Good Things Come to Those Who Weight: Effective Pairing Strategies for Multi-Task Fine-Tuning

Stanford CS224N Default Project

Nachat Jatusripitak
Department of Computer Science
Stanford University
nachatj@stanford.edu

Pawan Wirawarn
Department of Computer Science
Stanford University
pawanw@stanford.edu

Abstract

Although multi-task fine-tuning is a promising technique for improving NLP task performance in theory, empirical studies have shown that it can significantly decrease performance compared to single-task fine-tuning. The goal of our project is to explore effective task pairing strategies for improving fine-tuning performance on three downstream NLP tasks. We characterize task relationships by comparing the performance of models trained with paired tasks to single-task fine-tuned baselines while controlling for dataset size. We observe overarching trends in multi-task fine-tuning performance across different loss combination functions. Our results show that paraphrase detection benefits significantly from multi-task fine-tuning, especially when paired with semantic similarity. We find that unitary scalarization outperforms other loss combination techniques but only with proper weighting, which is necessary to realize performance improvements over single-task models.

1 Key information

- Mentor: David Lim
- External Collaborators (if you have any): None
- Sharing project: Yes
- Team contributions: We conducted all of the research, programming, experiments, writing, and poster-making as a pair. All of our work was done together with equal contribution.

2 Introduction

Multi-task learning emulates how humans learn multiple skills such that these skills can benefit each other. For example, reading enhances a person's ability to write because they gain a wider vocabulary and incorporate new literary techniques into their prose. In the context of machine learning models, Caruana (1997) characterizes multi-task learning as "an inductive transfer mechanism whose principle goal is to improve generalization performance." Multi-task learning involves using a shared representation across multiple tasks while training on them in parallel. In theory, what the model learns from one task can help it learn other tasks better.

However, the potential of multi-task learning has led to significant performance benefits in real-world machine learning models. Previous studies have shown that multi-task learning can significantly hurt model performance compared to conventional single-task training. These results have been attributed to so-called "conflicting" tasks, which Yu et al. (2020) characterizes as the presence of conflicting gradients between tasks during training. The problem of which tasks are considered "related" or not remains an open question in machine learning research, given that there is no clear definition of

task relatedness (Worsham and Kalita, 2020). Understanding which tasks work well together is an important step in realizing the benefits promised by multi-task learning.

The goal of our research is to study how tasks should be combined to effectively increase model performance. In doing so, we answer two questions: "which tasks should be paired" and "how should tasks be combined." We fine-tune a pre-trained minBERT model on different task pairings and observe how they affect task performance on sentiment analysis, paraphrase detection, and semantic similarity. We characterize task relationships by comparing these scores to single-task fine-tuned baselines while controlling for dataset size. In addition, we experiment with loss addition, unitary scalarization, and PCGrad, to see which loss combination techniques are effective for our use case.

Our results show that paraphrase detection benefits the most from multi-task fine-tuning, followed by sentiment analysis, and semantic similarity least of all. Paraphrase detection and semantic similarity mutually benefit each other. We observe that unitary scalarization generally outperforms loss addition and PCGrad. The strong performance of unitary scalarization biased towards a "main" task suggests that multi-task fine-tuning should be viewed not as an approach to train a truly multi-purpose model, but rather a technique to improve performance on a specific task.

3 Related work

Multi-task learning is not a new concept in machine learning literature. Caruana (1997) proposed that multi-task learning could improve generalization by allowing features developed in the hidden layer for one task to be used by other tasks in backpropagation. However, the same paper found that multi-task learning sometimes hurt task performance.

A number of multi-task optimization (MTO) methods have been developed to address these performance drops. Sener and Koltun (2018) solves for a Pareto optimal solution across multiple task-specific loss objectives instead of combining these losses into a single objective. Yu et al. (2020) uses gradient surgery to project conflicting task gradients onto each other, with positive results in reinforcement learning tasks. However, work by Kurin et al. (2022) and Xin et al. (2022) dispute the efficacy of MTO approaches, citing high memory and computational overhead as well as mixed empirical results.

