SMARTer BERT

Stanford CS224N Default Project

Naijing Guo
`njguo@stanford.edu`

Alexey Tuzikov
`atuzikov@stanford.edu`

Tatiana Veremeenko
`tataver@stanford.edu`

Abstract

Fine-tuning large language models is an effective way to achieve better performance on downstream tasks by leveraging language representations learned during pre-training. In this project we apply multi-task learning to fine-tuning process involving three different downstream tasks: Sentiment Analysis, Paraphrase Detection and Semantic Textual Similarity. We consider different learning methods, regularization techniques and hyperparameter choices in order to identify core principles and best approaches to achieve fast and effective training. We found that some of the commonly used regularization techniques, such as use of dropout layers, may not always lead to desired results, while simple dataset balancing may help achieve a good balance between training time and performance.

1 Key Information to include

- Mentor: Yuan Gao

2 Introduction

The advent of large language models (LLMs) has marked a significant milestone in the field of Natural Language Processing (NLP), bringing forth unprecedented capabilities in understanding and generating human language. These models, trained on vast corpora, learn intricate patterns and nuances of language, enabling them to perform a wide array of tasks with remarkable accuracy Ding et al. (2023). However, the quest for optimized performance on specific downstream tasks necessitates fine-tuning these pre-trained models. Fine-tuning, the process of adjusting a model's parameters to better suit a particular task, is a delicate and nuanced endeavor, demanding an in-depth understanding of both the model's architecture and the characteristics of the task at hand Hu et al. (2021). This project delves into the fine-tuning of a minimalist version of BERT Devlin et al. (2018), known as minBERT, exploring effective techniques and identifying crucial elements for achieving successful adaptation to downstream NLP tasks such as Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity.

The challenge of fine-tuning lies not only in the technical complexity of modifying a large language model but also in maintaining the delicate balance between leveraging the model's pre-trained knowledge and adapting it to new, task-specific objectives. The difficulty is further compounded by the need to prevent overfitting, ensure efficient training, and manage the computational demands of processing large datasets Ding et al. (2023). These challenges make the fine-tuning of LLMs a compelling area of research. Current methods in fine-tuning LLMs vary in their approach, with some focusing on the adjustment of learning rates, others on the modification of the model architecture, and yet others on the introduction of novel training routines or loss functions. Despite the successes achieved by these methods, they often fall short in addressing the multifaceted challenges of fine-tuning, such as effectively combating overfitting, optimizing training efficiency, and achieving high performance across multiple tasks. This project aims to build upon the existing body of work, proposing a comprehensive fine-tuning strategy that leverages a multi-task training routine and various regularization techniques to address these challenges.

3 Related Work

Efforts to enhance multi-task learning have been significant, focusing on the concept that models can improve their performance on a primary task by learning from related tasks Ruder (2017). This approach hinges on the idea that by sharing knowledge between tasks, a model can develop more robust generalizations. Multi-task learning (MTL) bolsters generalization capabilities by drawing on the specific insights gained from the training data of tasks that are interconnected Caruana (1998). This not only aids in excelling at current tasks but also paves the way for the model to adeptly handle new challenges and domains.

In the realm of multi-task learning, methodologies typically fall into one of two buckets: 'hard parameter sharing' or 'soft parameter sharing'. With hard parameter sharing, the strategy is somewhat akin to a communal workspace where the core components (hidden layers) are shared across all tasks, while allowing each task to have its own personalized finishing touches (task-specific output layers). This can be imagined as different departments in a company sharing the same office space but having their own desks. On the flip side, soft parameter sharing is like giving each task its own office (model), with efforts made to keep the decor (parameters) of these offices similar through a process akin to regular check-ins or guidelines, aiming to foster similarity among them Duong et al. (2015) Yang and Hospedales (2017).

