Rhapsody on a Theme of Gradient Surgery: Variations to Improve minBERT for Multi-Task Learning

Stanford CS224N Default Project

Christian Femrite
Department of Electrical Engineering
Stanford University
cfemrite@stanford.edu
Mentor: Timothy Dai

Abstract

While large language models and pretrained transformers have demonstrated significant utility in performing individual human language tasks, sharing structure within a model to perform multiple tasks with the same efficacy has proven to be a challenge. It is difficult to gain model size and compute efficiency across multiple tasks without sacrificing performance on individual tasks. A notion of conflicting gradients has been proposed as a framework for understanding why training a single model to perform many tasks can lead to suboptimal individual task performance, but the best method of de-conflicting these gradients remains an open question. Some notable proposed solutions implement "gradient surgery" (e.g. PCGrad Yu et al. (2020) and GradVac Wang et al. (2020b)), which involve projecting conflicting gradients from different training tasks onto one another. I implemented a miniaturized version of the BERT model Devlin et al. (2018) with three distinct task objectives: (1) sentiment analysis, (2) paraphrase detection, and (3) semantic textual similarity assessment, and then I conducted a series of experiments that adapt these "gradient surgery" algorithms from existing literature to incorporate new methods including (1) task scheduling considerations, (2) non-random gradient projection algorithm variants, (3) conflict-controlled learning rates, and (4) manually introduced gradient sparsity intended to reduce the computational workload of gradient surgery algorithms, which I term "gradient suppression". I found that simpler, stochastic optimization techniques tend to result in superior multi-task performance, and that the aforementioned gradient surgery algorithms are generally robust to gradient suppression.

1 Introduction

This work investigates a variety of training techniques to improve a single model's performance across multiple tasks using shared model parameters through a case study leveraging a miniature version of the BERT architecture Devlin et al. (2018) to perform (1) sentiment analysis, (2) paraphrase detection, and (3) semantic textual similarity assessment. Initially, I explore different data presentation techniques, which have been explored extensively in the literature (e.g. "domain generalization" Mansilla et al. (2021), "annealed sampling" Stickland and Murray (2019), but mainly, I focus on optimization procedures that occur during training, mostly related to intuitive manipulation of gradients and designed to break outside of the intentions of basic stochastic gradient descent. Specifically, these optimization procedures will seek to address the sometimes unavoidable occurrence of conflicting gradients when training the shared portion of the model on different tasks.

In particular, we are interested in determining training methods that can produce a multi-task model with comparable task-for-task performance with respect to single-task-trained models, since in practice, it is often easier to achieve higher performance with respect to each task by training separate models for each task. There are also several practical motivations for a study of this kind, including:

- **Edge Computing**: In situations where NLP models are deployed on lightweight or remotely located devices with lower computational capacity, it is highly desirable to be able to share those scarce computational resources across tasks. This is especially important for power and latency budgets, as well as in cases where multi-functional applications are needed.

- **Training Compute Arbitrage**: One natural question that arises is, if we were to train multiple models with the same architecture, each on a different task, would we need to use the same quantity of training data to train a multi-task model across each of those tasks? By simplifying the computational mechanisms required to implement training optimization procedures, it would be commendable to achieve an asymmetric benefit in reducing training durations and complexity.
- **Storage Consolidation**: As Kaplan et al. (2020) expand upon in their paper on Scaling Laws for Natural Language Models, there is an exponential relationship between model size and training loss. With model parameter counts now entering the "trillions" order of magnitude, requiring different models for different tasks comes with a remarkable storage penalty.

Considering these impetuses, we also recognize that the naive approach of training a multi-task model according to individual task objectives without regard to other tasks in a monolithic manner has proven insufficient because doing so neglects *negative interference* between tasks Wang et al. (2020a). Therefore, we seek to explore multi-task model optimization via intuitive exploration related to loss geometries. Specifically, we want to know:

1. How effective are existing techniques at mitigating *negative interference* between model gradients?
2. What impact does dataset presentation during training have on multi-task performance?
3. What natural mathematical relationships between task gradients could be exploited to improve the optimization process?
4. How robust are multi-task model gradients to suppression with respect to training objectives?

