Cuts and Stitches: Does Model Merging Produce Better Multitask Learners?

Stanford CS224N Default Project

Akshana Dassanaike-Perera
Department of Computer Science
Stanford University
akshana@stanford.edu

Suppakit Waiwitlikhit
Department of Computer Science
Stanford University
suppakit@stanford.edu

Koren Gilbai
Department of Computer Science
Stanford University
kgilbai@stanford.edu

Abstract

Parameter averaging has shown promising results as a method of transferring learned capabilities in models in a non-learning setting. In this project, we attempt to explore the use of parameter averaging across models with a shared structure as a means to increase model performance. Specifically, we attempt to independently finetune BERT across three tasks — sentiment classification, paraphrase detection, and semantic equivalence detection — and use Fisher-weighted parameter averaging as introduced by Matena and Raffel (2021) to create a final model that performs well across all three tasks. We implement the same experiment with the parameter averaging method of model soups proposed by Wortsman et al. (2022) to benchmark the usefulness of Fisher-weighted averaging. Furthermore, to evaluate the uniqueness of parameter averaging as a whole, we also implement multitask learning in a gradient-based setting with round-robin style training and gradient surgery Yu et al. (2020) to serve as baselines of other performance-focused frameworks. We evaluate the performance of each method through three uncorrelated measurements and provide in-depth analysis of the performance of each method through ablation studies. We find that parameter averaging through Greedy soups yields the most optimal model in terms of average task performance, matching 98.9% of the performance of the best single-task models. In contrast, our experiments show that parameter averaging through Fisher-weighted averaging fails to increase performance in a multitask setting over its best candidate model.

1 Key Information to include

- Mentor: Gabriel Poesia

2 Introduction

The modern paradigm of NLP has become one of learning through distinct phases of pre-training and fine-tuning. Large, resource-rich institutions push out increasingly larger models pre-trained on increasingly larger datasets that are then fine-tuned on smaller subsets of data for domain specific tasks by smaller groups or individuals. The sheer size of these models make pre-training them unfeasible on an individual basis. Furthermore, this paradigm raises some natural questions. How can we train one model to do well across multiple tasks? In other settings, how can we derive joint-task performance from multiple fine-tuned models?

Matena and Raffel (2021) attempts to explore alternatives from gradient-based fine-tuned learning in the context of transfer learning through parameter merging. When merging multiple candidate models, they use a candidate model's Fisher information matrix to scale how much weight its parameters get in the resulting end model. Their findings show promising results for Fisher-weighted averaging, but they note that their results are preliminary. Additionally, they only explore Fisher-weighted averaging in the context of multiple models trained on intermediate tasks; we want to explore Fisher averaging in the context of multiple multitask models.

In this work, we investigate parameter averaging through Fisher-weighted averaging and model soups in the context of model multitask learning. We first train multiple models across the tasks of sentiment classification, paraphrase detection, and semantic textual similarity. We then explore the use of parameter averaging for increasing model performance with unseen data across all three tasks.

3 Related Work

Wortsman et al. (2022)'s methodology of "model soups" demonstrated the use of parameter averaging to increase model performance. They found that averaging the weights of multiple fine-tuned models with different hyperparameter configurations can increase model accuracy and robustness. Specifically, they found success with "greedy soups", where models were merged by iteratively including candidate models that improve accuracy on held-on out data. "Greedy soups" showed performance benefits in both accuracy and robustness to distribution shifts when compared to picking the individual model with the highest accuracy.

As opposed to "greedy soups", Matena and Raffel (2021) proposed Fisher-weighted averaging, a novel methodology for parameter averaging that weights candidate models' parameters according to their Fisher information matrix. The Fisher information matrix, formally defined as the covariance matrix of a model's score function, can be interpreted as the sensitivity of a model's likelihood function with respect to each of its parameters. Matena et al. interprets the Fisher information matrix as the Laplace approximation of the model's posterior distribution. Thus, Fisher-based merging can be viewed as maximizing the resulting model's posterior distribution with respect to the candidate models. Matena et al. found that Fisher-weighted averaging outperformed gradient-based learning in four out of the seven benchmarks they tested.

Matena and Raffel (2021) uses Fisher-weighted averaging to merge models trained on intermediate tasks and Wortsman et al. (2022) uses greedy soups to merge models trained with different hyper-parameter configurations. Both works demonstrate promising results for parameter averaging as a method to increase model performance and robustness to out of distribution data for a particular task. We extend on their findings by evaluating parameter averaging in a multitask setting.

