Unitary Scalarization or Gradient Surgery? Best Practices for Multitask Fine-Tuning

Stanford CS224N Default Project

John McEnany
Biophysics Ph.D. Program
Stanford University
jmcenany@stanford.edu

Abstract

In an effort to build efficient deep learning networks which can be broadly applied to a given field of tasks, many researchers have turned to multitask learning (MTL), where a network is simultaneously trained on multiple tasks with their own loss functions. A simple and efficient way to perform MTL is unitary scalarization, where the multitask loss function is a weighted sum of the loss functions for individual tasks. Some research suggests that more complex multitask optimization methods, such as gradient surgery, lead to more reliable learning by reducing destructive interference between the gradients of different tasks. However, others argue that these methods are unneeded and any observed benefit is just a regularization effect, which can be achieved through more efficient methods. In this work, we extend past research on this topic (which has primarily been performed for image recognition tasks) to the context of natural language processing. By finetuning BERT sentence embeddings on three simple downstream tasks, using both unitary scalarization and gradient surgery, we find that gradient surgery provides minimal benefit over simple unitary scalarization, making it difficult to justify the additional cost in time and memory. However, it may lead to faster learning when model architecture is such that similar embedding features must simultaneously be applied to two different tasks.

1 Key Information to include

- Mentor: Manasi Sharma
- External Collaborators: None
- Sharing project: No

2 Introduction

From a human perspective, an understanding of natural language involves simultaneous, often-seamless integration of many different tasks. These tasks are closely intertwined: for example, understanding the meaning of a word aids in understanding the meaning of a sentence, which is necessary for higher-level tasks such as summarizing a paragraph. This intuition has led researchers in the field of natural language processing (NLP) to construct neural networks capable of multitask learning (MTL), potentially using information gained via training on one task to better perform on another. MTL has further practical benefits: training on specific downstream NLP tasks is often hindered by the limited size of pre-labeled datasets, so training on multiple tasks allows for further training without need for additional task-specific data [1]. MTL is even more tractable thanks to pretrained language models such as Bidirectional Encoder Representations from Transformers (BERT) [2], which are effective starting points for many downstream tasks [1].

The increasing prevalence of MTL in NLP tasks strongly motivates research about how best to structure training regimens for simultaneous learning of many different tasks. In particular, one fundamental question is how to incorporate information about different tasks when making gradient updates. A simple and efficient approach, deemed "unitary scalarization," is simply to minimize the sum of task-specific loss functions, allowing one step of backpropagation to return a gradient update incorporating information from every task [3]. However, the efficacy of this approach is cast into doubt by the phenomenon of gradient interference, where learning of one task hinders the learning of another rather than acting synergistically. For example, training a translation model on multiple languages at once can result in both synergistic and interference effects [4]. As a way to remedy this problem, some researchers have presented "gradient surgery" techniques, which aim to explicitly remove the parts of gradient updates that contribute to interference, while leaving synergistic updates intact [5]. Despite their potential benefits, gradient surgery methods pay a cost for their increased complexity in runtime and memory usage, making it important to test whether they are actually effective [3]. Furthermore, much of the research comparing gradient surgery to unitary scalarization has been performed on image processing tasks [3] [5], leaving it unclear which approach is most efficient in natural language processing.

In this work, we take a step toward addressing the question of whether gradient surgery outperforms unitary scalarization on a simple multitask finetuning problem. We finetune BERT sentence embeddings [2] on three downstream tasks (sentiment analysis, paraphrase detection, and semantic similarity analysis) using both gradient update methodologies. We also use two different network architectures to combine two separate sentence embeddings for the paraphrase detection and semantic similarity tasks, in order to investigate whether destructive gradient interference is more evident when the model uses similar embedding structures for different downstream tasks. We find that gradient surgery does not result in a marked improvement over unitary scalarization, although it may speed up learning when the joint sentence embeddings used for paraphrase detection and semantic similarity analysis are similar. While more research is necessary to reach a firm conclusion (particularly research using models with more than three downstream tasks), we suggest that more efficient methods of modifying model architecture may be at least as effective as gradient surgery.