With these questions in mind, I perform a series of experiments that first seek to discover the best method of sampling tasks during training, then apply that towards experiments investigating variations on existing *gradient surgery* techniques Yu et al. (2020) Wang et al. (2020b). These experiments consist of (1) alternative methods to task scheduling during training, (2) alternative geometric compromises between tasks' computed gradients, (3) controlling the learning rate according to the cosine similarity of gradients, and (4) introducing gradient suppression. To the best of my knowledge, this is the first study that examines the specific modifications that I am introducing to these existing geometric approaches eliminating negative interference between gradients, as well as the first study that explores sparsity in computing gradient normalization with these gradient surgery algorithms (i.e. "gradient suppression").

2 Related Work

There exists a progression of work related to the reconciliation of training tasks that experience *negative interference* that seek to do so through various mechanisms of de-conflicting gradients, and these works have succeeded in improving performance for multi-task models Chen et al. (2017) Sener and Koltun (2018) Yu et al. (2020). Most prominently Yu et al. (2020) introduced a technique they have termed *projecting conflicting gradients*, abbreviated "PCGrad", which provides an algorithm intended to address three core problems they identify, namely (1) conflicting gradients, (2) high gradient curvature, and (3) large gradient differences. I leave the details of their rationale to their original work, but the core idea is that in cases where gradients have a negative cosine similarity, the optimizer instead uses the gradient's projection onto the normal plane of another task's gradient:

$$\mathbf{g_i^{PC}} \leftarrow \mathbf{g_i^{PC}} - \frac{\mathbf{g_i^{PC}} \cdot \mathbf{g_j}}{||g_j||^2}\mathbf{g_j} \tag{1}$$

Though this algorithm provides an elegant solution to the problems outlined, others, like myself, have considered ways to improve PCGrad. One notable contribution was a modification of PCGrad Wang et al. (2020b) termed "Gradient Vaccine" which instead implements an adaptive gradient similarity objective, addressing the concern that PCGrad does not allow for complex inter-task relationships since PCGrad restricts the compromise between conflicting gradients to the normal plane of one of the gradients. Instead, they incorporate the cosine similarity between tasks: $cos(\theta) = \phi_{ij}$, and track a similarity objective between two gradients (ϕ_{ij}^T) relative to this value, which is in turn tuned by another hyperparameter, β as follows (where i, j are task indicies, and k is a parameter group):

$$\mathbf{g_i^{GV}} \leftarrow \mathbf{g_i^{GV}} + \frac{||\mathbf{g_i}||(\phi_{ij}^T\sqrt{1-\phi_{ij}^2} - \phi_{ij}\sqrt{1-(\phi_{ij}^T)^2})}{||\mathbf{g_j}||(\sqrt{1-(\phi_{ij}^T)^2})} \tag{2}$$

$$\hat{\phi}_{ijk}^{(t)} = (1-\beta)\hat{\phi}_{ijk}^{(t-1)} + \beta\phi_{ijk}^{(t)} \tag{3}$$

3 Approach

Some approaches to multi-task learning rely on a general-purpose model that shares all parameters across all tasks, such as in the case of the NLP Decathlon McCann et al. (2018), but this research instead uses an architecture that shares most parameters between models and uses task-specific output layers each with a smaller number of parameters.

3.1 minBERT Implementation + PCGrad & GradVac Optimizers

The Default Final Project Handout served as a guideline for implementing a miniature version of the BERT architecture Devlin et al. (2018) that is in turn leverages the original implementation of the "attention" mechanism introduced in Vaswani et al. (2017). Doing so involved implementing a multi-head attention structure, a transformer layer, and a training optimizer that leverages the AdamW algorithm. With the standard minBERT model implemented, additional sets of cascaded linear layers were connected to perform inference for each of these three tasks:

1. **Sentiment Analysis (SST)**: Given a tokenized sentence, determine whether the expressed opinion is positive, negative, or neutral.
2. **Paraphrase Detection (PARA)**: Determine whether a given pair of text passages convey the same semantic meaning, i.e. the reuse of text giving the same meaning in another form.
3. **Semantic Textual Similarity (STS)**: Determine the degree of semantic equivalence of two texts (different from PARA since it is a subjective rating 0-5 instead of just yes/no).

After implementing the minBERT model and task-specific output layers, I implemented the PCGrad Update Rule and Gradient Vaccine Update Rule (see Appendix) in conjunction with the AdamW optimizer to test the impact of these algorithms on the model's performance for downstream tasks.