4 Approach

To evaluate parameter averaging in a multitask setting, we train and evaluate a pretrained BERT model on the tasks of sentiment classification, paraphrase detection, and semantic textual similarity (Devlin et al., 2018). We use HuggingFace's BERT base model uncased with a task-specific head for each task as outlined in figure 1. The sentiment classification head takes in BERT's sentence embedding and applies a linear layer and softmax to generate predicted probabilities for each sentiment class. The paraphrase detection head takes in two BERT sentence embeddings and applies mean-pooling and a multilayered perceptron (MLP) to predict if one of the sentences is a paraphrase of the other. The semantic textual similarity head takes in two BERT sentence embeddings and applies mean-pooling, a linear layer, and cosine similarity to identify the semantic similarity of the two sentences.

4.1 Baselines

We utilize 4 baselines to contextualize our experimental findings. First, we finetune 3 BERT base models independently across all three tasks to create three model baselines, each one finetuned on one task. We evaluate each model on the task it was trained on to provide a single-task baseline for each task. For our last baseline, we train one BERT base model round-robin style where the model trains on each task sequentially within one epoch.

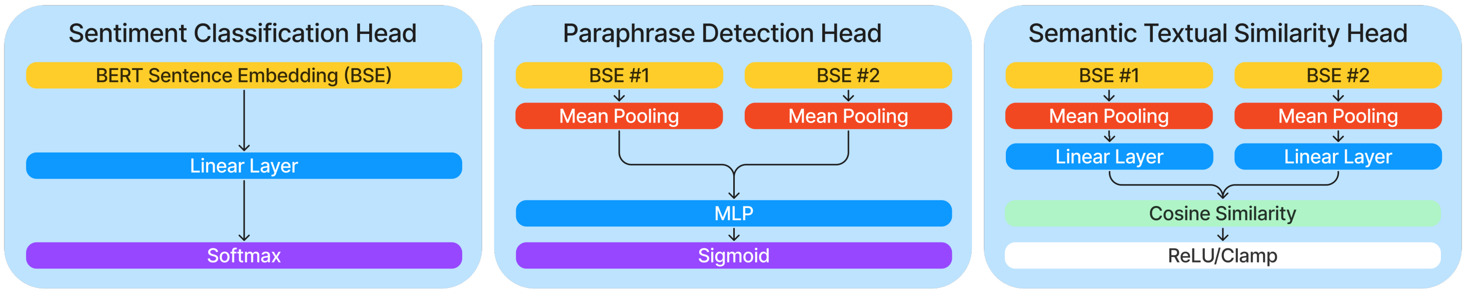


Figure 1: Task-Specific Output Heads

4.2 Gradient Surgery

In addition to round-robin style training, we implement gradient surgery from Yu et al. (2020) to help evaluate the uniqueness of parameter averaging. Before applying gradient updates, we compute the loss for a batch of training examples from each task and then backpropagate to construct the model's computation graph for each task. However, before applying gradient updates, we project each task's gradient onto every other task's gradients on a per-parameter basis, and subtract any conflicting projections to align the direction of all gradient updates.

On top of the architecture proposed by Yu et al. (2020), we add a hand-tuned weighting parameter to individually weight the contribution of each task's gradient in the final gradient computation.

4.3 Model Soups

We implement "greedy soups"-based parameter averaging from Wortsman et al. (2022) as one method of parameter averaging across already trained models. In line with their "greedy soups" algorithm, we sort candidate models in decreasing order of their average accuracy on the validation splits of our three tasks and iterate through each model, only adding adding a candidate model to our "soup" if its addition increases the resulting model's average accuracy. To compute the resulting model from our included candidate models, we compute the mean of all included candidate models' parameters.

```
Greedy Soups Algorithm
sort candidate models in decreasing order of
      average val set accuracy
best_val_acc = 0
soup = {}
for i from 1 to len(models):
      if avg_val_acc(merge(soup U models[i]))
                  > best_acc:
            soup = soup U models[i]
            best_acc =
avg_val_acc(merge(soup))

return merge(soup)
```