Other research has focused on the task relationships themselves. Zamir et al. (2018) studied task relationships in the context of transfer learning, which Standley et al. (2020) built upon Zamir et al. (2018) by performing a similar analysis for multi-task learning in computer vision. Standley et al. (2020) discovered that some tasks improve others' performance at the expense of its own. They found that multi-task learning generally improved performance when conducted on a set of similar tasks. However Standley et al. (2020) combined the task losses by averaging them; our study tests a wider range of loss combination techniques in case performance is sensitive to the method used. We have seen limited studies of task relationships in NLP literature, which is a gap that we aim to fill with this research.

4 Approach

4.1 minBERT

We first completed a minimalist implementation of BERT, a transformer-based model that generates contextual word representations (Devlin et al., 2018). We implemented minBERT by referencing the skeleton code and the default project handout.

For each task, we implemented a prediction head consisting of a hidden block followed by a task-specific block, shown in Figure 1. The hidden blocks consist of a dropout layer, a linear projection layer, and a ReLU activation. Each task-specific block consists of a dropout layer, a linear projection layer, and a task-specific activation. We chose the softmax function for sentiment classification, the sigmoid function for paraphrase detection, and ReLU for semantic similarity, which are well-suited for their respective tasks. The prediction heads take in the `[CLS]` token embedding and pass it through the hidden block and then the task-specific block, returning the output.

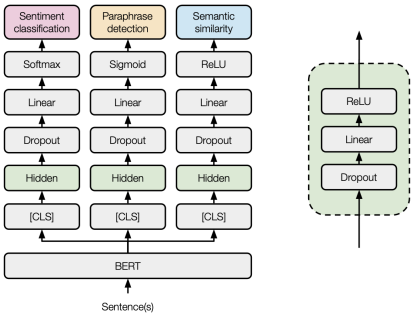

Figure 1: Diagram of model architecture, with task-specific prediction heads

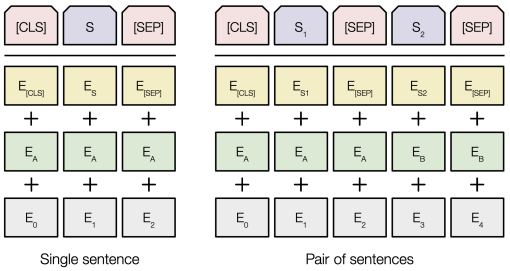

Figure 2: Tokenization strategy for single sentence and sentence pair inputs

We modified the skeleton code to implement our tokenization approach, which follows the procedure described in Devlin et al. (2018). Figure 2 describes how we tokenized our inputs to the various tasks, depending on the number of input sentences.

4.2 Controlling dataset sizes

In our preliminary experiments, we found that the unbalanced dataset sizes skewed performance improvements towards tasks with larger datasets while hurting tasks with smaller datasets, motivating the need to control for dataset size. These results can be found in Appendix A.

Some sampling strategies to address size imbalances include oversampling smaller datasets and undersampling larger datasets. We decided to undersample larger datasets to avoid the problem of overfitting on the smaller datasets. We use PyTorch Lightning's CombinedLoader, set to 'min_size' mode, which stops sampling after the smallest dataset is exhausted. This approach ensures that all fine-tuning sessions train the model on the same quantity of data for each task.

4.3 Fine-tuning on single and multiple tasks

We will sometimes abbreviate the task names using acronyms: sentiment analysis (`SEN`), paraphrase detection (`PAR`), and semantic textual similarity (`SIM`). We implemented code to fine-tune minBERT on combinations of these tasks. We use our implementation of the AdamW optimizer (Loshchilov and Hutter, 2019) to minimize loss, saving model weights with the highest dev set score. We use cross-entropy loss for sentiment analysis and paraphrase detection and mean squared error loss for semantic similarity.