3 Related Work

Yu et al. propose a gradient surgery method called PCGrad to reduce destructive gradient interference [5]. This method identifies when gradients corresponding to different tasks conflict, and projects conflicting gradient updates to be orthogonal to each other while leaving synergistic updates unchanged. The authors describe a set of circumstances where learning is stalled due to destructive interference between tasks: conflicting gradients, variation in gradient magnitudes, and large local curvature in the loss function. They argue that PCGrad helps learning escape these regions, and support their claim by demonstrating that PCGrad leads to better performance on downstream image processing tasks.

However, some researchers are doubtful of the general applicability of PCGrad and similar methods of modifying single-task gradients. Kurin et al. argue that these "multi-task optimizers" are essentially acting as regularizers, meaning that the performance they achieve is no better than unitary scalarization when combined with traditional, less computationally expensive regularization methods [3]. The computational cost of multi-task optimizers comes from the fact that calculating single-task gradients requires a number of backpropagation steps proportional to the number of tasks, an issue which unitary scalarization avoids. Kurin et al. find that neither PCGrad nor a variety of comparable methods consistently outperform unitary scalarization on a suite of image analysis tasks. The conflict between Yu et al. and Kurin et al. motivates further research: is gradient surgery truly necessary, or is cost-effective unitary scalarization just as good? Moreover, should our conclusions change when we move from the world of image analysis to natural language processing?

4 Approach

As a simple example of multitask learning, we simultaneously train a network on three tasks: sentiment classification, paraphrase detection, and semantic similarity analysis. The backbone of the model is the pretrained BERT model, which generates a vector embedding for a given sentence or phrase [2]. We will not restate in detail the architecture of BERT here, but it is composed of an embedding layer which combines embeddings for individual tokens and their positions in a

sentence, followed by a series of 12 BERT encoder layers, composed of multi-headed attention and feed-forward components. The end result is a pooled output vector representing the sentence embedding. The BERT model has been extensively pretrained on a next-sentence prediction task, making it a good starting point for further finetuning on our more specific downstream tasks.

Since we are primarily concerned with the possibility of constructive or destructive gradient interference between tasks in the BERT layers, the "tails" of our network corresponding to each task are relatively simple. For sentiment classification, a dropout layer followed by a dense linear layer projects the embedding to a five-dimensional vector (corresponding to the five sentiment classifications in the SST dataset, detailed in Section 5.1), whose entries are interpreted as logits and used for cross-entropy loss.

Paraphrase detection and semantic similarity analysis require incorporating information from two different sentence embeddings. We have developed two possible approaches to accomplish this task. In our "joint embedding" approach, we combine the two sentence embedding vectors $\vec{v}_1, \vec{v}_2 \in \mathbb{R}^h$ into a joint embedding vector $\vec{v}_J \in \mathbb{R}^{2h}$ as follows:

$$\vec{v}_J = [\vec{v}_1 + \vec{v}_2 \quad |\vec{v}_1 - \vec{v}_2|], \tag{1}$$

where the absolute value sign represents elementwise absolute value. This joint embedding vector is then passed through a dropout layer and a fully connected linear layer to produce a single logit. In the "multiplicative attention" approach, both vectors are passed through a dropout layer followed by an attention layer to produce a single logit ℓ:

$$\ell = \vec{v}_1(A + A^T + I)\vec{v}_2^T + b, \tag{2}$$

where $A \in \mathbb{R}^{h \times h}$ is a learnable attention matrix and b is a learnable scalar bias parameter. Notably, both of these approaches are manifestly symmetric under interchange of $\vec{v}_1$ and $\vec{v}_2$, reflecting the symmetric nature of the tasks we are attempting to learn. In the case of paraphrase detection, the logit is passed into a binary cross-entropy loss function. In the case of semantic similarity analysis, the loss function is $\mathcal{L}_{\text{sim}} = 1 - r$, where r is the Pearson correlation coefficient between the logits and true similarity scores over the batch. The overall network architecture is summarized in Fig. 1.

