StudentBERT: Multitask Training with Teacher Models

Stanford CS224N Default Project

Shang Jiang
Department of Computer Science
Stanford University
`zcyy2022@stanford.edu`

Shengtong Zhang
Department of Mathematics
Stanford University
`stzh1555@stanford.edu`

Abstract

In this project, we explore a variety of techniques that significantly improve the performance of minBERT — a pretrained language model — on three downstream tasks: sentiment analysis, paraphrase detection, and sentence similarity. We find that applying Cross-Encoding on sentence inputs, Round Robin as training schedule, SIAR as regularization and Model Merging as overall training approach yields the biggest improvement on the model's performance. We achieve a score of 0.79 on the test dataset without ensembling, and hold 14th on the test leaderboard at the time of writing.

1 Key Information to include

- Mentor: Olivia Lee
- External Collaborators (if you have any): N/A
- Sharing project: N/A
- Team contributions: The team members contributed equally to the project code and report.

2 Introduction

In recent years, large language models (LLMs) like BERT and ChatGPT has found numerous applications. Since it is prohibitively expensive to train an LLM from scratch, most applications take an off-the-shelf pre-trained LLM and finetune it on specific downstream tasks. To reduce the cost of finetuning and model inference, we often wish to finetune a single LLM to perform multiple tasks simultaneously. Data scarcity is common for domain-specific applications.

In this CS224N default project, we finetune a pre-trained minBERT model on three downstream tasks: sentence sentiment classification (SST), paraphrase detection (PARA), and sentence similarity (SEN). Our main challenges include

1. PARA and SEN are sentence pair tasks, while minBERT models are designed to take only one input sentence. Finding the best way to deal with this discrepancy is a non-trivial task.
2. SST and SEN suffer from data-scarcity, with less than 10,000 training samples for each of them. Aggressive fine-tuning for these two tasks can easily lead to over-fitting and poor generalization Jiang et al. (2020).
3. While PARA and SEN are similar tasks, SST is very different from them. It is a notoriously difficult optimization problem Yu et al. (2020) to learn a shared embedding that performs well for different tasks.

We experiment with several approaches for dealing with these challenges. We encode the sentence pair tasks using *cross-encoding (CE)*, merging the two sentences into one before passing it to the model. We use *Smoothness-Inducing Adversarial Regularization*(SIAR) Jiang et al. (2020) to increase model robustness and mitigate overfitting. We employ a *Round Robin (RR) Training Schedule* to train over multiple tasks simultaneously. Furthermore, we develop a *Model Merging(MM)* approach, inspired by knowledge distillation Hinton et al. (2015), to reduce tradeoff and train a single minBERT embedding with high performance across multiple tasks.

3 Related Work

Sentence Pair Tasks Cross-encoding is the same approach used in the original BERT paper Devlin et al. (2019). A parallel approach is bi-encoding, pioneered by Reimers and Gurevych (2019) for pre-trained LLMs. In their approach, the two sentences are separately passed into BERT models with tied weights to produce embeddings u and v. They are then passed to either a cosine similarity function or a Siamese network to produce the final similarity score. Bi-encoding allows exceptional performance for *pair-regression tasks*, where the objective is to find two most similar sentences among a pool of n sentences. However, our experiments show that bi-encoding performs poorly for our specific PARA and SEN tasks, so we stick with cross-encoding.

SIAR The adversarial regularization technique lies at the intersection of two active fields of machine learning: preventing overfitting and defending against adversarial attacks. Many alternative approaches have been developed to prevent overfitting in data-sparse finetuning. For example, Howard and Ruder (2018) propose learning rate tuning tricks including *discriminative fine-tuning* (where different layers are tuned with different learning rates) and *slanted triangular learning rates* (where learning rates first increases then decreases). Another major development is *Low-Rank Adaptation(LoRA)* Hu et al. (2022), where the parameters of each layer are restricted to a low rank perturbation.

There is a large body of works on how adversaries can manipulate the input of large language models to create malicious output. For example, just last year Zou et al. (2023) show attacks that inject objectionable contents into commercial LLMs including ChatGPT, Bard and Claude. The authors of SIAR Jiang et al. (2020) test SIAR-trained models on an adversarial dataset ANLI Nie et al. (2020) and report improvement over the default BERT model. We point out that more effective approaches such as explanation prompting Kavumba et al. (2023) have been developed.