In the field of NLP, prior research has demonstrated various methodologies for enhancing task performance, such as incrementally deepening layer sharing across tasks with each training iteration Lu et al. (2016); integrating cross-stitch units to enable models to autonomously decide how networks for specific tasks can utilize the insights from another task through a linear combination of outputs from earlier layers Misra et al. (2016); employing a singular model to execute chunking, tagging, named entity recognition, and semantic role labeling by applying a communal neural network across texts while using distinct output layers for each task Collobert et al. (2011), and exploiting linguistic hierarchies Søgaard and Goldberg (2016), Hashimoto et al. (2017).

A burgeoning direction in transfer learning involves initially pre-training a model architecture to learn from a language modeling objective, subsequently fine-tuning this model for a specific supervised task Dai and Le (2015) Howard and Ruder (2018). In this project, we utilize the BERT architecture Devlin et al. (2018), which underwent pre-training with dual objectives: to fill in "masked" words from a given sentence, and to ascertain whether two sentences are sequentially placed in a text corpus.

4 Approach

4.1 Model Architecture

Our model architecture is based on a minimal BERT Devlin et al. (2018) and ADAMW optimizer that we implemented by completing the provided skeleton code. The pooled representation of the [CLS] token from the last BERT layer output is then used as a sentence representation for downstream classification tasks. For tasks requiring single prediction for a pair of sentences, we first concatenate the sentences, using the special [SEP] token. To make the resulting model able to return predictions for each downstream task, we added three independent linear layers, taking BERT sentence representations as inputs. We follow outputs of these linear layers with appropriate normalizations (softmax for sentiment analysis, sigmoid for paraphrase detection) and loss functions (cross entropy for sentiment analysis, binary cross entropy for paraphrase detection and MSE for semantic textual similarity). These heads are trained separately on the corresponding data sets.

4.2 Multi-task learning and baseline

Soft-parameter sharing for multi-tasking with BERT requires a relatively large number of parameters Stickland and Murray (2019). Therefore, we explore how to do hard parameter sharing. The training procedure consists of two stages: pretraining and multi-task learning. During the pretraining stage the parameters of the lexicon encoder and Transformer encoder are learned using two unsupervised prediction tasks: masked language modeling and next sentence prediction. Possible improvements at this stage are beyond the scope of this work. We start our research with pretrained weights available online.

At the multi-task learning stage, we use minibatch based stochastic gradient descent (SGD) to learn the parameters of the model (i.e., the parameters of all shared layers and task-specific layers). For our **baseline**, we use the following algorithm: for each epoch, for each dataset, for each batch within a dataset, we calculate the loss and update all parameters of the layers related to the given task, as well as the shared parameters. This algorithm results in very slow training and significant overfitting on the paraphrase detection task. To address these challenges, we evaluate other approaches to data splitting: equal data splitting and task-proportional data splitting, based on the hypothesis that data imbalance may bias the learning towards the largest dataset. Balancing datasets per each epoch may help achieve more even gradient descent, while fully utilizing richness of large Quora dataset.

4.3 SMART Loss

To address overfitting and improve the model's robustness and efficiency, we implement and use SMART loss Jiang et al. (2019), a technique that introduces a principled regularized optimization approach. This technique is based on imposing a smoothness-inducing adversarial regularizer on the model during fine-tuning. Mathematically, the regularization component can be expressed as follows:

$$R(\theta) = \frac{1}{2} \max_{\|\epsilon\|_2 \leq \rho} L(\theta + \epsilon, \phi) - L(\theta, \phi)$$

Here, $R(\theta)$ represents the regularization term for the model parameters θ. L is the loss function; $f(\theta)$ denotes the model with parameters θ; x and y are input-output pairs, and σ is the adversarial perturbation bounded by ϵ, ensuring the perturbation is small and the model's output remains smooth within the neighborhood of x. SMART fine-tuning is expected to significantly improve our model's robustness and efficiency, leveraging the potential to enhance generalization by minimizing sharp minima in the loss landscape. This approach is particularly promising for achieving better performance metrics in sentence-level NLP tasks, and we anticipate it will provide valuable insights into optimization strategies that benefit model training and evaluation.

