OptimusBERT: Exploring BERT Transformer with Multi-Task Fine-Tuning, Gradient Surgery, and Adaptive Multiple Negative Rank Loss Learning Fine-Tuning

Stanford CS224N Default Project

Ryder Matheny
Department of Computer Science
Stanford University
mathenyr@stanford.edu

Shawn Charles
Department of Computer Science
Stanford University
hesterc@stanford.edu

Gabe Seir
Department of Computer Science
Stanford University
Gabe Seir@stanford.edu

Abstract

Natural language transformers offer an opportunity to explore the capabilities of machines in understanding human language. This paper explores the effectiveness of extending a standard minBERT transformer to be more effective in completing sentiment analysis, paraphrase detection, and semantic textual similarity tasks. The team contributes to the current knowledge based by exploring optimal methods for extending the standard minBERT model using a Multiple Negatives Rank Loss (MNRL) learning (a technique found in RNN models such as GMail Smart Reply), multi-task fine-tuning, and gradient surgery. The paper highlights that a generalizable model, both accurate in training and defensive against over-fitting, can be achieved through the use of these techniques, both in isolation and in conjunction. We found that our extended minBERT model obtained the best test results using a combination of multitask classifier, Cosine Similarity MNRL, multitask finetune, gradient surgery, and two hidden layers per classification head (with ReLU activation + dropout).

1 Introduction

An artificial intelligence model that is able to display a more intrinsic understanding of human language is more valuable than one that is only able to effectively complete one singular language task. Implementing a BERT model to perform effectively on a single task is thus not sufficient in practice. It fails to capture the full potential of natural language transformers in regards to their ability to perform well on multiple different language tasks more effectively than previous methods such as recurrent neural networks. The current baseline model, multitask classifier trained solely on STS data, proved insufficient at generalizing to the required tasks. We propose several techniques, either used in combination or in isolation, aim to improve about the generalizability and thus value of the transformer model. We identified several such permutations including multi-task fine-tuning with gradient surgery, multi-task fine-tuning in isolation, and a best test result with combination of multitask classifier, Cosine Similarity MNRL, multitask finetune, gradient surgery, and two hidden layers per classification head (with ReLU activation + dropout).

2 Related Work

In the paper "Efficient Natural Language Response Suggestion for Smart Reply," Henderson et al. (2017) explore several methodologies for recreating and improving Google's recurrent neural network Smart Reply system. The researchers in this experiment focus on altering the method for computing $P(y|x)$ (where x is a message and y is a suggested response, or computing the probability of a label given an input) by leveraging a feed-forward model in combination with a factorized dot product. Combining sampling with Multiple Negative Rank Loss Learning to compute the loss function was an additional proposed alteration, which seeks to increase the likelihood of correct response-input pairs by driving correct response and input embeddings closer together and driving incorrect response and input embeddings further away. This research group is attempting to solve the problem of computationally expensive RNNs and convolutions with an architecture built on utilizing n-gram embeddings through the method of multiple negatives and feed-forward neural networks. The major finding of this paper is that when multiple negatives are used in a multi-loss architecture, in comparison to Seq2Seq, it provided a conversion rate of 104% and a latency of 2%.

In the report "Gradient Surgery for Multi-Task Learning" by Yu et al., researchers propose and investigate the effectiveness of performing gradient surgery on conflicting gradients within multi-task learning structures. The proposed methodology, called projecting conflicting gradients (PCGrad) for resolving the issue of lost information as a result of tasks providing conflicting gradients is to project each gradient onto the normal plane gradient that it conflicts with (for each conflicting gradient). Furthermore, the method does not "introduce assumptions on the form of the model" (Yu et al.) because non-conflicting gradients are not adjusted. The major contribution of this piece is the proposal of PCGrad which is a model-agnostic way to compute more meaningful gradients in multi-task learning situations.

In the paper "BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning" Cooper and Murray (2019) propose of methodology of multi-task fine-tuning using a single BERT and few additional parameters. The team proposes using "projected attention layers" (PALs), or "[m]ulti-head attention, with shared V^E and V^D across layers (not tasks)" where "V^E is a $d_s x d_m$ 'encoder' matrix [and] V^D is a $d_m x d_s$ 'decoder' matrix with $d_s < d_m$"(Cooper and Murray (2019)). The contribution of this paper to the greater knowledge base is that PALs enables the viable use of a singularly fine-tuned BERT that requires a fraction of the parameters of multiple fine-tuned models.