Training Multiple Tasks Perhaps the most straightforward way to maximize model performance across multiple tasks is the *sum-of-loss* approach, where the optimization objective is simply the sum of the loss function of each individual task. Our baseline model applies a direct gradient descent on this optimization objective. Yu et al. (2020) introduces *gradient surgery* as a better algorithm to optimize the sum-of-loss objective, though unfortunately we do not have time to attempt this approach.

We cannot find a published work that analyses our round-robin training approach. Teams from previous years Le and Lin (2024); Zhang and Duan (2024) have found this approach to be useful.

Model Merging Model merging is a relatively new concept in the study of LLMs Goddard et al. (2024). In our case, we are interested in merging multiple large language models with identical architecture and initialization that have been fine-tuned to different tasks (Section 2.2.1 of Goddard et al. (2024)). Our approach, which regularizes the student model by the KL divergence with a teacher models, is pioneered by Kim and Rush (2016) in the knowledge distillation(KD) literature for neural machine translation. This approach is used by Khanuja et al. (2021) to merge multiple monolingual LLMs into a single multilingual LLM. Recently, Gu et al. (2024) adapts Kim and Rush (2016)'s work to LLMs by replacing the KL divergence with the reverse KL divergence. Their work focuses on training a smaller language model with a larger language model, and they do not mention model merging in their work. We believe that our application of Model Merging to the default project is an original idea.

Other approaches to model merging are based on weight averaging, such as linear averaging of weights and Drop And REscale (DARE) Yu et al. (2023).

4 Approach

Our project uses the default minBERT model which follows the architecture in the original BERT paper Devlin et al. (2019): it consists of a word embedding layer and 12 BERT encoder layers. Each BERT encoder layer has a multiheaded attention mechanism, an additive & normalization layer with a residual connection, a feed-forward layer, and another additive & normalization layer with a residual connection. This model is denoted as Bert$(\cdot)$. It outputs a $D = 768$ dimensional embedding vector. To finetune the minBERT model, we add on top of the base minBERT model three attached heads $H_{\mathsf{SST}}, H_{\mathsf{PARA}}, H_{\mathsf{SEN}}$ for the three downstream tasks.

We do not consider model ensembling and voting, where multiple minBERT models are used to handle different tasks. Indeed, such approaches would double the training and inference costs in practice. Instead, we focus on exploring how a single minBERT model can generate good embeddings for the three tasks simultaneously.

4.1 Baseline Model

We want to set a relatively high bar for our baseline so that we can explore more advanced techniques for further improvement. For our baseline model, we use cross-encoding and sum-of-loss. We next provide a complete description of our baseline architecture, loss function and training method.

The SST tasks is a single sentence sentiment classification task with $C = 5$ classes, so our choice of H_{SST} is a $D \times C$ dense linear layer. Given input sentence s, our model outputs logits

$$\hat{y}_{\mathsf{SST}}(x) = H_{\mathsf{SST}}(\text{Bert}(x)).$$

The PARA and SEN tasks are sentence pair tasks with a single output logit. The heads H_{PARA} and H_{SEN} are both two-layer dense linear networks with a hidden layer of size 32 and a single output. We further use a ReLU function to ensure that the output of SEN is non-negative. For input sentences x_1, x_2, our model outputs the logits

$$\hat{y}_{\mathsf{PARA}}(x_1, x_2) = H_{\mathsf{PARA}}(\text{Bert}(x_1 + \text{[SEP]} + x_2)),$$
$$\hat{y}_{\mathsf{SEN}}(x_1, x_2) = \mathsf{ReLU}(H_{\mathsf{SEN}}(\text{Bert}(x_1 + \text{[SEP]} + x_2))).$$

For each task $s \in \{\mathsf{SST}, \mathsf{PARA}, \mathsf{SEN}\}$, our ground-truth (gt) loss function can be described as

$$\mathcal{L}_{s,gt} = \frac{1}{n}\sum_{i=1}^{n} \ell_{s,gt}(y_i, \hat{y}_s(x_i))$$

where $\{(x_i, y_i)\}_{i=1}^{n}$ are the labeled training samples. As SST and PARA are classification tasks, we choose $\ell_{s,gt}$ to be the Cross-Entropy loss for SST and Binary Cross-Entropy(BCE) loss for PARA. For SEN, given that the evaluation metric is the correlation coefficient which captures linearity, we use the Mean-Squared Error(MSE) loss function as $\ell_{s,gt}$. In the baseline we simply take $\mathcal{L}_s = \mathcal{L}_{s,gt}$ as the loss function for each task.