Our baselines are single-task fine-tuned models. As our experiment, we fine-tune minBERT on all possible task pairings: `SEN + PAR`, `PAR + SIM`, and `SEN + SIM`. We combine losses using the techniques described in the following section.

4.3.1 Combining loss gradients

The naive approach to combining task losses is to sum them:

$$\mathcal{L}_{add} = \mathcal{L}_1 + \mathcal{L}_2 \qquad \nabla\mathcal{L}_{add} = \nabla\mathcal{L}_1 + \nabla\mathcal{L}_2$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are the losses for the first and second task, respectively.

Another approach for combining task losses is to perform unitary scalarization, where the combined loss is a convex combination of the individual losses:

$$\mathcal{L}_{unit} = \alpha\mathcal{L}_1 + (1-\alpha)\mathcal{L}_2 \qquad \nabla\mathcal{L}_{unit} = \alpha\nabla\mathcal{L}_1 + (1-\alpha)\nabla\mathcal{L}_2$$

where $\alpha \in [0, 1]$ is a weight parameter.

Yu et al. (2020) propose a multi-task optimization technique called PCGrad, a form of gradient surgery, to address the problem of conflicting gradients. If two task gradients conflict (i.e. have negative cosine similarity), PCGrad replaces the gradient of the i-th task $\nabla\mathcal{L}_i$ onto the normal plane

of another conflicting tasks' gradient $\nabla \mathcal{L}_j$:

$$\nabla \mathcal{L}_i^{PC} = \nabla \mathcal{L}_i - \frac{\nabla \mathcal{L}_i \cdot \nabla \mathcal{L}_j}{\|\nabla \mathcal{L}_j\|^2}$$

These projected gradients are then summed to form the combined loss gradient. We use Tseng's PyTorch implementation of PCGrad.

5 Experiments

5.1 Data

We use the Stanford Sentiment Treebank (SST) (Socher et al., 2013) for sentiment analysis, the Quora dataset (Shankar Iyer, 2017) for paraphrase detection, and the SemEval dataset (Agirre et al., 2013) for semantic textual similarity. The SST consists of 11,855 single sentences from movie reviews, which are parsed into 215,154 unique phrases. Each phrase has a label of negative, somewhat negative, neutral, somewhat positive, or positive. The Quora dataset consists of 400,000 question pairs with labels indicating whether or not the pairs are paraphrases of each other. The SemEval dataset consists of 8,628 sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning).

5.2 Evaluation method

We evaluate the scores achieved by the multi-task models against the baselines to quantify the impact of task pairings on performance on the three downstream tasks. We use prediction accuracy for sentiment analysis and paraphrase detection, and Pearson correlation for semantic similarity and calculate the relative score change. Our absolute scores can be found in Appendix B.

5.3 Experimental details

We fine-tune minBERT on the following task groupings: (1) `SEN`, (2) `PAR`, (3) `SIM`, (4) `SEN + PAR`, (5) `PAR + SIM`, and (6) `SEN + SIM`. We use a batch size of 8, hidden dropout probability 0.5, and train for 10 epochs. We used weight decay 0.01, learning rate 10^{-5}, $\varepsilon = 10^{-6}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. This process is repeated for each loss combination technique. For unitary scalarization, we test weights from 0.1 to 0.9 in increments of 0.1.

5.4 Results

Note: we abbreviate the names of the unitary scalarization models in the form $\texttt{TASK}_1(\alpha_1) + \texttt{TASK}_2(\alpha_2)$. For example. `SEN (0.2) + PAR (0.8)` refers to the `SEN + PAR` task pairing with unitary scalarization where `SEN` has weight 0.2 and `PAR` has weight 0.8.

5.4.1 Sentiment analysis

In the sentiment analysis task, we found that the `SEN (0.4) + PAR (0.6)` pairing and the `SEN + SIM` pairing with loss addition scored the highest on the dev set, as shown by the heatmap in Table 1. Both the `SEN (0.4) + PAR (0.6)` pairing and `SEN (0.9) + SIM (0.1)` pairing scored 1.35% higher than the baseline. Loss addition did not improve the score relative to the baseline and PCGrad only showed improvement for the `SEN + PAR` pairing.