3.2 Sampling Tasks for Multi-Task Training

The simplest approach to sampling tasks for training would be to cycle through each task's dataset in a fixed order, which is referred to as "round robin" sampling. An equally simple alternative method could be sampling from each task's dataset in a random order. However, we have far more samples in the Quora dataset compared to the other datasets, so by the time we train our minBERT model on all of the data from the other datasets, we will only have trained it on a small portion of the data from the Quora dataset. Therefore, I considered the selection and ordering of training data to establish an ideal methodology for presenting training data to the minBERT model before continuing with further experiments. In an effort to better understand the implications of this disparity in dataset quantities, I employed the *annealed sampling* approach described by Strickland & Murray in their work on *projected attention layers* Stickland and Murray (2019), which suggests sampling from the dataset N_i associated with task i with probability p_i based on a parameter α as follows (e is the current epoch and E is the total number of training epochs):

$$p_i \propto N_i^\alpha \tag{4}$$

$$\alpha = 1 - 0.8\frac{e-1}{E-1} \tag{5}$$

3.3 Variations on Random Task Projection PCGrad for Multi-Task Optimization

In an effort to expand upon the work done by Yu et al. (2020) in introducing PCGrad, I devised a number of simple gradient de-conflicting methods that intentionally choose the tasks that are used when projecting gradients (as opposed to the random selection used in the original paper), as this is a detail of the original PCGrad Update Rule that is not sufficiently justified or explored. This was also further motivated by observing that the complexity introduced by the Gradient Vaccine Update

Rule may not achieve sufficient performance benefits to justify its substantially more computationally expensive algorithm. Therefore, I have conducted a series of experiments to explore the impact of modifying the task selection involved in the PCGrad algorithm on design accuracy during training. Assume there are N tasks and the index i corresponds to the current training objective task:

- **Averaged PCGrad**: Modify the standard PCGrad Update Rule to instead perform the PCGrad Update Rule between the current task and each of the previously computed gradients for all other tasks, and take the average of those results as the projected gradient for the current task.

$$\mathbf{g_i^{PC}} \leftarrow \frac{1}{N-1} \sum_{j \propto (\text{tasks} \neq i)} \left(\mathbf{g_i^{PC}} - \frac{\mathbf{g_i^{PC}} \cdot \mathbf{g_j}}{||g_j||^2} \mathbf{g_j} \right) \tag{6}$$

- **Most Similar PCGrad**: Instead of choosing a random task to project the current task's gradient, $\mathbf{g}_i^{PC}$, choose the gradient projection onto the gradient from the other task that has the **closest** cosine similarity.

$$\mathbf{g_i^{PC}} \leftarrow \mathbf{g_i^{PC}} - \max_{j \propto (\text{tasks} \neq i)} \left(\frac{\mathbf{g_i^{PC}} \cdot \mathbf{g_j}}{||g_j||^2} \mathbf{g_j} \right) \tag{7}$$

- **Most Different PCGrad**: Same procedure as described in *"Most Similar PCGrad"*, but instead choose the gradient projection onto the gradient from the other task that has the **farthest** cosine similarity.

$$\mathbf{g_i^{PC}} \leftarrow \mathbf{g_i^{PC}} - \min_{j \propto (\text{tasks} \neq i)} \left(\frac{\mathbf{g_i^{PC}} \cdot \mathbf{g_j}}{||g_j||^2} \mathbf{g_j} \right) \tag{8}$$

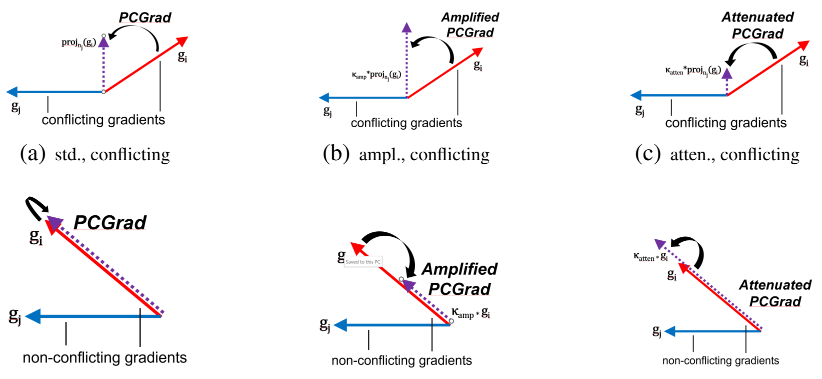
3.4 Learning Rate Amplification/Attenuation via Cross-Task Cosine Similarity