Finally, we train our baseline model by performing gradient descent on the sum of three losses

$$\mathcal{L} = \mathcal{L}_{\mathsf{SST}} + \mathcal{L}_{\mathsf{PARA}} + \mathcal{L}_{\mathsf{SEN}}.$$

4.2 SMART Regularization

To mitigate overfitting, we implement Smoothness-Inducing Adversarial Regularization (SIAR) following Jiang et al. (2020). For the readers' convenience, we restate their algorithm.

For each task s, SIAR adds a "smoothness-inducing adversarial regularizer" $\mathcal{L}_{s,siar}$ to the ground-truth loss function $\mathcal{L}_{s,gt}$

$$\mathcal{L}_s = \mathcal{L}_{s,gt} + \lambda_s \mathcal{L}_{s,siar}$$

where λ_s is a tunable parameters. $\mathcal{L}_{s,siar}$ is defined in terms of the model parameter θ as

$$\mathcal{L}_{s,siar}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_\infty < \epsilon} \ell_s(\hat{y}_s(\tilde{x}_i; \theta), \hat{y}_s(x_i; \theta))$$

where ℓ_s is a measure of divergence. For SST and PARA, the model outputs a probability distribution, so following Jiang et al. (2020), ℓ_s is chosen as the symmetrized KL-divergence

$$\ell_s(\tilde{y}, y) = D_{KL}(\tilde{y} \| y) + D_{KL}(y \| \tilde{y}).$$

For SEN, the model outputs a linear measure of similarity, so ℓ_s is chosen as the MSE loss

$$\ell_s(\tilde{y}, y) = (\tilde{y} - y)^2.$$

Jiang et al. (2020) suggests augmenting SIAR with Bregman Proximal Policy Optimization(BPPO). However, we find that BPPO does not work well with Round-Robin training, so we decide to not use it in our final model (we do use it when training a teacher model for MM below.)

4.3 Round-Robin(RR) Training

We implement Round-Robin training as a better alternative to sum-of-loss training (see section 5.4). Each epoch, we first iterate through the entire SST dataset, optimizing only $\mathcal{L}_{\mathsf{SST}}$. We then repeat this on the PARA and the SEN datasets. After running this process for enough epochs, we take the best model, and run a few more epochs of fine-tuning on the SST head H_{SST} only.

4.4 Model Merging (MM)

In the Model Merging approach, we first train three models each to optimize the performance on a single task – SST, PARA or SEN. Then we use these three models as the teacher models to jointly train a new student model. The performances of the three teacher models are detailed below.

To make the student model learn from the teacher models, we introduce a new loss function $\mathcal{L}_{s,\text{teacher}_s}$. In each epoch for each task s, we sample a mini-batch of training data and pass it the student and teacher models. Then we measure the divergence between the predictions from the student model and from the teacher model, using the same loss function ℓ_s as in SIAR

$$\mathcal{L}_{s,\text{teacher}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell_s(\hat{y}_s(x_i; \theta), \hat{y}_s(x_i; \theta_{\text{teacher}})).$$

Our final loss function for each task $s \in \{\mathsf{SST}, \mathsf{PARA}, \mathsf{SEN}\}$ looks like

$$\mathcal{L}_s = \mathcal{L}_{s,gt} + \lambda_s \mathcal{L}_{s,siar} + \gamma \mathcal{L}_{s,\text{teacher}_s}$$

where recall that $\mathcal{L}_{s,gt}$ is the ground-truth loss function, $\mathcal{L}_{s,siar}$ is the SIAR regularization, γ is the tunable parameter for teacher-student loss, and teacher$_s$ is the teacher model for the task s.

5 Experiments

5.1 Data

We use the default datasets provided for the project with no extra data.

- Stanford Sentiment Treebank (SST-5) dataset consists of 11,855 single sentences from movie reviews extracted from movie reviews. Each phrase has a label of negative, somewhat negative, neutral, somewhat positive, or positive. The dataset has 8,544 train examples, 1,101 dev examples, and 2,210 test examples.
- Quora Question Pairs (QQP) dataset consists of 404,298 question pairs with labels indicating whether particular instances are paraphrases of one another. The dataset has 283,010 train examples, 40,429 dev examples, and 80,859 test examples. In the earlier stages of the project, we sample a subset of QQP consisting of 32,000 training examples. We write SMALL whenever we use this smaller dataset.
- SemEval STS Benchmark dataset consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). The dataset has 6,040 train examples, 863 dev examples, and 1,725 test examples.