3 Approach

The baseline infrastructure of the approach used to perform sentiment analysis, paraphrase detection, and semantic textual similarity tasks for the later described datasets is the base minBERT model. The model leverages multi-head self-attention as presented by Vaswani et al. (2023) in the computation: $\text{Attention}(Q, K, V) = \text{softmax}\frac{QK^T}{\sqrt{d_k}}V$, where Q, K, and V are matrices of queries, keys, and values, respectively, and d_k is the dimension of each query and key. The forward functionality of each layer in the minBERT transformer is heavily inspired by the transformer model architecture outlined by Vaswani et al. (2023) in their Figure (1). The research team manipulated the architecture of the provided skeleton code and implementations from parts pertaining to strictly sentiment analysis to make the baseline model functional for all three downstream tasks. To extend the model's functionality to paraphrase detection and semantic textual similarity, each of two input sentences were converted into their own embedding using the base BERT model; two distinct [CLS] token embeddings were then obtained; the token embeddings were concatenated together, dropout was applied, and then a linear layer of the result was returned.

The team implemented the momentum based Adam Optimizer as its baseline optimizer implementation according to pseudo-code crafted by Kingma and Ba (2017) with adjustments that accounted for the full set of parameters being provided (whereas the source pseudo-code assumed the parameters were learned one-at-a-time). Furthermore, in a design decision inspired by the project outline and the aforementioned pseudo-code, the team chose a more efficient implementation, replacing bias-corrected first and second moment estimation and parameter updating with

$$\alpha_t \leftarrow \frac{\alpha \cdot \sqrt{1-\beta_2^t}}{1-\beta_1^t} \qquad \theta_t \leftarrow \frac{\theta_{t-1} - \alpha \cdot m_t}{\sqrt{v_t} + \epsilon}$$

where α is the step size, β_1 and β_2 are exponential decay rates for the moment estimates, θ's are parameters at different times t, m is the first moment vector, and v is the second moment vector.

In order to extend the standard minBERT model, the team chose to implement more effective fine-tuning methodology including MNRL learning, multi-task fine-tuning, and gradient surgery and devised approaches to combat over-fitting.

The team implemented MNRL learning during the training stage of the finetuning process with the goal of improving the BERT embeddings for all three tasks. In this process, inspired by the approach presented by Henderson et al. (2017), the team minimize the distance between the embeddings of similar sentences, and maximize the distances from the rest of the sentence embeddings in a given batch. Instead of the baseline loss function, we use

$$J(\mathbf{x}, \mathbf{y}, \theta) = -\frac{1}{K}\sum_{i=1}^{K}[S(x_i, y_i) - \log\sum K_{j=1}^{K}\exp(S(x_i, y_j))]$$

where $S(x_i, y_i)$ is given by the Pearson correlation of our predicted similarity scores to the actual similarity scores of our training data.

Deciding what constitutes a "similar" sentence pair involved instituting a dynamic threshold hyper-parameter based on two different techniques of comparing embeddings similarity. The first of these techniques involved using the label similarity scores from the semantic textual similarity dataset (found within the dataset annotations). When this technique was used, the threshold score was initialized to 4.4 (where the maximum similarity score possible was 5) and increased by .05 each epoch. If a sentence pair met the threshold similarity, the embedding of each sentence was used within the appropriate MNRL loss function as the positive embedding example and each other sentence embedding in the batch was used as a negative to each of the distinct positive embeddings. This included creating pairs between sentences that were not paired in the dataset itself, and counting them as negative examples. The second technique first found the embeddings of each sentence in a given batch from pair the STS dataset, then computed the cosine similarity between all possible pairs. The threshold for cosine similarity comparison was initialized to .8 and increased by .02 every epoch. If the two embeddings met the threshold, then they constituted a positive pair and were used to compute the loss in the same way as the previous example. Cosine similarity offered a more mathematical approach to similarity in comparison to the human annotation scores. Increasing the threshold each time ensured that the fine-tuning adaptively became more precise as what constituted a similar sentence pair became stricter.