We performed quadratic regressions for the unitary scalarization models to identify the relationship between the value of the `SEN` weight parameter and the relative change in accuracy score on the `SEN` task. The `SEN + PAR` trendline, shown in Figure 3, is loosely quadratic ($R^2 = 0.644$) and shows significant variance while the `SEN + SIM` trendline appears virtually linear ($R^2 = 0.909$). Neither seem to follow well-defined functions, but the accuracy scores fall off precipitously as $\alpha \to 0$.

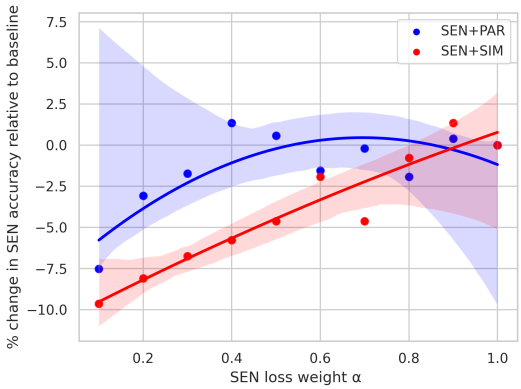

Figure 3: Relationship between SEN loss weights and relative SEN accuracy change

Loss function	SEN+PAR	SEN+SIM
α = 0.1	-7.51	-9.63
α = 0.2	-3.08	-8.09
α = 0.3	-1.73	-6.74
α = 0.4	1.35	-5.78
α = 0.5	0.58	-4.62
α = 0.6	-1.54	-1.93
α = 0.7	-0.19	-4.62
α = 0.8	-1.93	-0.77
α = 0.9	0.39	1.35
Addition	-1.54	-5.59
PCGrad	0.39	-5.01

Task pairing

Table 1: % change in SEN accuracy relative to baseline by loss function used

5.4.2 Paraphrase detection

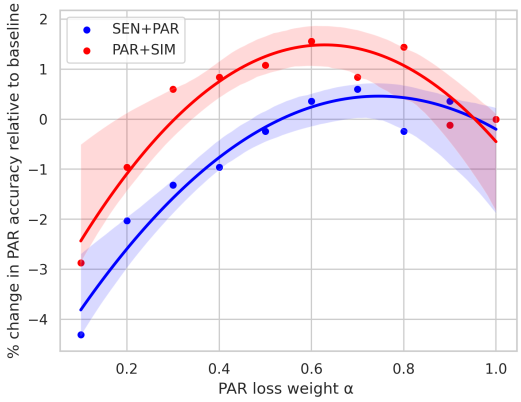

Figure 4: Relationship between PAR loss weights and relative PAR accuracy change

Loss function	SEN+PAR	PAR+SIM
α = 0.1	-4.31	-2.87
α = 0.2	-2.03	-0.96
α = 0.3	-1.32	0.60
α = 0.4	-0.96	0.84
α = 0.5	-0.24	1.08
α = 0.6	0.36	1.56
α = 0.7	0.60	0.84
α = 0.8	-0.24	1.44
α = 0.9	0.36	-0.12
Addition	-0.36	0.84
PCGrad	0.00	0.96

Task pairing

Table 2: % change in PAR accuracy relative to baseline by loss function used

In the paraphrase detection task, we find that the `SEN (0.3) + PAR (0.7)` pairing and the `PAR (0.6) + SIM (0.4)` pairing score the highest on the dev set, as shown by the heatmap in Table 2. The `SEN (0.3) + PAR (0.7)` pairing scored 0.6% higher than the baseline while the `PAR (0.6) + SIM (0.4)` pairing scored 1.56% higher than the baseline. Both loss addition and PCGrad improved upon the baseline in the `PAR + SIM` pairing but not in the `SEN + PAR` pairing.

