sets, the output of the model is skewed heavily towards outputting 5. This means the cosine similarity output by the STS linear head is 1/close to 1 for a large portion of inputs, which may occur if the rank of the STS linear head has become very low. Based on Figure 5, it is likely that the gradient signal to the STS linear head is low, since the sigmoid is applied to the output of the STS linear head, and the output of the STS linear head is in a region where the sigmoid is close to 1. This likely causes the later finetuning process to also not receive as much of a gradient signal for the STS task.

7 Conclusion

We analyze the effect of PCGrad and LP-FT on multitask learning with BERT. We find that PCGrad improved performance on STS without significantly adversely affecting performance on other tasks. However, LP-FT worsened performance on the STS task due to optimization issues. One question for future work is whether it is possible to modify the similarity head/the way similarity is computed to

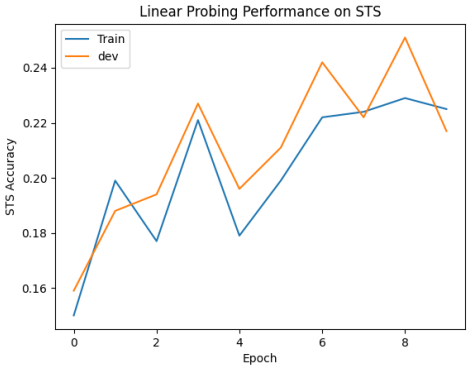

Figure 4: Train and dev accuracy on STS of linear probing.

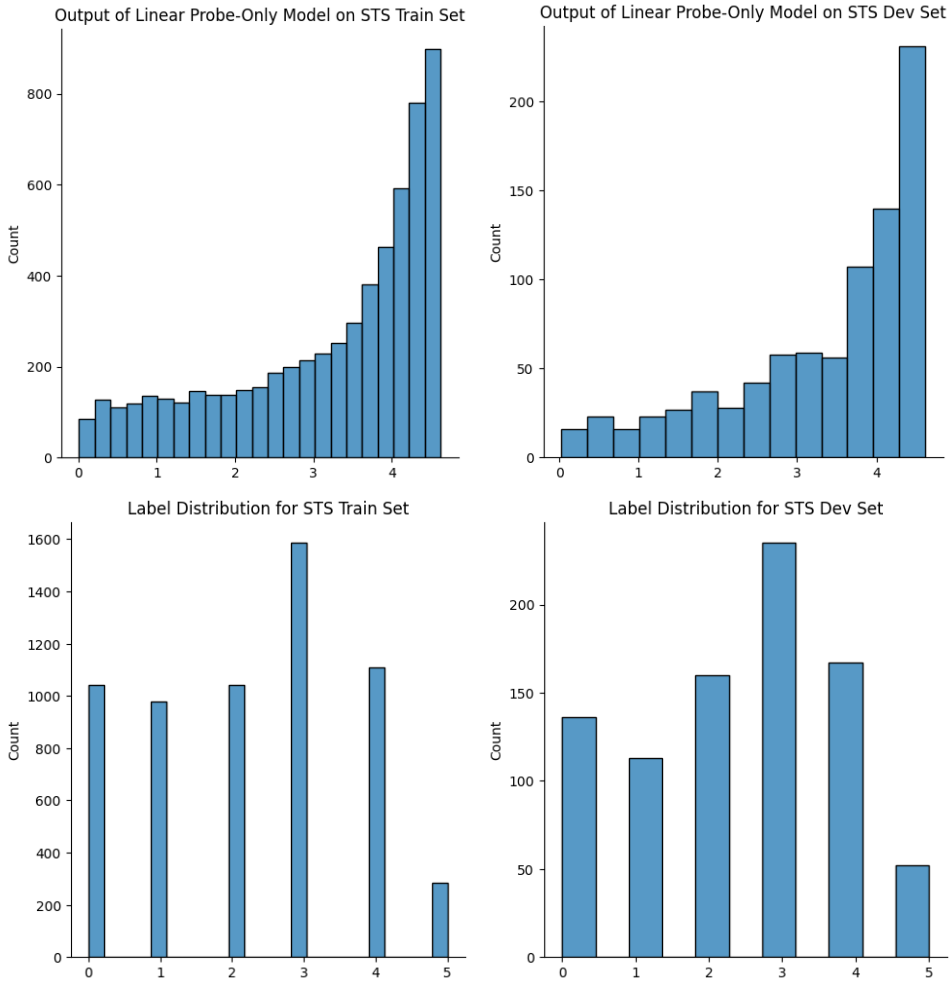

Figure 5: The top two figures show the output of the STS linear probing model on the STS train and dev sets respectively. For comparison, the bottom two figures show the distribution of labels in the STS train and dev sets.

improve the performance of the LP-FT models. Another question for future work is whether surgical finetuning Lee et al. (2022) can improve performance in multitask NLP.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. 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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Shankar Iyer, Nikhil Dandekar, and Csernai Kornel. First quora dataset release: Question pairs.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. 2022. Surgical fine-tuning improves adaptation to distribution shifts. *CoRR*, abs/2210.11466.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.