Figure 2: Greedy soups pseudocode

4.4 Fisher-Weighted Averaging

As opposed to model soups, we implement Fisher-weighted averaging as proposed in as proposed in Matena and Raffel (2021). As opposed to model soups which computes the resulting model based off an unweighted average, Fisher-weighted averaging computes a weighted average of candidate models based off an approximation of the candidate models' Fisher information matrices. Storing the true Fisher information matrix requires $O(|\theta|^2)$ space complexity (where θ represents the model's parameters), so we approximate it by only calculating the diagonals of the Fisher information matrix as:

$$\hat{F_\theta} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{y\sim p(y|x_i;\theta)}[(\nabla_\theta \log p(y|x_i))^2] \tag{1}$$

```
Fisher-weighted Averaging
fisherMatrices = compute_fisher_matrices(models)
M_final, D = {}, {}  # final model parameters and
                     # divisors
for model M in models:
      compute the weighted average of Fisher information
            matrices F[M] for each parameter theta
      update M_final and D with the computed values

create the final model M* by dividing the final model
    parameters in M_final by their corresponding divisors
    in D

return M*
```

```
Compute Fisher Information Matrices
fisherMatrices = {}
    for model M in models:
        for task T in tasks:
             for data in T[:SAMPLE_SIZE]:
                   if T.type is regression:
                         compute regression result μ
                         sample 5 probabilities
      gaussian_p from Gaussian Kernel N ~ (μ, σ^2)
                         compute gradients ∇_θ log
      gaussian_p with respect to M weights
                   if T.type is classification:
                          compute logits and log_logits
                          compute gradients ∇_θ
      log_logits with respect to M weights
        Update fisherMatrices[M_i] with gradient values

return fisherMatrices
```

Figure 3: Fisher-merging pseudocode

To maximize the resulting model's posterior distribution with respect to the candidate models, we compute $\theta^* = argmax_\theta \sum_{i=1}^{M} \lambda_i \log p(\theta|\theta_i, F|i)$, which has the closed form solution:

$$\theta^{*(j)} = \frac{\sum_i^M = 1\lambda_i F_i^{(j)} \theta_i^{(j)}}{\sum_i^M = 1\lambda_i F_i^{(j)}} \tag{2}$$

where λ_i is a hyperparameter that is hand-tuned to weigh model-importance during merging such that $\sum_i \lambda_i = 1$. An intuitive way to think of Fisher-Weighted averaging is that Fisher-weighting gives the output's 'dependence' on each parameter.

In order to compute equation 1 in the context of semantic similarity, we attempted to quantize the output of our model's STS head into discrete outputs. Our initial experiments showed that this led to a steep performance decline, and therefore, we extend the original paper by instead estimating the posterior distribution of STS with a Gaussian Kernel with a mean of the regression output and sampling a few points to bootstrap the Fisher information matrix.

5 Experiments

We design our experiments with three research questions in mind:

RQ 1: Can we match single-task training performance in a multitask setting?
RQ 2: Can parameter averaging increase performance in a multitask setting?
RQ 3: Can Fisher-weighted averaging increase performance over other parameter averaging methods in a multitask setting?

We discuss each research question throughout our experimental results and analysis and directly answer them in our conclusion.

5.1 Data

We will be using the SemEval STS Benchmark Dataset (STS) for the task of semantic equivalence, the Quora Dataset for Paraphrase Detection (QDPD) for the task of paraphrase detection, and the Stanford Sentiment Treebank (SST) for the task of sentiment classifcation. Additionally, for certain experiments, we use the Stanford Natural Language Inference (SNLI) corpus for further pretraining.

5.2 Evaluation method

We will evaluate model performance on SST and QDPD using accuracy on each dataset. For the STS dataset, we calculate the Pearson correlation of predicted similarity to true similarity instead of accuracy. As a measure of robustness across tasks, we will compute performance for every model across all validation sets of SST, QDPD, and STS to quantify performance in a multitask setting.

5.3 Experimental details

Unless otherwise stated, all experiments are ran over the course of 10 epochs with a learning rate of $1e^{-5}$ and a dropout rate of 0.3. All experiments ran were "fine-tuning" experiments in that we updated the parameters of our pretrained BERT model at training time. The MLP in the paraphrase detection head has a hidden layer size of 256; all other heads have a hidden layer size of 128. Finally, all experiments utilize weight decay with $\lambda = 0.1$.