Both regression trendlines closely fit the data, with $R^2 = 0.939$ for `SEN + PAR` and $R^2 = 0.897$ for `PAR + SIM`. The regression results, shown in Figure 4, suggest that the relative change in accuracy score is quadratic and that there exists an optimal weight value around 0.75 for the `SEN + PAR` pairing and around 0.6 for the `PAR + SIM` pairing. There is a similar falloff in accuracy as $\alpha \to 0$.

5.4.3 Semantic similarity

In the semantic similarity task, we found that the `SEN + SIM` pairing with loss addition, the `PAR (0.2) + SIM (0.8)` pairing, as well as the baseline produced the highest Pearson correlation scores on the dev set. As shown in the heatmap in Table 3, most models performed slightly worse than the baseline. The `SEN + SIM` pairing with loss addition performed exactly equal to the baseline while the `PAR (0.2) + SIM (0.8)` pairing scored merely 0.11% above the baseline. PCGrad performed especially poorly on both task pairings.

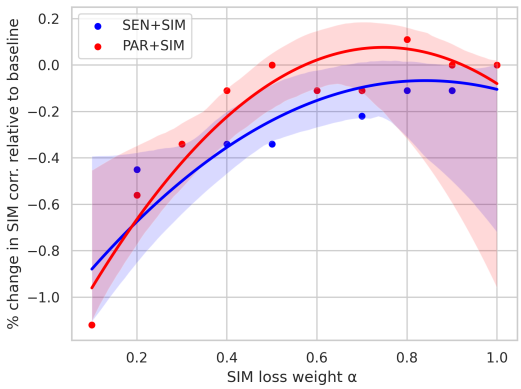

Figure 5: Relationship between SIM loss weights and relative SEN Pearson corr. change

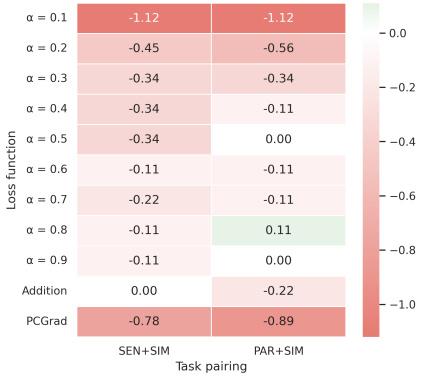

Table 3: % change in SIM Pearson corr. relative to baseline by loss function used

Our regression analyses in Figure 5 found a loosely quadratic relationship for both trendlines. However, neither were particularly close fits, despite $R^2 = 0.802$ for the `SEN + SIM` pairing and $R^2 = 0.901$ for the `PAR + SIM` pairing. Like the other tasks, the Pearson correlation score drops off as $\alpha \to 0$.

5.4.4 Test leaderboard submission

We fine-tuned our models for 25 epochs to boost our scores for the test leaderboard. For sentiment analysis, we chose the `SEN (0.7) + PAR (0.3)` model with no weight decay because we suspected that weight decay caused some underfitting in sentiment analysis. For paraphrase detection, we chose the `PAR (0.8) + SIM (0.2)` model with weight decay 0.01. For semantic similarity, we chose the `PAR (0.2) + SIM(0.8)` model with weight decay 0.01.

Our leaderboard scores are 0.524 for sentiment analysis, 0.866 for paraphrase detection, and 0.898 for semantic similarity, for an average score of 0.779. We were pleased with these results and attribute them to the performance benefits from multi-task fine-tuning, which validates our approach.

6 Analysis

6.1 Discussion of experimental results

Paraphrase detection was the most receptive to multi-task fine-tuning among the three tasks. Accuracy scores on the `PAR` dev set improved by up to 0.6% and 1.56% when paired with `SEN` and `SIM`, respectively. In this task, multi-task fine-tuning improved on the baseline across nearly all loss combination techniques, with the exception of low values of α in the unitary scalarization models and some occasional outliers.