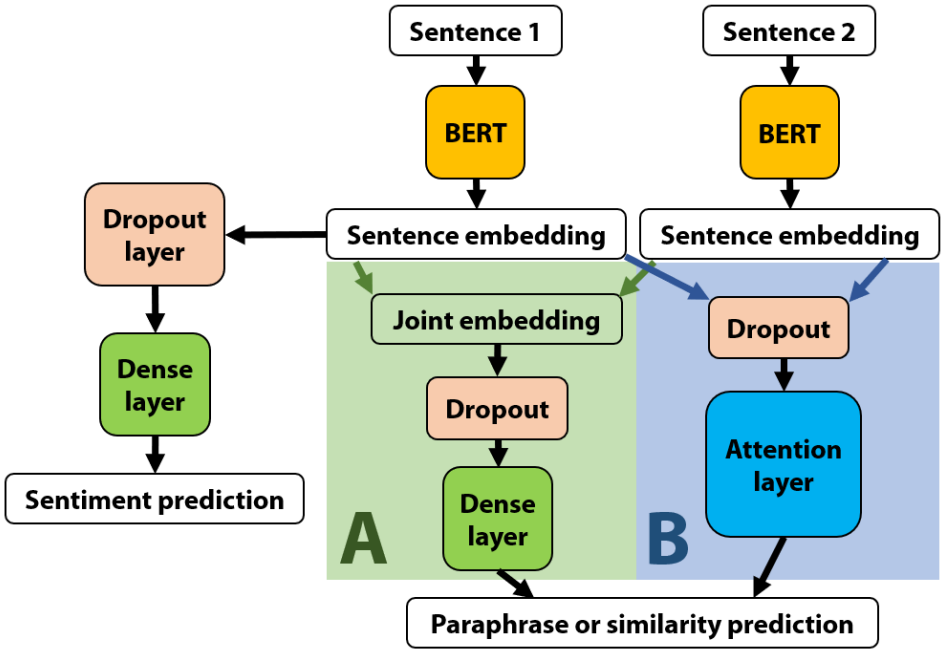

Figure 1: Overview of model architecture as described in Section 4. Section (A) corresponds to the joint embedding approach, while section (B) corresponds to the multiplicative attention approach for extracting information from two sentence embeddings at once.

The main component of this project is how to perform gradient updates based on the three loss functions corresponding to each task. For the unitary scalarization approach, we define our overall

loss function as

$$\mathcal{L} = \mathcal{L}_{\text{sent}} + \beta\mathcal{L}_{\text{para}} + \gamma\mathcal{L}_{\text{sim}}, \tag{3}$$

where β and γ are hyperparameters representing the relative weights for each task. For the gradient surgery approach, we implement PCGrad [5], both replicating the algorithm ourselves and using an online implementation in Pytorch for comparison [6]. In order to tease out the presence of destructive gradient interference, we will compare our multitask training to training on a single task. We will also investigate whether gradient surgery is more effective when both the paraphrase and similarity tasks use the same method of extracting a logit from two sentence embeddings (joint embedding or multiplicative attention). It is possible that destructive interference is more likely when network architecture produces similarities between the "tails" corresponding to two different tasks, which must result in the "head" embedding vectors encoding more information in the same way (e.g., the dot product of two vectors corresponding to semantic similarity and paraphrase probability).

As a baseline model for comparison, we have the minBERT implementation in the default project handout, which only learns the sentiment classification task. Unitary scalarization can be considered a baseline to compare against gradient surgery, as can runs trained only on single tasks (thus preventing any possibility of gradient interference). Since we are not modifying the sentiment classification tail, we do not necessarily expect improvement on that task over the baseline, but it is possible that learning the other tasks could improve performance on sentiment classification.