In our implementation, we used a simpler version of this SMART loss (Algorithm 1), focusing on the intuition behind making the model outputs robust to small perturbations as introduced in the paper, without the full complexity of adversarial training. The simplified SMART loss implementation adds Gaussian noise to the embedding of input tokens before passing them through the model. This process is designed to mimic small perturbations in the input space, encouraging the model to learn more generalized and robust features. The regularization term is computed as the mean squared difference between the logits (outputs before applying softmax) of the original and perturbed inputs. This term is then added to the original loss function, combining the task-specific loss with the regularization loss. This simplified version avoids the computationally intensive process of adversarial perturbation generation, which involves iteratively optimizing for perturbations that maximize the model's loss. This makes the training process more efficient and accessible, especially for our purpose with limited computational resources. Moreover, even though it's a simplified approach, adding noise to the embedding and penalizing large differences between the outputs of perturbed and unperturbed inputs can still significantly improve model robustness. This method still encourages the model to learn representations that are not overly sensitive to small variations, enhancing generalization.

5 Experiments

5.1 Data

For Sentiment Analysis task we use Stanford Sentiment Treebank dataset (11,855 single sentences from movie reviews, labeled as negative, somewhat negative, neutral, somewhat positive, or positive). For Paraphrase Detection we use a subset of Quora Question Pairs Dataset (202,152 question pairs, labeled as paraphrases or not). For semantic textual similarity we use SemEval STS Benchmark Dataset (8,628 different sentence pairs, labeled on a scale from 0 (unrelated) to 5 (equivalent meaning).

5.2 Evaluation method

To evaluate model performance, we use accuracy for sentiment analysis and paraphrase detection, and Pearson correlation for semantic textual analysis, as well as mean value of all three metrics to

Algorithm 1 Simplified SMART Regularization for Fine-Tuning Pre-trained Models

```
1: Input: Train dataset $D$, Model $M$ with parameters $\theta$, Noise standard deviation $\sigma$, Regularization
   weight $\lambda$, Learning rate $\alpha$
2: Output: Fine-tuned Model $M$
3: for each epoch do
4:     for each batch $(x, y)$ in $D$ do
5:         Compute embeddings $E = M.\text{Embedding}(x)$
6:         $E_{\text{perturbed}} \leftarrow \text{ApplyPerturbations}(E, \sigma)$
7:         $\text{logits} \leftarrow M.\text{ForwardFromEmbeddings}(E)$
8:         $\text{logits}_{\text{perturbed}} \leftarrow M.\text{ForwardFromEmbeddings}(E_{\text{perturbed}})$
9:         $\text{loss}_{\text{original}} \leftarrow \text{CrossEntropy}(\text{logits}, y)$
10:        $\text{loss}_{\text{reg}} \leftarrow \lambda \cdot \text{Mean}((\text{logits} - \text{logits}_{\text{perturbed}})^2)$
11:        $\text{total\_loss} \leftarrow \text{loss}_{\text{original}} + \text{loss}_{\text{reg}}$
12:        Perform backpropagation and update $\theta$ using $\alpha$ and total_loss
13:    end for
14: end for
15: function APPLYPERTURBATIONS$(E, \sigma)$
16:    $E_{\text{perturbed}} \leftarrow E + \sigma \cdot \text{RandomGaussianNoise}(\text{shape}(E))$
17:    return $E_{\text{perturbed}}$
18: end function
```

track performance across the tasks. To measure effects of our improvements, we use metrics of our baseline models on the same datasets.

5.3 Experimental details and results

5.3.1 Training hyperparameters

We run all our experiments with learning rate $1e-5$, batch size 8 for a variable number of epochs (10, 20 or 40).