5.2 Training and Evaluation

For all experiments unless noted otherwise, we use the default AdamW optimizer with the default learning rate of $1e-5$. We use the default batch size of 8 and train until the model converges.

We use the default evaluation metrics provided in the project handout. Specifically, For SST and PARA, we compute the accuracy computed between true labels and predictions. For SEN, we use the Pearson correlation coefficient between the true values and predictions. The formula for the overall score is

$$\frac{1}{3}\cdot \mathsf{SST} + \frac{1}{3}\cdot \mathsf{PARA} + \frac{1}{6}\cdot (\mathsf{SEN}+1).$$

5.3 Bi-encoding vs. Cross-encoding

On SMALL, we compare the bi-encoding approach of Reimers and Gurevych (2019) with our baseline cross-encoding approach. The result shows the clear advantage of cross-encoding.

	SST	PARA	SEN	Overall
Baseline	0.502	0.841	0.862	0.758
Bi-Encoding	0.508	0.769	0.768	0.720

5.4 SIAR Hyperparameter Experiment

For SIAR, we experiment with different values of the hyperparameter λ_s. For each experiment we use Cross-Encoding and set the max number of epochs as 20. The table below summarizes the dev accuracies (SMALL) of the three tasks under different λ_s values.

λ_s	SST	PARA	SEN	Overall
0.01	0.486	0.838	0.857	0.751
0.1	0.484	0.848	0.855	0.753
0.5	0.494	0.832	0.858	0.752
2	0.520	0.833	0.870	0.763
3	0.530	0.826	0.867	0.763
5	0.510	0.798	0.867	0.747

The result shows that the performance is best when $\lambda_s = 2$ or $\lambda_s = 3$. Furthermore, $\lambda_s = 3$ performs better when we only consider the data-scarce tasks SST + SEN, which are the tasks where SIAR is expected to be more useful. Therefore, we choose to use $\lambda_s = 3$ in our full dataset training.

5.5 Sum-of-Loss vs. Round-Robin Experiment

We identify a few limitations of the sum-of-loss approach.

- The sum-of-loss approach requires loading batches for all three tasks simultaneously in each training epoch. Due to our limited GPU resources, this often results in CUDA out-of-memory errors. Although we attempted to reduce the batch size, this adjustment led to a longer and less efficient training process.
- We observed that the dev accuracy of PARA is capped, regardless of the number of training epochs. We believe this occurs because the sum-of-loss method averages the losses of all three tasks, which prevents any single task from reaching its optimal performance level.

Based on these limitations, we decide to experiment with Round-Robin approach. The table below compares their performances.

	SST	PARA	SEN	Overall
SIAR + Sum-of-Loss	0.530	0.872	0.873	0.779
SIAR + RR(before SST tuning)	0.477	0.897	0.869	0.770
SIAR + RR	0.518	0.897	0.870	0.783

During the round robin phase, we observe a tradeoff between SST and PARA dev accuracy. We believe this is due to the following reasons:

- PARA and SEN are similar tasks, so they mutually enhance each other's performance. Because we use MSE loss for the SEN task, its loss is larger, and we observe that the model for the SEN task converges very fast within the first few epochs. This in turn benefits the PARA task.
- PARA has much more data compared to SST, resulting in significantly more iterations for PARA training in each epoch. Training on the PARA task optimizes the model's parameters in a way that is highly advantageous for PARA's predictions.

5.6 Model Merging Experiment

Finally, we implement our model merging approach. We first train three teacher models, designed to maximize performance on each specific task.

- SST: We train on the SST dataset alone using SIAR and BPPO.
- PARA: We train on PARA and SEN datasets jointly using SIAR and Round-Robin.
- SEN: We train PARA and SEN datasets jointly using SIAR, Round-Robin, and a random sample of the PARA training dataset per epoch.

Then we train a single student model using SIAR, Round Robin, and our model merging algorithm defined in Section 4.4. Below we compare the performance of our models between the teacher models and the student model.