However, sentiment analysis and semantic similarity did not show the same improvement from multi-task fine-tuning. Although this observation could indicate that there is something inherent about the paraphrase detection task that enables it to improve more, a more likely explanation is that the `PAR` models are simply under-trained. In our approach, we had to control for dataset size by heavily undersampling from the `PAR` dataset, so the `PAR` models did not have as many opportunities to learn, especially within 10 epochs. Therefore, it is plausible that the model benefited directly from additional information regardless of the nature of the tasks.

Our results show that paraphrase detection and semantic similarity make a good task pairing. The `PAR + SIM` pairing outperformed the `SEN + PAR` pairing on paraphrase detection while the `PAR + SIM` pairing did not hurt performance on semantic similarity as much as the `SEN + SIM` pairing. These results imply that `PAR + SIM` is a mutually beneficial pairing. Indeed, paraphrase detection and semantic similarity are fundamentally related sentence pair tasks: sentences that are paraphrases of each other often share a similar meaning, though the former is a binary classification task while the latter is a regression task. Along these lines, the `SEN + PAR` pairing likely performed better than the

PAR + SIM pairing on sentiment analysis because both sentiment analysis and paraphrase detection are classification tasks.

From our results, we see that multi-task fine-tuning generally led to modest performance improvements. The strong overall performance of unitary scalarization with $\alpha > 0.5$ indicates that weights should be biased towards one task rather than being split equally. Intuitively, biased weighting is equivalent to the model being focused on one main task while learning from another auxiliary task on the side. When α becomes too small, we put more emphasis on the auxiliary task than on the main task. The model becomes "distracted" by the auxiliary task, focusing too much on it, leading to the performance dropoff that we observed. On the other hand, equal weighting leads to a model that is decent at both tasks but good at neither: a jack-of-all-trades but a master of none. This idea explains why PCGrad performed better than loss addition but worse than unitary scalarization; PCGrad retains loss addition, but with additional treatment of conflicting gradients.

6.2 Qualitative evaluation of model behavior

We proceed with a qualitative evaluation of some model behavior by computing confusion matrices for sentiment analysis and paraphrase detection, and generating scatterplots for semantic similarity. For each task, we select the top performing models and compare their behavior to the baseline to understand how particular task pairs affect the distribution of predicted classes and values.

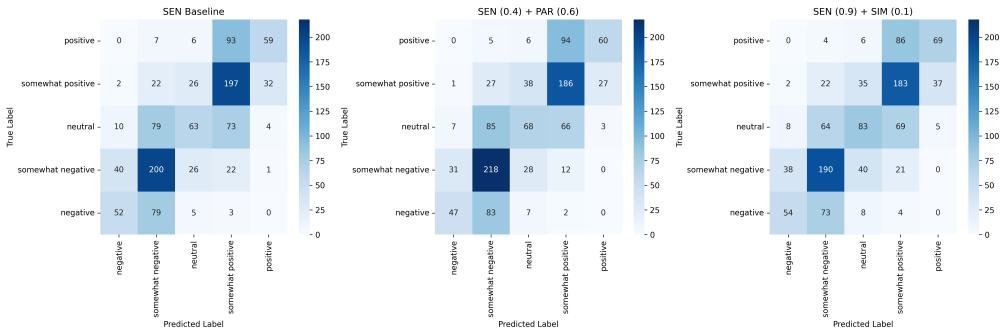

Figure 6: Confusion matrices for sentiment analysis task.
Models (left to right): SEN baseline, SEN (0.4) + PAR (0.6), SEN (0.1) + SIM (0.9)

Sentiment analysis: SEN + PAR is better than the baseline and SEN + SIM at predicting the 'somewhat negative' class accurately. With over 200 examples in the 'somewhat negative' class, this comparative advantage accounts for SEN + PAR's high performance on sentiment analysis, even though it is worse than the baseline at predicting the 'somewhat positive' class. As shown in Figure 6, SEN + PAR tends to falsely classify these examples as 'neutral' and 'somewhat negative.' On the other hand, SEN + SIM is better than the baseline and SEN + PAR at predicting the 'negative,' 'neutral,' and 'positive' classes, which have fewer examples between them. As a regression task that predicts values across a continuous range, SIM may have influenced the model to discern inputs across the whole sentiment spectrum with more nuance.