5.3.2 Data splitting

- Equal data splitting (**split1**): To achieve approximately equal number of gradient steps per task per epoch, we split the Quora dataset randomly into 17 parts and use one part per epoch in succession, bringing the number of training examples per epoch to 8544/8328/6040 for sentiment/paraphrase/similarity tasks accordingly.
- Task-specific proportional data splitting (**split2**): In addition to splitting the Quora dataset, we also tried to split the STS dataset into 17 parts to approach training similar tasks (paraphrase detection and semantic textual similarity) more uniformly and prevent overfitting on the small STS dataset. This change reduced the number of STS training examples per epoch and gave us an 8544/8328/355 ratio for sentiment/paraphrase/similarity tasks.

We found that both approaches reduce the training time by approximately 4 times (from 20 min/epoch to 5 min/epoch), while allowing to achieve better results at the same time. We chose to use **split1** as the baseline for following experiments, since it demonstrates the best balance between performance and time efficiency.

	SST acc		Quora acc		STS corr	
	10 epochs	40 epochs	10 epochs	40 epochs	10 epochs	40 epochs
baseline	0.465		0.883		0.879	
split1	**0.530**	**0.519**	**0.854**	**0.883**	**0.874**	**0.88**
split2	0.505	0.51	0.844	0.886	0.863	0.87

Table 1: Data splitting strategies on dev dataset.

5.3.3 Dropout

We tried four values for the dropout layer applied on top of the BERT pooled output before each of three top task-specific heads: $0, 0.1, 0.2, 0.3$. We found that while for shorter training duration using dropout of 0.1 slightly improves the mean model metrics, it does not help in the long run (40+ epochs)

Dropout value	Mean dev score after 20 epochs	Mean dev score after 40 epochs
0 (baseline)	0.772	0.780
0.1	0.778	0.777
0.2	0.775	
0.3	0.773	

Table 2: Effects of applying dropout

5.3.4 SMART Loss

For our SMART loss experiment, we used the following hyperparameters values: $\lambda = 10$ and $\sigma = 0.05$, where λ is the weight of the regularization term in the loss function formula and σ represents standard deviation of perturbations applied to sentence embeddings. The highest achievable scores on the dev dataset turned out to be very close to the baseline. However, there's a noticeable regularization effect on training metrics across tasks that for some reason does not translate into better performance on dev dataset for the sentiment task (Figure 1).

	Train mean score	Dev mean score
split1	0.962	0.761
split1+smart	0.958	0.759

Table 3: Effect SMART loss on mean scores

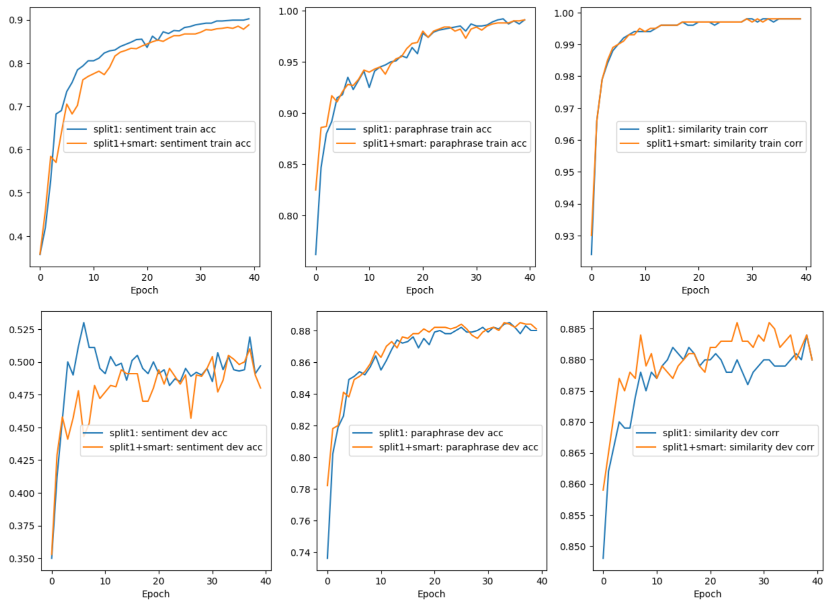

Figure 1: Effect of SMART loss on learning curves

5.3.5 Test leaderboard results

While some of the regularization techniques we tried looked promising in combating overfitting in the short term, we found that for longer training times, the model using balanced splitting strategy without additional extensions was able to achieve the best performance on the dev dataset. Our final score on the test leaderboard is 0.771. This shows that in our case data splitting approach is the most beneficial yet simple, while regularization seems to be hampering its effects in the long run.