The team implemented multi-task fine-tuning to optimize the fine-tuning of the model on each task based on the training dataset that best coincided with the task. The team leveraged the iterative "round robin" technique described by Cooper and Murray (2019) where batches of training examples from each task were obtained separately (the team extracted relevant information from each sample). Despite a potential drawback of "round robin" being over-fitting small datasets and under-fitting larger datasets (because of more frequent iterations through the smaller datasets) (Cooper and Murray (2019)), the the team noticed that the baseline performed the worst on the semantic textual similarity task, which had a relatively small dataset in comparison to the others, so the team decided more attention to this dataset would be beneficial to overall general performance. Additionally, as will be discussed later, other methods were instituted to prevent negatively consequential over-fitting. After the data was sampled iteratively, three separate losses were computed using the appropriate information: cross entropy loss was used for the SST related data since the scores are categorical, binary cross entropy is used for paraphrase detection related data since the scores are binary, and Huber loss was used for semantic textual similarity data since the scores are continuous. Then, the losses were added and the final gradient was computed using the appropriate optimizer.

The last major architectural extension involved the use of gradient surgery within the multi-task fine-tuning methodology. Within the first layer of multi-task fine-tuning, each task shares parameters. The gradients as a result of the loss of the task thus all have an impact on the first layer of the neural network (and each task then separates into its own layer for subsequent gradient computations). Thus, if the gradients of two or more tasks don't agree with one another, indicated by a negative dot product between gradients, it could lead to a loss of information as they seek to cancel each other out; to remedy this, the team projected each gradient onto the normal plane of the other (when gradients were conflicting) and use these projections in a technique known as projected conflicting

gradient, or PCGrad (Yu et al.). This means that information is not lost in the first layer of the network. Though the team initially implemented gradient surgery by hand, a decision was made to leverage a preexisting library created by Cheng-Tseng which changed the optimizer from Adam Optimizer to PCGrad.

One of the identified threats to the effectiveness of the extended minBERT model was over-fitting. As a result, three steps were taken to combat this risk and thus improve performance on test sets. Two of which related to adjusting hyper-parameters and one was a more minor architectural development. Firstly, the team experimented with varying levels of dropout, specifically in experiments that related to multi-task fine-tuning. For example, multi-task fine-tuning training was first conducted using a dropout probability of .3; upon noticing high levels of over-fitting, dropout was increased to .43. Additionally, the weight decay values for the different permutations were adjusted based on perceived over-fitting. L2 regularization or weight decay was also added at a value of $1e-3$ to fight over-fitting. Apart from hyper-parameter adjustments, the team implemented a technique called homoscedastic weighting within the multi-task fine-tuning architecture. In order to address Aleatoric uncertainty ("uncertainty with respect to information our data cannot explain") that is homoscedastic ("task dependent") (Kendall et al.) and model over-fitting, the team leveraged a loss weighted technique drafted by (Kendall et al.) founded on:

$$L_1(W) = ||y_1 - f^W(x)||^2$$
$$L_2(W) = ||y_2 - f^W(x)||^2$$
$$L(W, \sigma_1, \sigma_2) = \frac{1}{2\sigma_1^2} L_1(W) + \frac{1}{\sigma_2^2} L_2(W) + \log(\sigma_1) + \log(_2),$$

where $f^W(x)$ is the output of a neural network with input x and weights W, y is the expected output, and σ is a variance parameter

The team created a parameter that required a gradient for each task representing the log variance; this parameter starts at 0 (so initially everything is weighted the same); after each task obtains a gradient, the uncertainty loss is calculated and the gradient of this loss is applied to the log variance parameter in the backward step. The losses are then weighted by the above computation. The computation "can be trivially extended to arbitrary combinations of discrete and continuous loss functions, allowing us to learn the relative weights of each loss in a principled and well-founded way" (Kendall et al.).