Another natural thought, is even if we choose the tasks to project randomly, do we want to be updating our model parameters more or less in cases where there are conflicting gradients? It may be beneficial to take greater steps or lesser steps in directions where *projecting conflicting gradients* should occur according to the algorithm. This leads me to a modification of the parameter update step of the Adam algorithm using the cosine similarity ϕ_{ij} between the two gradients by introducing a derived parameter κ in conjunction with the PCGrad Update Rule as follows:

$$\phi_{ij} \propto [-1, 1] \rightarrow \begin{cases} 1 + \phi_{ij} = \kappa \propto [0, 2] \text{ (attenuate conflicting gradient steps)} \\ 1 - \phi_{ij} = \kappa \propto [2, 0] \text{ (amplify conflicting gradient steps)} \end{cases} \tag{9}$$

Then the parameter update step of the Adam algorithm becomes:

$$\theta_t \leftarrow \theta_{t-1} - \kappa \cdot \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t + \epsilon}) \tag{10}$$



(a) std., conflicting (b) ampl., conflicting (c) atten., conflicting

(d) std., nonconflicting (e) ampl., nonconflicting (f) atten., nonconflicting

Figure 1: PCGrad amplification and attenuation technique visualizations.

3.5 Gradient Suppression Dropout for Multi-Task Robustness

Conceptually, the widely-adopted technique of "dropout" in training neural networks is used with the intention of providing inference resiliency by allowing the neural network to experience learning through multiple nodal pathways during training to produce the same prediction, thereby allowing the final implementation to superimpose these multiple pathways for a more robust inference. This is done by eliminating some nodes randomly during different batches of training, and scaling the other nodes' weights proportionally. Now, apply the same concept in the context of *projecting conflicting gradients*. Each gradient vector is a high-dimensional object, with many influences affecting the direction in which it points. Therefore, I thought it would be interesting to study what would happen if some of the gradient vector components were suppressed, hopefully achieving an effect similar to the dropout technique, i.e. to determine whether the nuanced inter-dimensional dissimilarities of gradients across tasks might be better reconciled in this manner. $\mathbf{b}$ is sampled as a Bernoulli distribution with some probability p (a distinct, newly introduced hyperparameter) for each batch, just as this procedure is done for hidden dropout layers. For this experiment, hidden layer dropout probability was also varied to determine if the combination of these two techniques could have additive effects, especially for sparse dropout and suppression probability combinations.

$$\mathbf{g_i^{PC}} \leftarrow \mathbf{b} \cdot \left(\mathbf{g_i^{PC}} - \frac{\mathbf{g_i^{PC}} \cdot \mathbf{g_j}}{||g_j||^2} \mathbf{g_j} \right), \text{ where } \mathbf{b} \in \mathbb{R}^N \text{ and } b_i \in \{0,1\} \tag{11}$$

4 Experiments

4.1 Data

- **Task #1 - Sentiment Analysis with Stanford Sentiment Treebank Dataset (SST)**: This dataset consists of single sentences from movie reviews labeled with a positivity score $\in$ {negative, somewhat negative, neutral, somewhat positive, positive} $\rightarrow$ {1, 2, 3, 4, 5}.
  Sample Breakdown: `train`=8544, `dev`=1101, `test`=2210
- **Task #2 - Paraphrase Detection with Quora Dataset (PARA)**: This dataset consists of question pairs from *quora.com* with labels indicating whether the pair is a paraphrase of one another or not.
  Sample Breakdown: `train`=283,010, `dev`=40,429, `test`=80,859
- **Task #3 - Semantic Textual Similarity with SemEval STS Benchmark Dataset (STS)**: This dataset consists of sentence pairs of varying similarity manually labelled according to the subjective equivalency of their meaning on a scale of 0 (unrelated) to 5 (equivalent meaning).
  Sample Breakdown: `train`=6040, `dev`=863, `test`=1725

4.2 Evaluation Method

Each model's loss is calculated using a dataset-appropriate loss function, and `dev` accuracy is used to ascertain the effectiveness of the trained model's inference capabilities for each task.

- **Task #1 - Sentiment Analysis (SST)**: *categorical cross-entropy*, since the sentiment of each movie review falls into a category corresponding to a positivity score.
- **Task #2 - Paraphrase Detection (PARA)**: *binary cross-entropy*, since each pair of sentences either is or is not a paraphrase pair, so the only two inference possibilities are correct and incorrect.
- **Task #3 - Semantic Textual Similarity (STS)**: *mean squared error*, since this dataset is labeled according to the degree of similarity between sentence pairs.