Model	SST acc	Paraphrase acc	STS correlation	Mean score
split	0.487	0.883	0.882	0.771

Table 4: Test dataset performance

Model	SST acc	Paraphrase acc	STS correlation	Mean score
split	**0.519**	**0.883**	**0.88**	**0.781**
split+smart	0.51	0.884	0.882	0.778
split+dropout	0.519	0.875	0.876	0.777

Table 5: Dev dataset performance

6 Analysis

6.1 Sentiment error analysis

Sentiment analysis task proves to be the hardest to achieve high model performance on. The reasons why multi-task learning might not be working for it as well as for the other two tasks may include principal differences in their structure, limiting potential knowledge transfer, as well as amount of information that can be extracted from the provided sentiment dataset. The confusion matrix of our best model (Figure 2) shows that making the right call for extremes, as well as neutrals is very challenging for a model. Additional pretraining on sentiment data, or additional features may have helped model to combat underfitting on the dev and test datasets.

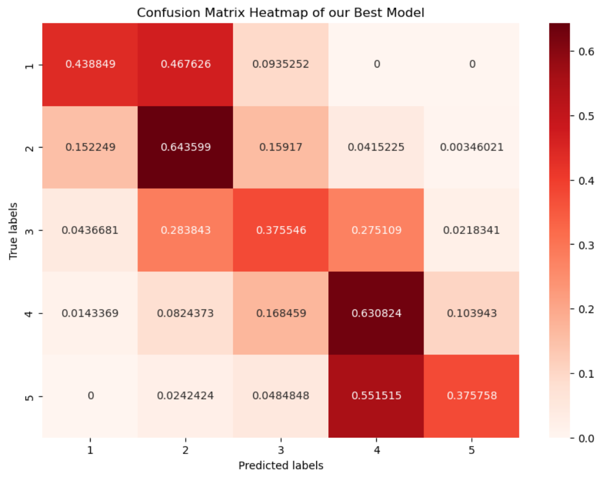

Figure 2: Confusion matrix heatmap of our best model

Analysis of data examples helps explain the heatmap. In particular, it is easy to mistakenly assign a review with the highest rating 4 a score of 3. Some reviews from the SST dataset with ground truth label 4 that could be labeled 3 by either a human labeler or a model:

- A compelling motion picture that illustrates an American tragedy
- Dramas like this make it human
- “Red Dragon” is entertaining
- Nicole Kidman makes it a party worth attending
- A taut, sobering film

At the same time, it's hard to mistakenly assign 4 to a description with label 3 because probablity of an extreme score given a strong word is quite high. For example, there is not a single review labeled 3 that has words "awesome" or "fantastic", and only 6 out of 2322 reviews labeled 3 contain the word "great".

The same logic is applicable to the other extreme - the reviews that have ground truth label of 0 but could easily be rated 1 by either a human labeler or a model:

- Unfortunately, it's also not very good
- But the second half of the movie really goes downhill
- There 's no energy
- For dance completists only
- ... overly melodramatic ...

At the same time, words that usually describe the lowest score of 0 represent such a strong feature that both human reviewer and the model are unlikely to make a mistake of assigning a review with strong words a score of 1 instead of 0. For example, out of 2218 reviews with ground truth label 1, the word "horrible" is used only 2 times, "awful" 8 times, and not a single use of phrase "very bad".