4 Experiments

4.1 Data

The research project leveraged the Stanford Sentiment Treebank (SST), CFIMDB, the Quora, and Semeval STS datasets, as provided and described in the default handout. SST is a corpus of single sentence movie reviews labelled with sentiment rated on a scale 0 to 4, with a higher value demonstrating a higher sentiment. The CFIMDB dataset consists of highly-polar movie reviews that are binarily labeled on sentiment: positive or negative. These datasets are utilized for finetuning and testing of the minBERT implementation for sentiment analysis. The Quora dataset includes pairs of questions with binary labels, indicating if the sentences are paraphrases of each other. The Semeval STS dataset consists of pairs of sentences that vary in similarity on a scale from 0 to 5. These datasets were respectively used for finetuning of the multitask-BERT model for paraphrase detection and semantic textual similarity. The dev sets were also utilized for evaluation of the multi-task BERT model. Additionally, in the final variation of the model, with multitask fine-tuning, the datasets were used to synchronously train the multi-task classifier on their respective tasks. In this way, each of the datasets were utilized within the same training loop to train their specific task.

4.2 Evaluation method

We evaluated our model using its performance for the three tasks (sentiment analysis, paraphrase detection, and semantic textual similarity) on data the model hadn't seen before. Specifically, for evaluation of sentiment analysis we used accuracy (percent of labels correctly predicted) as our quantitative metric for performance on dev and test sets. For evaluation on paraphrase detection we also used accuracy as our quantitative metric for performance on dev and test sets. For evaluation of semantic textual similarity we used Pearson correlation of our predicted similarity to the actual similarity as our quantitative metric for performance on dev and test sets.

4.3 Experimental details

The minBERT multitask classifier model was run on the training datasets with the following default parameters: 10 training epochs, a learning rate of $1e-5$, dropout of 0.3, weight decay of $1e-3$, a single linear classifier head for each task applied to the pooling output from BERT, and BERT embedding fine-tuning enabled. In some cases of experimentation, less epochs were used; these cases will be reported in depth in the results section.

The model was run with different configurations of extensions determined by setting specific flags as true or false. For example, we ran models with and without gradient surgery, and all models using extensions will be described in depth in the results section. Hyper-parameters, such as dropout, were also experimented with, and the specific details of which will be detailed in the results section.

4.4 Results

Throughout our project, the research team developed a myriad of extensions upon the multitask classifier. Thus, to get a better understanding of how each extension, and combinations of extensions, impacted the performance of the multitask classifier compared to the baseline each of these extensions, and combinations, were trained and evaluated against the dev set.

In the first line of Figure 1 the results of the part 1 minBERT implementation are presented, showing an overall dev accuracy of $.467$. With MNRL, where similar sentence embeddings were determined using a threshold with training data labels, the overall dev score actually decreased slightly but the STS score slightly improved. The increase in STS was expected, but the group did not expect a decrease in overall accuracy. When MNRL, where similar sentence embeddings were determined via cosine similarity instead, a 2% increase was observed as evident in the second row of Figure 1. This result was as expected as noted in the literature.

As multitask finetuning was implemented, significant improvement upon the baseline was found. As indicated in the third row of data, multitask fine-tune without gradient surgery created an accuracy increase of $.187$ compared to the original baseline. Overall, the team believes that the significant increase in performance is a condition of training on all datasets and tasks instead of training on a single and applying the model on all 3 tasks. When tested with multitask finetuning and gradient surgery, the model found the same accuracy in all categories. This was contrary to our initial beliefs: that multitask with gradient surgery would out-perform multitask without it. The research team speculates that this was a function of over-fitting to the training data and the domination of the paraphrase dataset. When over-fitting, the parameters are conditioned too heavily upon the training data leading to a lack of generalizability for the model. When this is the case, the model would be conditioned to patterns in the training data that could make them behave similarly, regardless of gradient surgery, on the dev set. Additionally, the team believes that the gradients for paraphrase could have dominated the other tasks. If the gradients were to dominate the others, gradient surgery would not have a large impact on the other tasks, and could lead to similar results.