4.3 Experimental Details

After the initial determination that the "round robin" approach to sampling tasks for training would be optimal, this was the method used thereafter. Each experiment consisted of 10 epochs, each of which consisted of 500 batches of size 8 each. Training was conducted with `num_linear_layers`=2 for each task's output head, and a learning rate of `lr`=1e-5. In the case of GradVac, I used the recommended value of $\beta = 10^{-2}$. To submit to the DEV and TEST leaderboards, I took the best-performing model (PCGrad) and trained for 50 epochs instead of just 10 epochs.

4.4 Results

4.4.1 Sampling Tasks for Multi-Task Learning

METHOD	TASK	AdamW	PCGrad	GradVac
ROUND ROBIN	SST	0.483	0.511	0.472
	PARA	0.838	0.841	0.816
	STS	0.513	0.875	0.867
	AVERAGE	**0.611**	**0.742**	**0.718**
RANDOM	SST	0.422	0.463	0.467
	PARA	0.804	0.822	0.823
	STS	0.575	0.881	0.881
	AVERAGE	**0.600**	**0.722**	**0.724**
ANNEALED SAMPLING	SST	0.390	0.422	0.408
	PARA	0.841	0.835	0.836
	STS	0.834	0.821	0.818
	AVERAGE	**0.688**	**0.693**	**0.687**

Table 1: minBERT performance by task sampling method. Round robin generally performs better due to relative preference for SST task. Only AdamW optimizer benefits from annealed sampling.

4.4.2 Variations on PCGrad

TRAINING STRATEGY	SST	PARA	STS	AVERAGE / OVERALL
Single-Task Baseline	0.517	0.963	0.890	**0.790**
Multi-Task Baseline	0.483	0.838	0.513	**0.611**
PCGrad	0.511	0.841	0.875	**0.742**
PCGrad - 50 epochs (DEV)	0.491	0.856	0.875	**0.761** (autograder)
PCGrad - 50 epochs (TEST)	0.518	0.857	0.878	**0.771** (autograder)
"Averaged" PCGrad	0.477	0.839	0.876	**0.731**
"Most Similar" PCGrad	0.493	0.842	0.876	**0.737**
"Most Different" PCGrad	0.488	0.833	0.872	**0.731**
"Amplified" PCGrad Learning	0.253	0.632	0.014	**0.300**
"Attenuated" PCGrad Learning	0.326	0.799	0.858	**0.661**

Table 2: Accuracies for gradient surgery methods training minBERT. The standard PCGrad algorithm remains the most effective multi-task method, though falls short of single-task baseline models.

4.4.3 Gradient Suppression Dropout

		GRADIENT SUPPRESSION PROB.		
HIDDEN DROPOUT PROB.	TASK	$\mathbb{P} = 0.30$	$\mathbb{P} = 0.45$	$\mathbb{P} = 0.60$
$\mathbb{P} = 0.15$	SST	0.463	0.468	0.455
	PARA	0.809	0.803	0.803
	STS	0.866	0.862	0.861
	AVERAGE	**0.713**	**0.711**	**0.706**
$\mathbb{P} = 0.30$	SST	0.458	0.463	0.467
	PARA	0.808	0.808	0.810
	STS	0.859	0.862	0.861
	AVERAGE	**0.708**	**0.711**	**0.713**
$\mathbb{P} = 0.45$	SST	0.446	0.425	0.444
	PARA	0.804	0.805	0.798
	STS	0.862	0.857	0.861
	AVERAGE	**0.704**	**0.696**	**0.701**

Table 3: minBERT performance for different hidden dropout and gradient suppression probabilities. Higher hidden dropout appears to have a detrimental effect, but higher gradient suppression does not.

5 Analysis

The first group of experiments endeavoring to determine an optimal task sampling order shows that simpler methods can be more effective when training techniques do not stack well together. Overall, the AdamW optimizer experienced a benefit from the *annealed sampling* technique, but *annealed sampling* was actually an encumbrance for performance with the PCGrad and GradVac optimizers. This may initially seem counterintuitive, but one possible explanation could be that an unbalanced task selection when using the PCGrad or GradVac optimizers could cause a disproportionate number of training batches to perform projections from one particular task's gradient, leading the network to favor that task's training objectives over the others even more disproportionately than the AdamW optimizer would. The important distinction is that my experiment with *annealed sampling* relates to the selection of the i-th task that is used for the projection, whereas the PCGrad and GradVac algorithms prescribe a random selection of the j-th task being projected onto.