5 Experiments

5.1 Data

For sentiment analysis, we use the Stanford Sentiment Treebank (SST) as a dataset, which contains 11,855 sentences [7]. Phrases within these sentences are labeled into one of five categories, ranging from negative to positive. We use 8,544 of these examples for training and 1,101 examples for development testing, with 2,210 examples held out in the test set. The model is trained to minimize the cross-entropy between its predicted probability across classes and the true class.

For paraphrase detection, we utilize the Quora dataset [8], which contains 400,000 question pairs labeled based on whether or not the questions are paraphrases of each other. 141,506 examples will be used for training, 20,215 for development testing, and 40,431 for final testing. Similarly to the sentiment dataset, the model must correctly predict whether pairs of sentences are paraphrases or not, minimizing cross-entropy loss.

For semantic textual similarity, we use the SemEval STS Benchmark dataset [9], which contains 8,628 sentence pairs with similarities rated on a 0-5 scale. 6,041 examples will be used for training and 864 for development testing, with 1,726 examples held out. Since this is a continuous metric, the model is evaluated based on the correlation of its predicted logits with the true similarity (as measured by Pearson correlation).

5.2 Evaluation method

Accuracy is a reasonable initial metric to evaluate classification-based predictions (i.e., the sentiment and paraphrase prediction). However, precision and recall provide a more complete summary of the performance of the model on the binary classification task of paraphrase prediction – in particular, it helps account for the test and/or training sets having unequal numbers of entries for each class. Pearson correlation is our scale-invariant metric of choice to evaluate the sentiment data, which we also use as a loss function. No single metric can fully evaluate performance of a multi-task model, but we can extract meaningful qualitative insights from a variety of these simple quantitative metrics.

We are particularly interested in comparing the performance of our model across different conditions: with and without gradient surgery, and using different combinations of joint sentence embedding versus multiplicative attention to calculate paraphrase probability and similarity prediction. We can also validate our implementation of PCGrad by comparing its performance to the out-of-the-box implementation. Since PCGrad imposes a notable computational cost, it would also be worth comparing runtime and memory cost of gradient surgery versus unitary scalarization. On a similar note, the number of epochs it takes for the model to learn effectively is also relevant: perhaps gradient surgery takes more time per epoch but requires fewer epochs for effective learning.

	SST Batch Size	Quora Batch Size	STS Batch Size
No Gradient Surgery	16	40	16
Surgery (Our Implementation)	8	20	8
Surgery (Ref. [6] Implementation)	8	10	8

Table 1: Batch sizes used for experiments.

5.3 Experimental details

Experiments were run with a BERT hidden size $h = 768$, learning rate $\alpha = 10^{-5}$, paraphrase weight $\beta = \log(1/5)/\log(1/2)$, and similarity weight $\gamma = -\log(1/5)$ (chosen to roughly rescale loss functions by the size they would be for a randomly guessing model). To account for the large size of the Quora paraphrase dataset, larger batch sizes were used for this dataset, and data from the other datasets were re-sampled within a given epoch until the training had covered an entire epoch of the Quora dataset. Due to its larger memory cost, smaller batch sizes had to be used when implementing gradient surgery. Combined batch sizes for different conditions are summarized in Table 1. The number of training epochs ranged from 10 to 15, but all optimal models were obtained within 10 epochs. The AdamW optimizer (with or without gradient surgery) was used for all experiments.

5.4 Results

We first analyze the baseline results obtained via single-task training. The sentiment classification dev accuracy (obtained via the simple minBERT model) was 0.518. Fig. 2 shows dev results for the other two tasks. We observe that both multiplicative attention and joint embedding approaches result in learning relative to the 0.5 paraphrase accuracy and 0 similarity correlation that would be obtained from random guessing. The two methods seem equally effective in the context of similarity correlation, but the joint embedding approach is slightly better for paraphrase detection. As such, we use the joint embedding approach for the paraphrase tail of the model in future analyses.