	SST	PARA	SEN	Overall
SIAR + RR	0.518	0.897	0.870	0.783
Teachers	0.550	0.904	0.878	0.798
SIAR + RR + MM	0.544	0.899	0.875	0.794

5.7 Summary of Results

The table below summarizes the dev dataset accuracies of our key approaches.

Approach	SST	PARA	SEN	Overall
Baseline	0.510	0.881	0.859	0.773
SIAR	0.530	0.872	0.873	0.779
SIAR + RR	0.518	0.897	0.870	0.783
SIAR + RR + MM	**0.544**	**0.899**	**0.875**	**0.794**

Our best model is the last model that uses Cross-Encoding, SIAR, Round-Robin and Model Merging approaches. The test dataset accuracy of our best model is

	SST	PARA	SEN	Overall
SIAR + RR + MM (Test)	0.533	0.897	0.872	0.790

6 Analysis

6.1 Sentiment Classification Analysis

To understand why the performance on SST is much worse than the other two tasks, we create a confusion matrix of SST predictions.

We observe that:

- The model is good at differentiating between good (class 0 or 1) and bad (class 3 or 4) reviews.
- The model is NOT good at predicting a review is neutral when it's truly neutral.
- The model is NOT good at differentiating class 0 from class 1.
- The model is NOT good at differentiating class 3 from class 4.

	0	1	2	3	4
0	62	69	6	62	0
1	44	204	27	14	0
2	10	77	72	58	12
3	2	23	35	158	61
4	0	7	4	56	98

Table 1: Confusion matrix for SST dev. Each value at position (i, j) is the number of examples with true class i that were predicted to be class j.

We read some of the movie reviews in the SST training dataset ourselves together with logits predicted by our model. See Table 2 for an example. Surprisingly, we find that we often agree more with the model's logits than with the ground truth labels. Because of the ambiguous nature of these movie reviews and the subjectivity of human evaluation, the model's performance on the SST task is expected.

6.2 Model Merging Analysis

The regular loss function captures the difference between the predictions and the ground truth labels of the training data, which are discrete. The teacher-student loss function captures the difference between the predictions from the student and the logit outputs from the teacher models, which are float numbers. The latter provides richer information than the "hard" labels in the former. As an analogy, learning from the ground truth is like reading a solution manual that only tells you the answer, while learning from the teacher model is like learning the solution from a TA. There are a few implications:

- The logits from the teacher models is a more accurate labeling of the training data compared with the ground truth labels. For example, in a binary classification task, 70% of human annotators think the sentence is positive and the remaining 30% think it as negative. The ground truth label is 1, but a good teacher model's logit can capture the 70% vs 30% distribution information that the "hard" label does not.

  In our scenario, the SST dataset is very noisy, and we can easily find several examples where the teacher logits paint a more accurate picture than the provided labels. See Table 2.
- The additional teacher-student loss function is more "sensitive". For example, without the additional loss function, the student model may be able to correctly predict the label and the regular loss function gets smaller, so the student model slows down the learning. But it may not correctly capture the true distribution of the data. With the additional teacher-student loss function, the student can continue learning and improving.
- The additional loss function provides regularization, ensuring that the student model does not diverge too much from any of the teacher models and avoid overfitting the training data.

Sentence	Label	Logits
the rock is destined to be the 21st century 's new “conan” and that he 's going to make a splash even greater than arnold schwarzenegger, jean-claud van damme or steven segal.	3	[-1.66, -1.12, -0.43, 2.15, 2.31]
dramas like this make it human .	4	[-1.94, -0.85, 0.73, 2.51, 0.84]
you should pay nine bucks for this : because you can hear about suffering afghan refugees on the news and still be unaffected.	2	[0.45, 1.21, 1.42, -0.53, -1.72]

Table 2: SST training samples where the teacher model's logits are more reasonable than the ground truth label. The first sentence appears more positive than the second, yet the labels indicate otherwise. The third sentence leans on the negative side, yet is labeled 2(neutral).

6.3 Paraphrase Detection Analysis

We look at some examples in the dev dataset, and we find that our model incorrectly predict some sentence pairs as paraphrases when they are not. For example, the pair of sentences `how hard is it to double major in mechanical engineering and physics?` and `is it wise to double major in mechanical engineering and physics considering my interests?` are not paraphrases. Another example is the pair of `is it possible to see who has viewed my profile page?` and `can you see who viewed your profile on quora?`. These examples show that the model is limited in understanding the different semantic meanings the two sentences convey when they have very similar structures.