In experimenting with the "averaged", "most similar", and "most different" variations on PCGrad, it did not appear that any of them had a substantial benefit, which suggests that the randomness of tasks selected for projection in the original PCGrad algorithm is not as critical of an issue as I originally suspected. There may be a unique dynamic of this experiment since there are exactly three tasks, as during each batch, the PCGrad algorithm is applied to a *pair* of tasks, and the focus on the gradient relationship between any *pair* of tasks comes at the expense of the relationship of those tasks with the third task. Perhaps this could be thought of as somewhat analogous to the famous *three body problem* in physics in terms of the ordered "orbit" of these three tasks around a common objective based on the "force" they apply to each others' gradients. In general, for each of these first two experiments, there is a theme of the near-interchangeability of randomness (considering the "random" approach in task sampling performed almost as well as "round robin") and balance (comparing the "round robin" and "averaged PCGrad" approaches) in training techniques.

Now considering the experiment exercising influence on the learning rate with respect to the cosine similarity while using PCGrad, neither amplification or attenuation seems to have a positive influence on the cross-task training objective. There are a couple of conjectural reasons for why this may be. Firstly, amplification of conflicting gradients results in a noticeable decrease in performance, possibly because the true nature of the PCGrad Update Rule is the provision of an adversarial compromise between the two gradients, so stepping in a normalized direction more than absolutely necessary may poison or distort the step the optimizer is trying to take to improve the model. Secondly, attenuation of conflicting gradients seems to only be slowing training progress as opposed to harming it, as the optimizer is still stepping in the direction of the normalized gradient, but does not move as drastically. Perhaps at the expense of longer training times, a version of the PCGrad algorithm that implements attenuated learning might achieve a better converged result, but that is left to future work.

Finally, the study on gradient suppression generally showed that even considerable masking of gradient dimensions used in calculating gradient updates did not appreciably affect final multi-task model performance in a detrimental manner. This indicates that it should be possible to save some of the additional computational overhead required by using sparse gradient vectors to compute projections whenever using the PCGrad algorithm to implement training optimization.

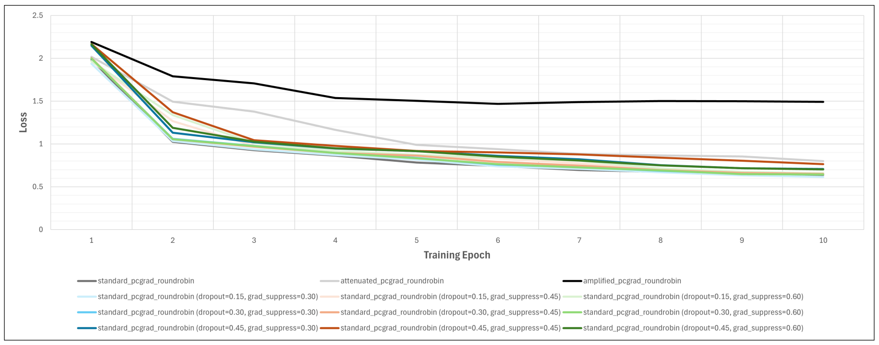


Figure 2: Training loss for PCGrad variations. Amplification and attenuation of PCGrad show the worst loss curves, while some of the higher hidden layer dropout curves lag behind the other curves.

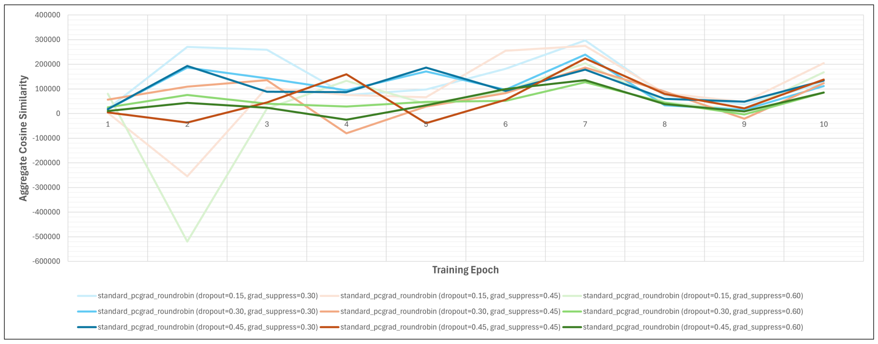


Figure 3: Aggregate cosine similarity by epoch for PCGrad variations. Interestingly, tests across all hidden dropout probabilities and gradient suppression probabilities converge towards a similar cosine similarity pattern at later epochs, perhaps due to model parameters being directed towards consistent gradient topology where multi-task performance is optimized.