The specific training details vary across experimental runs; we experiment with round-robin style training and joint multitask learning with gradient surgery as outlined in sections 4.1 and 4.2. When merging models with model soups and Fisher-weighted averaging, we follow the procedures as outlined in 4.3 and 4.4, respectively.

When running experiments with gradient-surgery, we experiment with different weightings on each task's gradients.

For specific experimental runs, we experiment with intermediary task training our model on the SNLI dataset before training on SST, QDPD, and STS. When training on the SNLI dataset, we train our model on sentence classification for 20 epochs with the same hyperparameters as above.

In other runs, we implement SMART regularization with Kullback-Leibler divergence as outlined in Jiang et al. (2020) to explore the impact of input-based regularization in a multitask setting.

5.4 Results

Table 1: Results for experiments evaluated on the dev splits of task datasets.

Experiment	SST	QDPD	STS	Avg
Single-task training on SST	0.491			N/A
Single-task training on QDPD		0.764		N/A
Single-task training on STS			0.801	N/A
Round-robin multitask training	0.439	0.736	0.620	0.598
Gradient-Surgery multitask training	0.473	0.727	**0.796**	0.665
Gradient-Surgery w/ SMART	0.478	0.748	0.790	0.672
Gradient-Surgery w/ SNLI intermediary training	0.476	**0.761**	0.790	0.676
Greedy soups Merging*	**0.480**	0.758	0.795	**0.678**
Fisher-weighted Merging*	0.478	0.753	0.778	0.670

*Note: The Greedy soups and Fisher-weighted averaging trials were conducted by merging the three Pareto-optimal multitask models for each task (the three Gradient-surgery models listed above). More explanation is given in the analysis in section 6.

5.4.1 Single-Task Results

In order to find an optimal baseline results for each task, we trained models with various hyperparameters. For SST, we found that changes in the prediction head architecture had little effect on model performance. As such, we stuck to one linear layer with softmax to achieve our best result.

When training paraphrase detection, we experimented with multiple depth-configurations but found that prediction head depth had little impact on the results. We experimented using cosine similarity between sentence embeddings for paraphrase detection, but found that using an MLP yielded the best results.

For STS, we implemented the Siamese BERT architecture from Reimers and Gurevych (2019), and used cosine similarity to determine the similarity of sentences. After analyzing our model's output, we deduced that our model struggles to produce embedding pairs that have high cosine similarity, despite the sentences having high similarity indexes (4.0-5.0). We hence apply a clamp function on top of our cosine similarity to make the conditions for similarity more lenient. We find that adding clamping resulted in a 6% boost in performance. We provide in section 6.2 a comparison of the model accuracy on STS when applying various non-linear activations, ReLU, clamp(0, 0.9), and clamp(0, 0.8), after Cosine Similarity.

We note that none of the multitask models were able to outperform the best single-task training benchmarks. However, the average performance across the best single-task models is 0.685, indicating that our best multitask model (Greedy soups merging) matches 98.9% performance of its single-task counterparts.

5.4.2 Multitask Results

All of our multitask methods out-perform the round-robin baseline. For the standard Gradient-Surgery without SMART regularization or SNLI intermediary task training, we weight the gradients of SST, QDPD, and STS as $[0.5, 1, 1]$, respectively. During experimentation, we found that the gradients of SST can be quite dominating, especially with respect to the gradients of STS, so proportionately muting SST-related gradients benefited average task performance. In turn, our gradient-surgery model had the best multitask performance on STS dev set Pearson correlation.

Implementing SMART regularization led to incremental benefits in SST and QDPD accuracy over standard gradient surgery. Input-based regularization makes sense in a multitask setting, as the distribution of inputs that the shared-BERT model will encounter will naturally be wider than in a single task setting. As such, SMART regularization increased average task performance by 1%.

Adding intermediary task training on SNLI led to increased performance on QDPD by approximately 5%. Reimers and Gurevych (2019) saw a 1-2% performance boost on semantic textual similarity prediction when first training on SNLI before STS with their S-BERT architecture. Our results show a slight decrease in STS performance, but we believe this is a natural byproduct of the increase in performance on QDPD.

Greedy soups merging saw the best SST accuracy and best average task performance out of all multitask models. Greedy soups was able to find a point in the error-basin of candidate models that was lower than any individual model, further supporting its merits in finding unique minima over gradient-based learning even in a multitask setting.