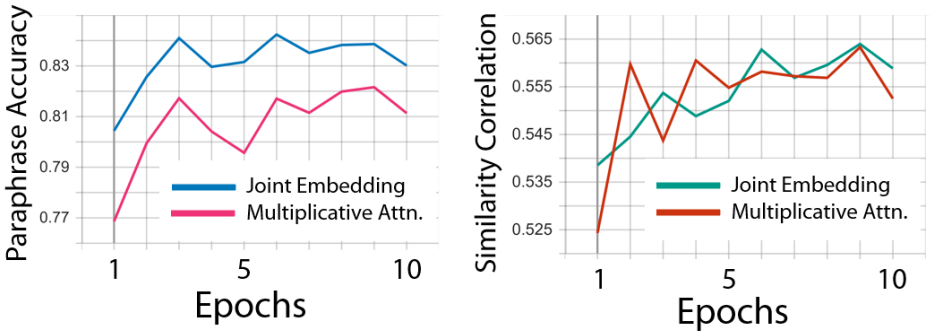

Figure 2: Comparison of dev performance for joint embedding and multiplicative attention approaches for single-task training on paraphrase detection and similarity analysis.

What happens when we introduce multi-task learning with unitary scalarization (Fig. 3)? Compared to single-task training (Fig. 2), not much changes: sentiment and paraphrase accuracies remain mostly unchanged, while similarity correlation slightly increases from 0.564 to 0.621 (see Table 2 for a full set of values). There is also little apparent difference in the accuracies obtained by using joint embedding on both the paraphrase and similarity tasks, or a mixed model using joint embedding on the paraphrase task and multiplicative attnetion on the similarity task. However, Fig. 3 shows one interesting difference: the joint-attention-only model requires more training than the mixed model to reach its peak dev similarity correlation, despite having similar correlations during training. This may reflect that it takes more training to find vector embeddings which "work" for both the paraphrase and sentiment tasks, when joint embeddings are being used for both.

Next, we investigate whether our results change when we implement gradient surgery. As shown in Table 2, our evaluation metrics do not change much: the joint-attention-only model yields almost identical results, while the mixed model appears to result in a slight increase in paraphrase accuracy

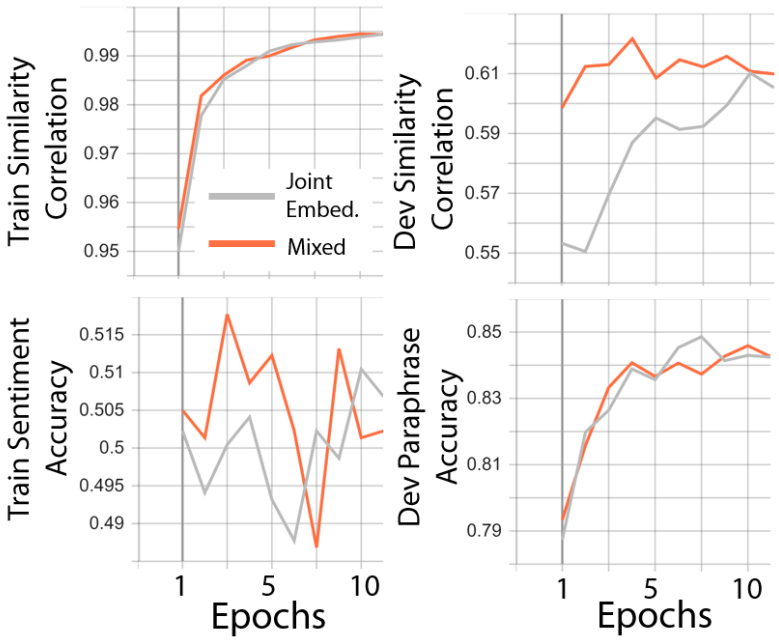

Figure 3: Comparison of dev and train performance for multitask learning with unitary scalarization, using models with only joint embedding and a "mixed" approach, where the paraphrase tail uses joint embedding and the similarity tail uses multiplicative attention.

and a slight decrease in sentiment accuracy (Fig. 4). However, the slower learning of similarity data for the joint-embedding-only model disappears, which may be a sign of gradient surgery eliminating destructive interference. While these results are for the version of PCGrad we implemented, the external implementation [6] gave similar results (Table 2). Finally, as shown in Table 3, not much changes when we consider the more fine-grained metrics of precision and recall for paraphrase evaluation: there is not a large asymmetry between the two metrics, nor do our different model choices have much impact. Perhaps our model is slightly better at recall (identifying the positive paraphrase pairs) than precision (avoiding misclassifying the non-paraphrase pairs).