6 Conclusion

This paper surveyed a wide range of variations on gradient surgery techniques including task sampling, intuitive mathematical variations on *projecting conflicting gradients* related to both inter-task relationships and learning rate, and gradient suppression via sparsity. In general, there appear to be diminishing benefits in increased complexity of gradient projection techniques. Though the original paper that introduced the *Gradient Vaccine* technique shows notable improvement using a slightly more computationally laborious algorithm (with respect to PCGrad) for the application of neural machine translation, this result may not generalize to other tasks, such as the SST, PARA, and STS tasks used in this study. With respect to all but the last experiment on gradient suppression, each involved studying an added element of complexity within the optimization procedure and subsequently demonstrated decreased or unimproved cross-task performance. By contrast, the gradient suppression experiment that involved a simplification mechanism demonstrated approximately equivalent cross-task performance, showing that there may be computational arbitrage opportunities to achieve equivalent multi-task model performance with fewer training resources and model requirements.

Further work may seek to investigate the effects of these techniques over longer training durations that would allow for the convergence of stochastic gradient descent. In particular, it would be interesting to know if "attenuation" of PCGrad would result in overall better metrics at the end of training over a longer time horizon, as one might anticipate a tradeoff between training time and final model quality that might result from smaller optimizer steps. Another area for future research could be developing a deeper understanding of the stability of the cosine similarity of inter-task gradients, especially across situations involving more than three task objectives. It is somewhat perplexing that for most of these training methods, the overall preponderance of non-conflicting gradients did not seem to emerge during later epochs as we would hope. This suggests an unanswered question about the nature of *gradient surgery*, namely, does it have the potential to not just dictate a reluctant compromise between tasks' gradients, but to actually reconcile them in a manner that discovers model parameters symbiotically beneficial to all training objectives?

7 Ethics Statement

In the effort of consolidating models into a single model that can perform multiple tasks, there are several ethical considerations and risks. In many applications involving high-risk environments, for example, autonomous driving, "co-bots" for manufacturing, and surgical robotics, it would not be appropriate to sacrifice any amount of accuracy in an effort to create a multi-task model due to the inherently severe safety consequences and risk to personal life that would result from failure at any one of the tasks that the multi-task model may have been trained to perform. This consideration means that we must bear the increased compute cost of using single-task-trained models if they perform better for each of these tasks (i.e. using whichever model achieves the highest level of accuracy and therefore the highest safety, whether that is a single-task-trained model or multi-task-trained model). Alternatively, if we seek to employ multi-task models for these kinds of risk-laden activities, we could add human-controlled safeguards such as haptic feedback (e.g. as an autonomous vehicle makes a lane change), advance approval requirements (e.g. mapping of a surgical incision before executing), and automatic stopping capabilities. This is not to say that a single model would be completing such a diverse range of tasks from those mentioned, but that with the proper regulation and human/AI teaming and intervention capabilities, it may be acceptable to use a multi-task-trained model within one of these domains if the human's intervention can achieve the same level of confidence that a single-task-trained model would require to be allowed to operate independently of human assistance.

An additional ethical consideration is the amount of electrical energy consumed to train these models. Where there are substantial differences in energy required to train multi-task models and energy required to train single-task models, it should be considered if the benefit of obtaining a capably trained model is worth additional energy expenditure, as well as if the anticipated performance of using a single multi-task-trained model would be sufficient to save repeated fine-tuning efforts for each task objective trained using a single-task trained model. In general, it should be carefully considered how the electrical energy used to perform training computations is obtained, which would ideally come from renewable sources such as solar, wind, geothermal, etc. Another consideration is at what times training takes place, which should seek to minimize burden on power grids during peak hours, perhaps by running overnight instead of during the daytime. It should also be considered how that additional power draw affects the energy economics of regular consumers using the power grid for other purposes, and if there should be regulations on the total portion of produced energy that should be dedicated to machine learning instead of to other purposes. In general, considering model characteristics such as shrinking models, batching, using lower-precision data types, early training stopping, using compression, and using sparsity where possible is also beneficial. As this consideration relates to this study, it should be possible to use higher levels of gradient multiplication sparsity in conjunction with the PCGrad algorithm without experiencing a substantial performance decrease.