The different variations of MNRL, threshold and cosine similarity, were then tested with multitask fine-tuning. When threshold MNRL was run with multitask finetuning, the models accuracy regressed slightly, in all categories, when compared to the overall accuracy of only multitask finetuning. Threshold cosine similarity performed the same overall, but showed a $.024$ increase, $.023$ decrease, and $.002$ decrease in SST, paraphrase and STS respectively. This was not as expected. The group believe that in conjunction MNRL and multitask fine-tuning would see an increase in STS and paraphrase and a stagnation or slight decrease in SST. This regression could be a result of task interference where optimizing STS with MNRL hurts the feature representation for the other tasks. MNRL maximizes the difference between positive and negative examples, which can strongly benefit some tasks, but in some cases, this could harm the parameters when considering another task.

The team then began to combat over-fitting and task domination of the model. The first adaptation to combat this was a multitask classifier with multitask finetuning, increased dropout probability, and homoscedastic loss weighting. This model was ran on 7 epochs, also to combat overfitting. The group hypothesized that homoscedastic weighting would help to limit both over-fitting and task domination as it uses adaptive, task-specific uncertainty parameters to regularize the losses of certain tasks, de-emphasizing dominant tasks. The model was run with a increased

hidden layer dropout rate of .4. During training, the model indicated less over-fitting, with lower train accuracies, but the data, found in Figure 3, did not indicate an increase in accuracy, as SST decreased by .033, paraphrase was the only task to increase by .008, and STS decreased by .013. This approach did not work as expected as paraphrase increased instead of being de-emphasized in the model. The homoscedastic weight may have inadvertently prioritized paraphrase, possibly because the task is binary and therefore easier or over time the model became more certain about paraphrase and consequentially emphasized its weight. Furthermore, we attempted one other "home run" models where we tried to implement our best possible one-shot models for a final attempt at training. Likewise ran on 7 epochs, we created a model with two additional hidden layers for each classification head. We applied dropout and a ReLU to each hidden layer before using a classification head for each task. Overall, this didn't significantly improve our performance like we hypothesized and performed similarly to multitask finetuning on the dev set.

When evaluated against the test set, the model with multitask finetuning and gradient surgery performed similarly to its performance against the dev set. This performance, an overall accuracy of .657 was better than expected by the team. The team believed that the indications of over-fitting during training would mean that the model would not be generalizable and could suffer a significant decrease in accuracy against the test set. Even though the dev set is not trained on, the team feared that the over-fitting to the training data alongside indirect learning on the dev set, such as hyper-parameter selection and model selection, could lead to lower performance on the test set. Overall though, the model was just as generalizable to the dev and test sets alike even if it was over-fitting on the train set. Lastly, we also ran our two homerun models on the test set, but both performed similarly to the model with just multitask finetuning and gradient surgery. The models did show some promise in mitigating over-fitting, as they both indicated increased generizability as they showed increase in accuracy from dev to test; contrary to the multitask finetune showing a decrease in accuracy.

Model configuration	**SST Dev Acc**	**Paraphrase Dev ACC**	**STS Dev ACC**	**Overall Dev ACC**
Baseline multitask classifier implementation	.132	.590	0.360	.467
Multitask classifier + Cosine Sim MNRL	.124	.605	0.379	.473
Multitask classifier + Threshold MNRL	.126	.567	0.370	.459
Multitask classifier + multitask finetune	.514	.774	.394	.662
Multitask classifier + multitask finetune + gradient surgery	.514	.774	.394	.662
Multitask classifier + Cosine Sim MNRL + multitask finetune + gradient surgery	.481	.793	0.369	.653
Multitask classifier + threshold MNRL + multitask finetune + gradient surgery	.505	.770	.367	.653

Figure 1: minBERT multitask classifier dev results.

MNRL configuration	**STS Dev ACC**
Baseline Threshold MNRL	0.370
Baseline Cosine Similarity MNRL	0.379

Figure 2: minBERT multitask classifier dev results.

Model configuration	SST Dev Acc	Paraphrase Dev ACC	STS Dev ACC	Overall Dev ACC
multitask classifier + multitask finetune + homoscedastic weighting + increased dropout (7 epochs)	.481	.782	.380	.651
multitask classifier + Cosine Sim MNRL + multitask finetune + gradient surgery + two hidden layers per classification head (with ReLU activation + dropout), (7 epochs)	.496	.794	.362	.657

Figure 3: minBERT "home run" models multitask classifier dev results.