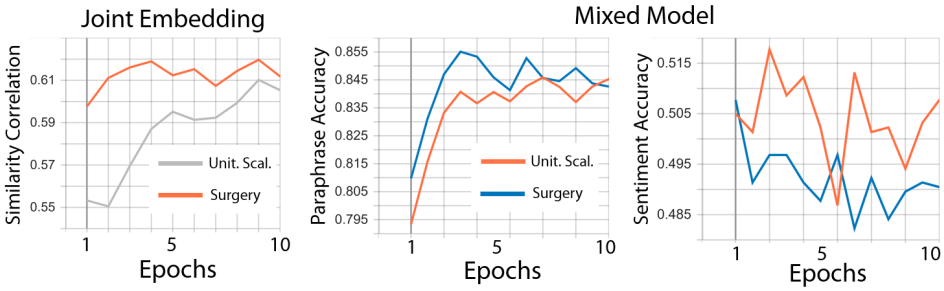

Figure 4: Comparison of dev performance for multitask learning between unitary scalarization and gradient surgery.

Overall, the effects of gradient surgery appear small at best, with minimal improvements to model performance. Furthermore, it comes with an increase in memory overhead and runtime, due to the need to calculate and store gradients for individual tasks. Indeed, we had to decrease batch size to save memory when implementing gradient surgery, and training took more than twice as long. Altogether, these results support the arguments of Ref. [3] that gradient surgery does not justify its increased computational costs over unitary scalarization.

	SST Dev	Quora Dev	STS Dev	SST Test	Quora Test	STS Test
Sentiment Only	0.518	-	-	-	-	-
Para. Only	-	0.842	-	-	-	-
Sim. Only	-	-	0.564	-	-	-
Unit. Sc. Joint	0.510	0.843	0.610	-	-	-
Unit. Sc. Mixed	0.513	0.843	0.616	0.517	0.846	0.599
Surgery Joint	0.512	0.844	0.614	-	-	-
Surgery Mixed	0.497	0.855	0.620	0.519	0.855	0.612
Surg. [6] Mixed	0.504	0.853	0.659	0.524	0.853	0.591

Table 2: Dev and test performances for the experiments described in the results section. SST and Quora scores are accuracies, while STS scores are Pearson correlation between logits and similarity. The first three rows are for single-task datasets using joint embedding for the paraphrase and similarity tasks. "Joint" refers to models using joint embeddings for both paraphrase and similarity tasks, while "Mixed" refers to models using joint embeddings for the paraphrase task and multiplicative attention for the similarity task. The last row represents data using the implementation of PCGrad by Ref. [6] for comparison.

	Precision	Recall
Unit. Sc. Joint	0.759	0.852
Unit. Sc. Mixed	0.763	0.843
Surgery Joint	0.760	0.855
Surgery Mixed	0.801	0.816
Surg. [6] Mixed	0.781	0.847

Table 3: Dev precision and recall for paraphrase pair identification on the Quora dataset. Models are as described in Table 2.

6 Analysis

Altogether, we observed that gradient surgery had very little impact in improving the performance of our model, and there are likely less computationally intensive methods that could achieve similar or greater performance increases. Why is this the case? Our single-task runs, which can be interpreted as ablation studies removing any possibility of gradient interference, had similar performance to our multitask model. This suggests that conflicting gradients may not have posed a barrier to learning to begin with, meaning that the realized perfomance gains of gradient surgery are small. Perhaps gradient surgery would be more useful for a model trained on more tasks, such as the 40 tasks in the CelebA dataset analyzed in Ref. [3]. However, increasing the number of tasks also increases the computational cost of gradient surgery relative to unitary scalarization, making it all the more crucial to justify through performance gains.