References

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2017. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *CoRR*, abs/1711.02257.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Lucas Mansilla, Rodrigo Echeveste, Diego H. Milone, and Enzo Ferrante. 2021. Domain generalization via gradient surgery. *CoRR*, abs/2108.01621.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *CoRR*, abs/1810.04650.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. 2020a. On negative interference in multilingual models: Findings and A meta-learning treatment. *CoRR*, abs/2010.03017.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. 2020b. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *CoRR*, abs/2010.05874.

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*, volume 33. Curran Associates, Inc.

A Appendix: PCGrad and GradVac Algorithms

```
Algorithm 1 PCGrad Update Rule
Require: Model parameters $\theta$, task minibatch $\mathcal{B} = \{\mathcal{T}_k\}$
1: $\mathbf{g}_k \leftarrow \nabla_\theta \mathcal{L}_k(\theta)\ \ \forall k$
2: $\mathbf{g}_k^{\text{PC}} \leftarrow \mathbf{g}_k\ \ \forall k$
3: for $\mathcal{T}_i \in \mathcal{B}$ do
4:     for $\mathcal{T}_j \overset{\text{uniformly}}{\sim} \mathcal{B} \setminus \mathcal{T}_i$ in random order do
5:         if $\mathbf{g}_i^{\text{PC}} \cdot \mathbf{g}_j < 0$ then
6:             // Subtract the projection of $\mathbf{g}_i^{\text{PC}}$ onto $\mathbf{g}_j$
7:             Set $\mathbf{g}_i^{\text{PC}} = \mathbf{g}_i^{\text{PC}} - \frac{\mathbf{g}_i^{\text{PC}} \cdot \mathbf{g}_j}{\|\mathbf{g}_j\|^2} \mathbf{g}_j$
8: return update $\Delta\theta = \mathbf{g}^{\text{PC}} = \sum_i \mathbf{g}_i^{\text{PC}}$
```

Figure 4: Algorithm for PCGrad Update Rule.

```
Algorithm 1 GradVac Update Rule
1: Require: EMA decay $\beta$, Model Components $\mathcal{M} = \{\boldsymbol{\theta}_k\}$, Tasks for GradVac $\mathcal{G} = \{\mathcal{T}_i\}$
2: Initialize model parameters
3: Initialize EMA variables $\hat{\phi}_{ijk}^{(0)} = 0, \forall i, j, k$
4: Initialize time step $t = 0$
5: while not converged do
6:     Sample minibatch of tasks $\mathcal{B} = \{\mathcal{T}_i\}$
7:     for $\boldsymbol{\theta}_k \in \mathcal{M}$ do
8:         Compute gradients $\mathbf{g}_{ik} \leftarrow \nabla_{\boldsymbol{\theta}_k} \mathcal{L}_{\mathcal{T}_i}, \forall \mathcal{T}_i \in \mathcal{B}$
9:         Set $\mathbf{g}'_{ik} \leftarrow \mathbf{g}_{ik}$
10:        for $\mathcal{T}_i \in \mathcal{G} \cap \mathcal{B}$ do
11:            for $\mathcal{T}_j \in \mathcal{B} \setminus \mathcal{T}_i$ in random order do
12:                Compute $\phi_{ijk}^{(t)} \leftarrow \frac{\mathbf{g}'_{ik} \cdot \mathbf{g}_{jk}}{\|\mathbf{g}'_{ik}\| \|\mathbf{g}_{jk}\|}$
13:                if $\phi_{ijk}^{(t)} < \hat{\phi}_{ijk}^{(t)}$ then
14:                    Set $\mathbf{g}'_{ik} = \mathbf{g}'_{ik} + \frac{\|\mathbf{g}'_{ik}\| (\hat{\phi}_{ijk}^{(t)} \sqrt{1 - (\phi_{ijk}^{(t)})^2} - \phi_{ijk}^{(t)} \sqrt{1 - (\hat{\phi}_{ijk}^{(t)})^2})}{\|\mathbf{g}_{jk}\| \sqrt{1 - (\hat{\phi}_{ijk}^{(t)})^2}} \cdot \mathbf{g}_{jk}$
15:                end if
16:                Update $\hat{\phi}_{ijk}^{(t+1)} = (1 - \beta)\hat{\phi}_{ijk}^{(t)} + \beta \phi_{ijk}^{(t)}$
17:            end for
18:        end for
19:        Update $\boldsymbol{\theta}_k$ with gradient $\sum \mathbf{g}'_{ik}$
20:    end for
21:    Update $t \leftarrow t + 1$
22: end while
```

Figure 5: Algorithm for Gradient Vaccine Update Rule.