Model configuration	SST Test Acc	Paraphrase Test ACC	STS Test ACC	Overall Test ACC
minBERT with multitask finetune and gradient surgery	.528	.774	.338	.657
multitask classifier + multitask finetune + homoscedastic weighting + gradient surgery + increased dropout	.520	.784	.334	.657
multitask classifier + Cosine Sim MNRL + multitask finetune + gradient surgery + two hidden layers per classification head (with ReLU activation + dropout)	.514	.792	.341	.659

Figure 4: multitask classifier test results.

5 Analysis

Our model uses a multitask finetuning framework to create a model optimized to complete sentiment analysis, paraphrase detection, and semantic textual similarity. Qualitatively, we expected this to increase the performance of our model as it's learning aspects from each task that we're evaluating on, and not just a single task. This means that each linear classifier is being fine-tuned to its respective task and the minBERT embeddings are getting specialized for these three tasks in particular.

The addition of MNRL to the model helped the model to understand the relation between text and determine similarity between inputs. As indicated by the increase in STS accuracy, and the increase in paraphrase accuracy for cosine similarity, MNRL helped build the models understanding of language nuance and semantic relationships. On the other hand, sentiment analysis accuracy decreased, meaning that MNRL may have biased training towards paraphrase and STS. This shows the need for careful consideration of additional extensions that are utilized alongside MNRL.

Gradient surgery, while hypothesized to help our model's accuracy, did not improve the capabilities of our model. This is because our model was already overfitting to the training data, and thus improving the gradients the model used to train would mean that the model would just overfit more and not actually improve its performance overall.

An interesting nuance to the model is revealed when the test versus dev accuracies are analyzed. The model that included homoscedastic weighting and increased dropout showed improvement when comparing test and dev accuracy. This shows a positive indication that, despite concerns of over-fitting, the countermeasures were successful in mitigating some of the effects. The models better performance on unseen test data indicates that the model maintained generalization, and could

indicate that the weighting and dropout alterations were successful in fighting over-fitting. Likewise the multitask classifier with two hidden layers additionally showed generizability, with an increase between dev and test accuracy, showing promise in the approach to mitigating over-fitting.

6 Conclusion

Overall, we developed a minBERT embeddings based model for the multitask classification of sentiment classification, paraphrase detection, and semantic textual analysis. As a baseline, we used the pre-trained minBERT embeddings and used linear classification heads for each task. We found that training on a single task gave us decent performance for that task, and poor performance on the other two. We then implemented extensions such as multitask fine-tuning to improve each linear classification head, Multiple Negatives Rank Loss Learning on similar sentences to improve our sentence embeddings, and gradient surgery to improve loss gradient calculation when training for all three tasks. We found that multitask fine-tuning had the largest overall positive impact on our results, and that Multiple Negatives Rank Loss Learning and gradient surgery afforded little improvement over just multitask fine-tuning. However, we recognize that our biggest limitation was that our models were severely over-fitting the train data (scoring nearly 100% accuracy on all 3 tasks) meaning that perhaps these improvements didn't have the room to show greater efficacy.

Lastly, we attempted to create "home run" models where we tried to implement our best one-shot models for a final attempt at training. We found that increasing dropout and adding homoscedastic weighting didn't improve our model or the overfitting problem, neither did increasing the weight decay, and adding additional hidden layers made a small improvement to the overall capabilities of the model.

In the future, we'd like to continue to combat our model's tendency to overfit as this was the biggest limitation in our work. Possible avenues to address this limitation include experimenting with different loss weight schemes, different methods of bath sampling to limit task bias or domination, and continued hyper-parameter experimentation.

References

Wei Cheng-Tseng. Pytorch-pcgrad. https://github.com/WeiChengTseng/Pytorch-PCGrad. Online; Accessed 13 March 2024.

Asa Cooper and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. Technical report, International Conference on Machine Learning.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. Technical report, Google.

Alex Kendall, Gal Yarin, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. Technical report.

Diederik Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. Technical report, International Conference on Learning Representations.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomex, Łukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. Technical report, Google.

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