Paraphrase detection: Both top-performing SEN + PAR models are better at detecting paraphrases but worse at detecting non-paraphrases than the baseline, as shown in Figure 7. This change could be caused by the presence of SEN, but the exact effect on this behavior is unclear because the model with lower SEN weight was better at detecting paraphrases than the one with higher weight. Meanwhile, PAR + SIM outperformed the baseline in detecting both paraphrases and non-paraphrases with no clear drawbacks. It is clear from this analysis that PAR benefits more from being paired with SIM than it does from SEN under our experimental conditions.

Semantic similarity: We generated a scatter plot for each of these models to compare the predicted values to the true values and overlaid a line to represent ideal model predictions. As shown in Figure 8 baseline predictions roughly follow this line with moderate variance but no apparent bias. With PAR + SIM, there is a clear difference in predictions for examples with high true similarity values. As shown by a leftward "curve" of the scatterplot, the PAR + SIM model tends to underpredict these examples. A possible explanation is that this occurs due to the binary nature of PAR influencing the model to make more clustered predictions. This bias is offset by PAR + SIM's predictions being

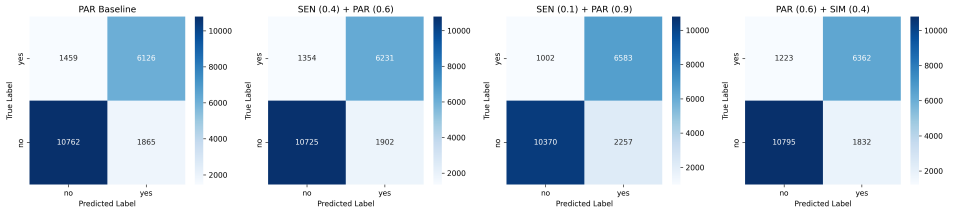

Figure 7: Confusion matrices for paraphrase detection task.
Models (left to right): `PAR` baseline, `SEN (0.4) + PAR (0.6)`, `SEN (0.1) + PAR (0.9)`, `PAR (0.6) + SIM (0.4)`

less spread out from the line, which explains its slightly higher Pearson correlation score than the baseline. These effects are nearly absent in the case of `SEN + SIM`, where the overall plot looks remarkably close to the baseline in variance and "slope," corresponding to comparable performance to the baseline.

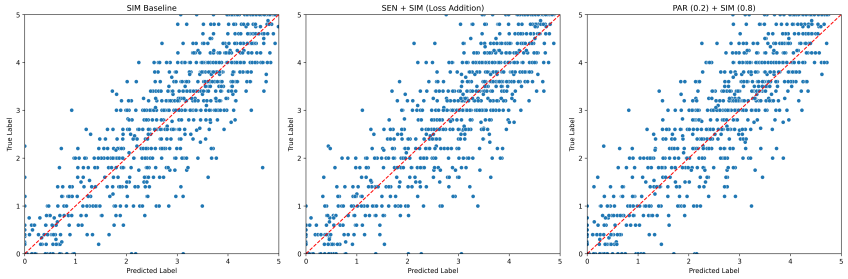

Figure 8: Scatter plots for semantic similarity task with predicted and true label on the horizontal and vertical axis, respectively. Models (left to right): `SIM` baseline, `SEN + SIM` with loss addition, `PAR (0.2) + SIM (0.8)`

7 Conclusion

We investigate the impact of task pairings on multi-task fine-tuning performance on three downstream NLP tasks and found that paraphrase detection benefits the most from multi-task fine-tuning, followed by sentiment analysis and semantic similarity. Our results indicate that paraphrase detection and semantic similarity are compatible because they mutually improve each other's performance, especially on paraphrase detection. Overall, unitary scalarization is an effective method for combining multiple loss functions for multi-task fine-tuning. We find that unitary scalarization should be biased towards a main task, which reliably receives a performance boost from the addition of an auxiliary task when given correct weighting.