6.4 Sentence Similarity Analysis

Our model achieves a strong correlation. There are a few cases where the prediction differs significantly from the label due to named entity recognition issues. For example, in the pair of sentences `carney sets high bar to change at boe` and `carney sets high bar to changes at bank of england`, the label is 5 because `boe` is the acronym for `bank of england`. But the model does not have such information and it thinks the two terms are distinct entities, and it gives a prediction score of 2. Another example is the pair of sentences `amazon launches new device for streaming video` and `amazon unveils new fire tv streaming video box`. The label score is 4, since "fire tv streaming video box" refers to the "device" in the first sentence. But the model fails to establish such relationship and gives the pair a score of 2.6.

7 Conclusion

In this paper, we present effective training approaches that enhance the multitask learning capabilities of the minBERT model. We experimented with various techniques, including Cross-Encoding, Round-Robin training, SMART regularization, and Model Merging, demonstrating that our model's performance surpasses the baseline across all three downstream tasks. Our analysis also highlights the model's limitations in understanding nuanced differences between sentences. Due to the limited timeframe of the project, we could not explore all our ideas. For future work, we would like to see experiments with ideas like dynamical loss weights, Multiple Negatives Ranking Loss Learning, and LoRA to achieve even better results.

8 Ethics Statement

We identify several ethical challenges when the model is applied to user facing applications:

- In the scenario where sentiment classification is employed on social media platforms for feed recommendation, if the model incorrectly classifies a negative post as positive and recommends it to users, the users may be exposed to hate speech or other negative contents in the post. Reading such content may have a negative impact on users' mental health. A strategy to mitigate the risk is to have human evaluators reading the posts manually and removing negative posts from users' feeds.
- In the same vein, our model is not robust against an adversarial attacker. Such attackers might exploit this to trick the model's classification system and inject misinformation or hate speech into the feed of users. We may attempt some techniques in Jain et al. (2023) to mitigate such attacks.
- If SemEval is applied to search engine optimization, and a user searches sensitive information, the SemEval algorithm might recommend factually incorrect texts that are semantically similar to the query. For example, if the user searches medical advice, SemEval might recommend a false advertisement or a passage promoting misinformation. To safeguard against such issues, we should not use exclusively SemEval for recommendation algorithms. Instead, we should combine SemEval with an algorithm that can identify texts from authoritative and reliable sources, and include a warning about potential misinformation in the search output.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's mergekit: A toolkit for merging large language models. *CoRR*, abs/2403.13257.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: Knowledge distillation of large language models.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *CoRR*, abs/2309.00614.

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*, pages 2177–2190, Online. Association for Computational Linguistics.

Pride Kavumba, Ana Brassard, Benjamin Heinzerling, and Kentaro Inui. 2023. Prompting for explanations improves adversarial NLI. is this true? yes it is true because it weakens superficial cues. In *Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 2120–2135. Association for Computational Linguistics.

Simran Khanuja, Melvin Johnson, and Partha P. Talukdar. 2021. Mergedistill: Merging language models using pre-trained distillation. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 2874–2887. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1317–1327. The Association for Computational Linguistics.

Jack Le and Febie Lin. 2024. Bert and beyond: A study of multitask learning strategies for nlp.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4885–4901. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *CoRR*, abs/2311.03099.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Yaohui Zhang and Haoyi Duan. 2024. Semantic symphonies: Bertrilogy and bertriad ensembles.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *CoRR*, abs/2307.15043.

A Ideas that don't work

As a reference to future students, we identify a few approaches we tried that result in no significant improvement.

- Architecture modification: We tried changing the activation function of our heads. We also try to change the pooling method of minBERT by adding two [CLS] tokens at the start, using the embedding of one for SST and the other for PARA and SEN. We feel that adding more layers to the model would only exacerbate overfitting.
- Loss weighting: In the sum-of-loss approach, we tried adding weights to the loss function of each task
$$\mathcal{L} = w_1\mathcal{L}_{\mathsf{SST}} + w_2\mathcal{L}_{\mathsf{PARA}} + w_3\mathcal{L}_{\mathsf{SEN}}.$$
- Hyperparameter Tuning: We tried tuning the learning rate and batch size of our model.