6.2 SMART loss

While SMART loss is developed to improve robustness and efficiency for fine-tuning natural language models, we did not find it to be helping in our case, especially for the sentiment analysis task. We hypothesize that it might be related to the fact that loss function of the sentiment analysis task may follow a shape very different from the other two tasks. A more careful choice of hyperparameters could've been beneficial.

6.3 Gradient conflicts

The data splitting strategies we applied allowed us to achieve faster and more balanced training across tasks. We started with an assumption that full training on the Quora dataset may drown effect of training on the other tasks because of its size. This hypothesis turned out to be only partially true, because it wasn't the case for the Semantic Textual Similarity. For Sentiment Analysis, however, our assumption proved to be correct: there's a strong inverse relationship between number of gradient steps we take for similarity tasks per epoch and the rate of training loss for the sentiment task(Figure 3). We believe this might be a strong indication that gradient-based multi-task learning methods like gradient surgery Yu et al. (2020) or gradient vaccine Wang et al. (2020) might be beneficial for improving multi-task learning in this case.

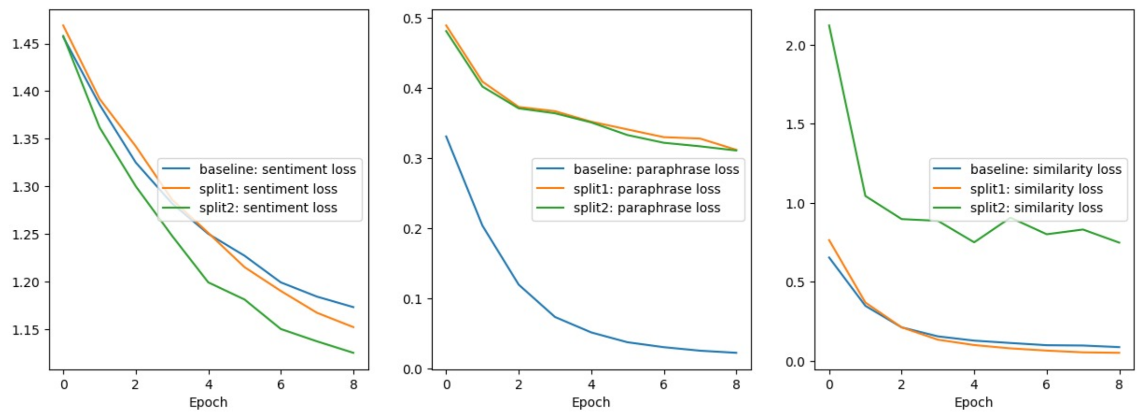

Figure 3: Effect of data splitting strategies on loss curves

7 Conclusion

We learned that splitting data strategies to balance number of gradient steps is a simple, but highly effective technique to achieve steady learning and shortened training time, that can also provide a valuable insight into the nature of the problem at hand and reveal hidden relationships in the gradient space. These differences in loss function landscapes are likely to be responsible for the fact that effects of regularization techniques we tried cancel out when applied generally, without task-specific hyperparameter tuning. Sentiment Analysis task proved to be the most challenging among the three and might require special treatment to achieve better performance. Some techniques that may help the model to capture and extract sentiment information from the data may include, but are not limited to additional pretraining, ensembling and additional feature engineering.

References

Rich Caruana. 1998. Multitask learning. *Autonomous Agents and Multi-Agent Systems*, 27(1):95–133.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Yuchong Qin, Guangxiang Yang, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5:220–235.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 845–850.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model finetuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Edward J. Hu, Sebastian Ruder, Aditya Siddhant, Michael Koren, Dani Yogatama, Ryan Cotterell, and Michal Łukasik. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Haoming Jiang, Danqing Xu, Jun Araki, and Graham Neubig. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Yu Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 2016. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages —. PMLR.

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

Yu Yang and Timothy M. Hospedales. 2017. Trace norm regularised deep multi-task learning. In *Workshop track - ICLR 2017*.

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