Although these results were backed by attempts to control for as many variables as possible, there remain a number of limitations to our study. In particular, our study used only one random seed and did not perform hyperparameter tuning due to computational and time constraints. Model performance is highly sensitive to hyperparameters and other training parameters, so these results may not generalize to all fine-tuning cases. In addition, our data on unitary scalarization performance could benefit from more data points, which would come from fine-tuning models with smaller spacing between weight values. As we mentioned, the paraphrase detection results should be interpreted with caution because our approach highly undersampled its dataset, which may have led to underfitting and further potential for learning if trained for more epochs.

From the limitations of our study, we see many avenues for further research in this area. In particular, future work can be done to extend the benefits of unitary scalarization to PCGrad, which is a potent technique in theory but underperformed in our experiments. It would be interesting to characterize the sensitivity of multi-task fine-tuning to changes in hyperparameters. Finally, we see potential to investigate task pairings for other tasks to see if they also increase performance compared to single-task fine-tuning.

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.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

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.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and Pawan K Mudigonda. 2022. In defense of the unitary scalarization for deep multi-task learning. *Advances in Neural Information Processing Systems*, 35:12169–12183.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31.

Kornél Csernai Shankar Iyer, Nikhil Dandekar. 2017. First quora dataset release: Question pairs. `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR.

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

Joseph Worsham and Jugal Kalita. 2020. Multi-task learning for natural language processing in the 2020s: where are we going? *Pattern Recognition Letters*, 136:120–126.

Derrick Xin, Behrooz Ghorbani, Ankush Garg, Orhan Firat, and Justin Gilmer. 2022. Do current multi-task optimization methods in deep learning even help?

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. 2018. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722.

A Absolute task performance scores with imbalanced datasets

Note: these runs were performed with a different architecture and different losses. They only motivate our decision to control for dataset size.

	SEN accuracy score		**PAR accuracy score**		**SIM Pearson correlation**	
Baseline	**0.520**		**0.888**		**0.843**	
Loss Fn.	**SEN + PAR**	**SEN + SIM**	**SEN + PAR**	**PAR + SIM**	**SEN + SIM**	**PAR + SIM**
Addition	0.504	0.524	0.887	0.888	0.835	0.806

B Absolute task performance scores by task grouping and loss function used

	SEN accuracy score		**PAR accuracy score**		**SIM Pearson correlation**	
Baseline	**0.519**		**0.836**		**0.895**	
Loss Fn.	**SEN + PAR**	**SEN + SIM**	**SEN + PAR**	**PAR + SIM**	**SEN + SIM**	**PAR + SIM**
$\alpha = 0.1$	0.480	0.469	0.800	0.812	0.885	0.885
$\alpha = 0.2$	0.503	0.477	0.819	0.828	0.891	0.890
$\alpha = 0.3$	0.510	0.484	0.825	0.841	0.892	0.892
$\alpha = 0.4$	**0.526**	0.489	0.828	0.843	0.892	0.894
$\alpha = 0.5$	0.522	0.495	0.834	0.845	0.892	0.895
$\alpha = 0.6$	0.511	0.509	**0.839**	**0.849**	0.894	0.894
$\alpha = 0.7$	0.518	0.495	0.841	0.843	0.893	0.894
$\alpha = 0.8$	0.509	0.515	0.834	0.848	0.894	**0.896**
$\alpha = 0.9$	0.521	**0.526**	**0.839**	0.835	0.894	0.895
Addition	0.511	0.490	0.833	0.843	**0.895**	0.893
PCGrad	0.521	0.493	0.836	0.844	0.888	0.887