Fisher-weighted averaging did not see exceptional performance in any metric. We do note that it does out-perform standard Gradient-Surgery by 1%, but it does not out perform its candidate models. We also experimented with Fisher-weighted merging across the three single-task baselines which yielded poor performance across all results. We believe that Fisher-weighted averaging struggles in a multitask setting, which we elaborate further on in the analysis section.

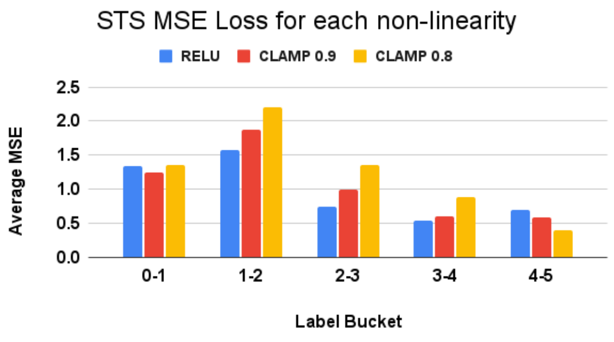
6 Analysis

6.1 Paraphrase Detection

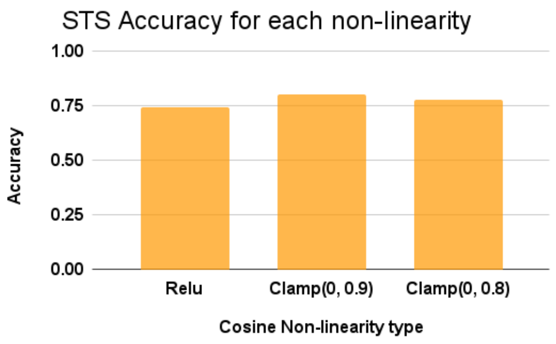
For paraphrase detection, we initially experimented with using cosine similarity on BERT sentence embeddings to detect paraphrases. However, after optimizing our STS head (which also utilized cosine similarity), we found that model performance on QDPD using cosine similarity was negatively correlated with model performance on STS using cosine similarity. We believe that sharing similarity metrics across tasks does not hold out in a multitask setting; optimizing the sentence embedding space for cosine similarity in a paraphrase-based context detracts from optimizing the sentence embedding space in a similarity-based context. As a result, we found that using an MLP for paraphrase detection gave us the best performance without sacrificing STS performance.

6.2 Semantic Textual Similarity

Our initial experiments demonstrated that our STS head struggled to identify similar sentences. Because the STS head uses cosine similarity, a similarity score of 5 would only correspond to two nearly-identical sentence embeddings in a 256-dimension space (which, in a gaussian setting, would occur with a probability of 0). As such, we decided to implement clamping, where our maximum cosine similarity (corresponding to a prediction of 5 in the context of STS) would be less than 1. After examining the MSE loss for examples of differing similarity as demonstrated in figure 4b, we found clamping to be especially useful for correctly identifying similar examples.



(a) Accuracy



(b) MSE Loss

Figure 4: STS Metrics across different nonlinearities

6.3 Greedy Soups

Our findings support much of the arguments that Wortsman et al. (2022) make in their justification for model soups. Whereas Wortsman et. al argues the competitive of model soups compared to model ensembles; our experiments demonstrate a unique advantage of model soups over ensembles: multitask performance. Our three tasks differ in both output format and input format; making ensembling techniques like mixture-of-experts routing impractical. Instead, model soups gives us the advantage of aggregate model performance across multiple tasks.

It is important to note, however, that we use greedy soups to combine models trained in a multitask setting. Models trained in a single-task setting lie in different loss landscapes, which means that averaging them yields generally meaningless results, as we'll elaborate further on in the next section on Fisher-weighted averaging.

In the context of aggregating model performance, greedy soups yielded the best average performance across all three tasks, indicating that greedy soups found a more optimal point in the "error basin" of all three tasks than gradient-based learning. When merging, we merged the three Pareto-optimal multitask models for each task — the three Gradient-surgery models listed in table 1. Through that, we see the biggest advantage of parameter averaging in a multitask setting: task balancing.

Gradient-based multitask learning inherently favors one task over another, even when aligning the gradients of all tasks in the same direction with gradient surgery (Yu et al., 2020). Javaloy and Valera (2022)'s Rotograd takes this a step further by homogenizing task gradients to the same magnitude. Through our experimentation, we found homogenizing task gradients to not work well; different tasks have different loss landscapes and learning progression.