We did observe some limited effects of gradient surgery which could be attributed to remedying gradient interference. In the left panel of Fig. 4, the joint-embedding only model with unitary scalarization requires many more epochs of training to reach accuracy similar to the one with gradient surgery. This is possibly because in the joint-embedding only model, both paraphrase detection and similarity must be encoded in the sum and difference of vectors – the only difference between the tails corresponding to each task are the weights in the linear layer. The dificulty of finetuning embeddings to work with both tasks at once may lead to slower training in a way that can be alleviated by gradient surgery. However, this is just one example, and could be caused by randomness in initializing the model; furthermore, the ultimate accuracies attained by each model are comparable.

Ref. [3] argues that any observed benefit of gradient surgery can be attributed to its regularizing effect. In this work, we only used dropout as an explicit regularization method across all models. Potentially, this aspect of gradient surgery resulted in the slightly improved performance for paraphrase detection and sentiment analysis observed in the "Surgery Mixed" runs; however, we also observe a comparable drop in performance on the sentiment dataset (Table 3). Nonetheless, we argue that all of these effects are small enough that gradient surgery does not justify its use for this set of tasks.

7 Conclusion

In this work, we have performed an analysis of the costs and benefits of gradient surgery in the context of a natural language processing multitask learning problem. We have found that in the context of these tests, gradient surgery fails to produce a noticeable enough benefit to compensate for its additional computational cost, supporting the conclusion of Ref. [3] that the simpler "unitary scalarization" method is all that is necessary for multitask learning. Nevertheless, we have identified potential signatures of the gradient interference which gradient surgery aims to solve, suggesting that methods to resolve this problem are still important – whether they be changes to model architecture or regularization.

Our work is extremely preliminary, and several extensions should be performed before relying on its conclusions. A larger sample size of experiments for each set of conditions is crucial, so that we have a sense of what variations between runs are due to random instantiations as opposed to the effects of gradient surgery or other model choices. In addition, we did not have time to perform a meaningful hyperparamter scan, varying the learning rate α or loss weights β and γ to find an optimal set of conditions. It is possible that one of the models tested would perform notably better at a different set of hyperparameter values. Furthermore, changing regularization methods (e.g., removing dropout or adding additional weight decay) could help identify whether gradient surgery provides a regularization effect that is substitutable for more standard regularizers. Finally, performing a similar study for a broader range of tasks would elucidate whether gradient surgery is more effective when the same model must simultaneously learn more than three tasks at once, despite its larger computational cost.

All in all, the promise of multitask learning algorithms demands a closer investigation into effective methodology for simultaneous learning on multiple tasks. It seems likely that techniques like the gradient surgery described in Ref. [5] are useful in certain contexts. Yet even if the conclusion of Ref. [3] that such methods are unnecessary do not generalize to all fields, their work still emphasizes that comparative studies of learning methodologies, rather than a variety of individual *ad hoc* methods, are crucial to build better models.

References

[1] Shijie Chen, Yu Zhang, and Qiang Yang. Multi-task learning in natural language processing: An overview. *CoRR*, abs/2109.09138, 2021.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and M. Pawan Kumar. In defense of the unitary scalarization for deep multi-task learning. *CoRR*, abs/2201.04122, 2022.

[4] Uri Shaham, Maha Elbayad, Vedanuj Goswami, Omer Levy, and Shruti Bhosale. Causes and cures for interference in multilingual translation, 2022.

[5] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *CoRR*, abs/2001.06782, 2020.

[6] Wei-Cheng Tseng. Weichengtseng/pytorch-pcgrad, 2020.

[7] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing*, 2013.

[8] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. First quora dataset release: Question pairs.

[9] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *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, June 2013. Association for Computational Linguistics.