As such, when experimenting with gradient surgery, we found the performance of tasks to vary across trials; certain trials may do relatively better in SST, whereas others do better in QDPD, and others in STS. However, we hypothesize that parameter averaging through greedy soups confers a method to find a more optimal point in the joint error basis across all three tasks.

We do not provide ablation studies for greedy soups as the algorithm itself only includes candidate models that increase dev split accuracy.

6.4 Fisher-weighted Averaging

We initially attempted Fisher-weighted averaging with three single-task trained models. However, our intial experiments showed extremely poorly performance; the resulting model performed the same as a random guessing for both QDPD and STS. In context, this makes sense. Averaging models produces a resulting model that lays between the error basins of its candidate models. Since all three tasks were trained on a single-task, each task's error basin was reached independent of the other tasks — meaning that the space in between their error basins would lack any significance in a multitask setting.

As such, we turned to Fisher-weighted averaging with multitask models. We merged the same three models as we did for our greedy soups model. In order to perform Fisher-weighted avraging over STS-B, Matena discretized the task, turning it into a classification task instead of regression. Our experiments show that this heavily degrades accuracy, and as such, we propose a method for performing Fisher-weighted averaging over regression tasks. Instead we approximated a candidate model's posterior distribution for the similarity head by applying a Gaussian Kernel with a mean set to the regression output.

Quantitatively, the Fisher-weighted resulting model performs worse than the Greedy soup model or the optimized Gradient-surgery models. We hypothesize that a Fisher-weighted averaging doesn't translate well in a multitask setting because of the uniqueness of posterior distributions across different tasks. Furthermore, the increased width of the input distribution in a multitask setting makes bootstrapping the Fisher-information matrix less effective than it would be in a single-task setting (even on top of having to approximate the STS probability distribution).

As such, in a multitask setting, we find greedy soups to be a superior method of parameter averaging compared to Fisher-weighted averaging. As demonstrated in the ablation study in table 2, Fisher-weighted averaging fails to match the performance of the best candidate model.

Table 2: Ablation study for Fisher-weighted Averaging

Experiment	SST	QDPD	STS	Avg
Gradient Surgery-base (Model 1)	0.473	0.727	0.796	0.665
Gradient Surgery w/ SMART (Model 2)	0.478	0.748	0.790	0.672
Gradient Surgery w/ SNLI (Model 3)	0.476	0.761	0.790	0.676
Fisher-weighted Merging w/ Model 1 and 2	0.489	0.745	0.780	0.671
Fisher-weighted Merging w/ Model 1 and 3	0.480	0.755	0.778	0.671
Fisher-weighted Merging w/ Model 2 and 3	0.470	0.753	0.788	0.670
Fisher-weighted Merging all three models	0.478	0.753	0.778	0.670

7 Conclusion

In this project, we explore the use of parameter averaging to increase performance in a multitask setting. We use our experiments to explore three research questions:

1. **Can we match single-task training performance in a multitask setting?**
   To a certain degree, yes. We demonstrate that our greedy-soups model matches 98.9% of the average performance of its best single-task counterparts.
2. **Can parameter averaging increase performance in a multitask setting?**
   We found that parameter averaging through "greedy soups" yielded the most optimal multi-task model in the joint error basin of all three tasks.
3. **Can Fisher-weighted averaging increase performance over other parameter averaging methods in a multitask setting?**
   Our experiments showed that Fisher-weighted averaging yielded worse performance than greedy soups. It is important to note that we bootstrap the diagonals of the Fischer information matrix and approximate the posterior distribution for STS. Our results may not be directly applicable to an experiment that directly computes the whole Fisher information matrix. Regardless, we still believe that our experiments serve as a fair demonstration of Fisher-weighted averaging as outlined in Matena and Raffel (2021) in a multitask setting.

In the process of answering those questions, we explore the intricacies of multitask learning and the tasks of sentiment classification, paraphrase detection, and semantic textual similarity. In the future, we want to experiment with more esoteric combinations of candidate models with greedy soups. Further work could also be done in evaluating parameter-averaged multitask models in the context of robustness to out of distribution data.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Adrián Javaloy and Isabel Valera. 2022. Rotograd: Gradient homogenization in multitask learning.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Michael Matena and Colin Raffel. 2021. Merging models with fisher-weighted averaging.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning.