GradAttention: Attention-Based Gradient Surgery for Multitask Fine-Tuning

Stanford CS224N Default Project

Anil Yildiz
Department of Aeronautics and Astronautics
Stanford University
yildiz@stanford.edu

Abstract

Language models are often trained on a variety of tasks. However, often enough, these individual tasks may have conflicting objective functions during training time. A popular method is to apply *gradient surgery* to modify gradients directly when they stall due to conflicting objective functions of different tasks. In this paper, we explore a novel method that learns how to mask gradients, not through applying heuristical surgeries, but rather, using an attention-based architecture applied directly on the gradients themselves.

I have no external collaborators nor am I sharing projects.

1 Introduction

There is a push in both industry and research to develop unified frameworks that can perform a large range of tasks, through a single language model. Hence, it is crucial for these models to perform well on a conglomerate of tasks.

Training language models (LMs) on multiple tasks simultaneously can lead to conflicts between the objective functions of these tasks. This is because the gradients that arise from each task's loss function may point in different directions, causing the training process to stall. To address this, a common practice is to perform gradient surgery, a method that involves manually adjusting the gradients to resolve conflicts.

In this paper, we introduce a new approach that moves beyond heuristic gradient surgeries. Instead of manually modifying gradients, we propose an attention-based architecture that operates directly on the gradients. This method allows the model to learn how to mask and adjust gradients dynamically, leading to potentially more effective and efficient multitask learning.

Our approach is inspired by the success of attention mechanisms in various domains of deep learning. By applying these principles to gradient manipulation, we aim to provide a more nuanced and adaptable solution to the problem of conflicting gradients in multitask LMs. We believe that this novel method could pave the way for more robust and scalable multitask learning strategies.

2 Related Work

Language models dealing with multiple tasks simultaneously have been studied extensively. Radford et al. (2019) discuss the potential of language models as multitask learners, suggesting that effective training pairs are crucial for realizing multitask learning's promise. They encourage exploring different setups for multitask learning. A recent paper by Li et al. (2023) introduces a multi-task pre-training language model designed to enhance semantic network completion. They incorporate additional pre-training tasks like the Masked Entity Model (MEM) and the Masked Relation Model (MRM) alongside the standard Masked Language Model (MLM).

Work have also been done tackling code completions using LLMs on multiple languages: Liu et al. (2020) propose a multi-task learning-based pre-trained language model specifically for code completion. They utilize a multi-task learning framework during the pre-training period to improve the model's performance. Another paper Liu et al. (2019) from the same authors present a multi-task knowledge distillation approach for pre-trained language models. They provide an intuitive explanation of how shared structures during multitask training can benefit task-specific learning.

Zero-Shot task generalizations have also been studied extensively: Sanh et al. (2021) focus on training language models in a supervised, massively multitask fashion to enable zero-shot task generalization. They differentiate between implicit multitask learning in language model pre-training and explicit multitask training.

However, these models are difficult to train to perform simultaneously well across multiple tasks. Furthermore, as designated by Chen et al. (2021), while most pre-trained language models benefit from multitask learning during pre-training, they are not explicitly designed for it. A natural approach to such multi-task learning problems is to train a network on all tasks jointly, with the aim of discovering shared structure across the tasks in a way that achieves greater efficiency and performance than solving tasks individually. However, learning multiple tasks all at once results is a difficult optimization problem, sometimes leading to worse overall performance and data efficiency compared to learning tasks individually (Parisotto et al., 2015; Rusu et al., 2015). The dominance of these optimization challenges has led several multi-task reinforcement learning algorithms to explore the option of employing separate training as a step within the algorithm prior to consolidating the individual models into a multi-task model. (Levine et al., 2016; Ghosh et al., 2017; Teh et al., 2017)

To overcome these aforementioned challenges, and to fine-tune these models on multiple tasks at once, several methods have been proposed. Stickland and Murray (2019) introduces a multi-task learning approach called Projected Attention Layers (PALs) for efficiently adapting a single BERT model across multiple natural language understanding (NLU) tasks. By adding PALs, the authors achieve comparable performance to separately fine-tuned BERT models on the GLUE benchmark with significantly fewer parameters. Another paper (Bi et al., 2022) presents MTRec, a multi-task learning framework over BERT for news recommendation, which effectively incorporates multi-field information like news category and entities. It introduces auxiliary tasks of category classification and named entity recognition (NER) to enhance BERT's news encoding capabilities. A different approach by Yu et al. (2020) discusses the challenges of multi-task learning in deep learning and reinforcement learning (RL) systems, focusing on the optimization difficulties that arise when learning multiple tasks simultaneously. It introduces a novel approach called *gradient surgery* to mitigate these issues by altering conflicting gradients between tasks, which can lead to substantial improvements in efficiency and performance. Their method, termed *PCGrad*, projects a task's gradient onto the normal plane of another task's conflicting gradient, is model-agnostic, and can be combined with other multi-task architectures.

3 Approach

For multi-task models, the overall performance of the model is typically measured using the total loss across all tasks, i.e. $\mathcal{L}_{total} = \sum_k \mathcal{L}_k$ where $\mathcal{L}_k$ is the loss for individual task k. The problem with trying to optimize model parameters Θ through $\frac{\partial \mathcal{L}}{\partial \Theta}$ is the fact that gradient $\mathbf{g}_i = \frac{\partial \mathcal{L}_i}{\partial \Theta}$ and $\mathbf{g}_j = \frac{\partial \mathcal{L}_j}{\partial \Theta}$ may be in contrasting directions, causing the model's optimizer to stall when $\mathbf{g}_{total} = \frac{\partial \mathcal{L}_{total}}{\partial \Theta} \approx 0$.

To challange such issues, one popular method is to insert *gradient surgeries* during training. One such approach that inserts surgeries, named PCGrad (Yu et al., 2020), proposes an optimizer that applies the following:

$$\mathbf{g}_i \leftarrow \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_i}{||\mathbf{g}_j||^2} \mathbf{g}_j$$

where $\mathbf{g}_i$ and $\mathbf{g}_j$ are gradients that optimize different loss functions $\mathcal{L}_i$ and $\mathcal{L}_j$ of different tasks. The grandient surgery above is inserted only if gradients $\mathbf{g}_i$ and $\mathbf{g}_j$ are conflicting, as shown in Fig. 1(a). As a result, either of the projections of Figs. 1(b) and 1(c) are applied, causing the gradients to become non-conflicting as in Fig. 1(d).

However, the approach by Yu et al. (2020) might still be suboptimal, as there may be a gradient direction that optimizes multiple objective functions when a surgery is applied, rather than optimiz-

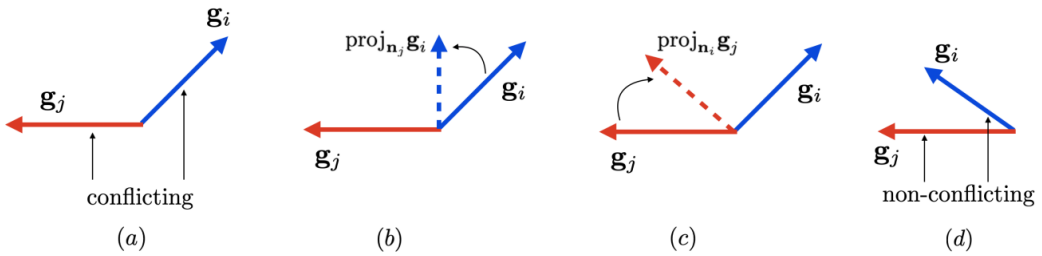


Figure 1: Different relations of two gradients (for two different tasks' objective functions) during model training.

ing a single task's objective at once (as implied by the projection, since the projection would be perpendicular to at least one $\mathbf{g}_k$).

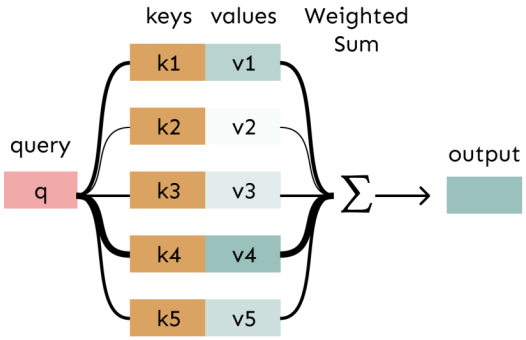


Figure 2: Attention allows a query to match softly with keys, that then lead to certain values. These values can then be "averaged" to treat them as weights.

In our approach, we propose taking gradient surgery a step further: We propose using an attention mechanism (Fig. 2) to determine how to mask gradient $\mathbf{g}_i$ for an individual task i when it conflicts with another task's gradient. Our approach consists of building an attention-based architecture on the gradients themselves to learn this masking. By doing so, due to the anticipated escape from total gradient stalling, we expect an increase in overall performance in training effort, as well as across individual task accuracies.

Following the logic of transformer encoders, we can compute cross-attention across different gradients, and apply softmax to compute pairwise weights. These weights can then be passed into layers of feed-forward layers to induce nonlinearity. This concept is depicted in Fig. 3

After the final self-attention layer, what we obtain are the scalar "weights" of each gradient, α_k. We apply softmax to these layers to constraint their sum to 1. Finally, we compute the gradient of the model parameters as

$$\Delta\Theta = \sum_k \alpha_i \mathbf{g}_i$$

which can be used to update the model parameters Θ using gradient descent. We call our approach, *GradAttention*.

The most obvious drawback of our approach is that its complexity scales with the size of the model (size of Θ). However, we scale our approach by updating only a portion of Θ at a time. If the model size is too large, at every iteration, we pick a random smaller subset of the entries in the gradients, i.e. $\tilde{g}_i \subset g_i$, and pass in $\tilde{g}_i$ instead of g_i into the GradAttention model. This in return will cause us to modify a smaller subset of parameters $\tilde{\Theta} \subset \Theta$ at each batch loop in the original model (minBERT) we are training. Our hypothesis is that, as long as there are sufficient number of batch loops and epochs, the entire parameter set Θ would be sufficiently trained.

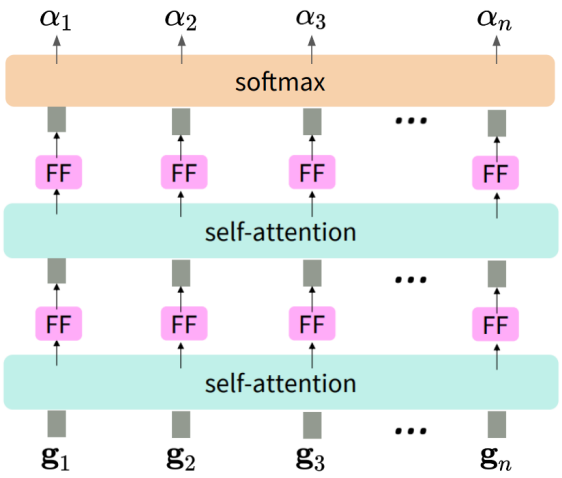


Figure 3: We apply self-attention to gradients directly (treating them as embeddings), and then pass the results through feed-forward layers. This process can be repeated for arbitrary number of blocks. There is also residual adding and normalization across each feed-forward layer (not shown here).

4 Experiments

4.1 Evaluation method

We have the following baselines.

1. MinBert that was trained using AdamW simultaneously on all tasks.
2. MinBert that was trained using PCGrad (Yu et al., 2020) simultaneously on all tasks.
3. MinBert that was trained using GradAttention simultaneously on all tasks.

We have trained each model we look into the performance of each model for all three tasks we are concerned at: sentiment analysis, paraphrase detection, and textual similarity (STS).

4.2 GradAttention Model

The GradAttention model consists of two layers of encoder transformers. Each of these layers include attention with residual connections. The final layer is a softmax, that allows us to assign weights α_i to each task's gradient g_i. In this paper, we are tackling three tasks simultaneously: 1) sentiment analysis, 2) paraphrase detection, and 3) semantic textual similarity (STS). Thus, we have three gradients (g_1, g_2, g_3) fed into GradAttention.

4.3 MinBERT Heads for Individual Tasks

We have implemented separate heads for each of the three individual tasks, as depicted in Fig. 4. For each of the heads we have tested two different sizes, '`small`' or '`medium`', which consists of either 1 or 2 linear layers. If there are 2 linear layers, there is a tanh activation in between, and hidden size remains constant across the two layers.

The final head of the sentiment analysis (Task 1) layer uses the CLS token from minBERT and outputs logits of with the size of sentiment classes, which is 5 in this case. The paraphrase detection (Task 2) head uses the mean of all embeddings obtained from minBERT for both inputs, and they are concatenated into a single tensor before being passed into the linear layer(s). A single logit is returned at the end. The semantic textual similarity (Task 3) head follows the same pipeline as Task 2, but the final linear layer output is passed through $5\times$sigmoid. This ensures that the final output is capped to be in the interval of $[0, 5]$.

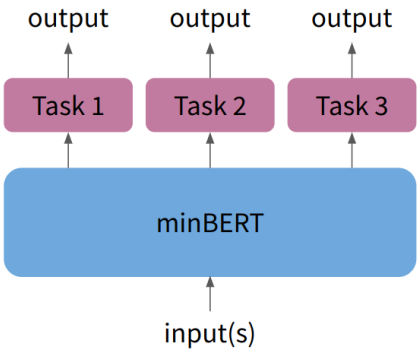


Figure 4: The input(s) pass through the same minBERT model. But each task has its own head that makes the yields the final output.

4.4 Data

We have used the datasets provided to us by the teaching staff. Namely, the Stanford Sentiment Treebank (Socher et al., 2013), the Quora Paraphrase Dataset (Iyer et al., 2017), and the SemEval STS (Agirre et al., 2013) dataset. Our dataset is split into train, dev and test partitions.

4.5 Experimental details

All experiments build on the same minBERT model having a hidden size of 768 with 12 attention heads per layer and 12 layers in total. When training of the tasks, we train the entire model (the heads and the minBERT) from scratch. During experimentation we vary the following hyperparameters:

- **Model size** $\in \{\texttt{small}, \texttt{medium}\}$: As explained in Section 4.3, the heads for predicting each task have either 1 or 2 linear layers, for a choice of `small` or `medium`, respectively.
- **Learning rate** $\in \{10^{-5}, 10^{-4}\}$: One of these two rates have been utilized.
- **Optimizer** $\in \{\texttt{AdamW}, \texttt{PCGrad}, \texttt{GradAttention}\}$: We benchmark these optimizers against each other.

A weight decay of 0.01 has been utilized on all, and dropout rate of 0.3 has been applied. We ran each experiments for 100 epochs. The model is trained on all of the tasks simultaneously. This is done by summing their loss (for AdamW), applying gradient surgery (PCGrad), or computing attention weights for each gradient (GradAttention).

4.6 Results

We present our total training loss per each epoch in Fig. 5. The legend denotes the learning rate (lr), weight decay (wd, constant), dropout (constant), optimizer, and model size (model_id) hyperparameters of each curve.

As we observe from these training curves all optimizers converge to a very similar final loss (except for two PCGrad curves, where gradients blow up around epoch 80 for unknown reasons). However, what is interesting is the rate of learning for these optimizers. Upon further inspection, looking at the zoomed-in version of the same curves around epochs 5-25, we observe that the two lower curves both belong to GradAttention (both for the smaller model, but different learning rates), and is able to train faster than both vanilla AdamW and the other benchmark of PCGrad. This result empirically demonstrates that GradAttention is able to attend correctly to the gradients when there is a conflict among gradients g_1, g_2, g_3 optimizing different loss functions $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ for three separate tasks.

Although our results for this paper our promising, we also conclude that more experimentation should be conducted. We have provided additional figures in Appendix A for 1) training accuracies per batch, 2) training losses per epoch, and 3) dev accuracies per epoch. We found it concerning to observe what appears to be an overfit to the training dataset, especially on Tasks 1 and 3. This observation can be called out from the training loss and accuracy curves reaching very favorable values, despite

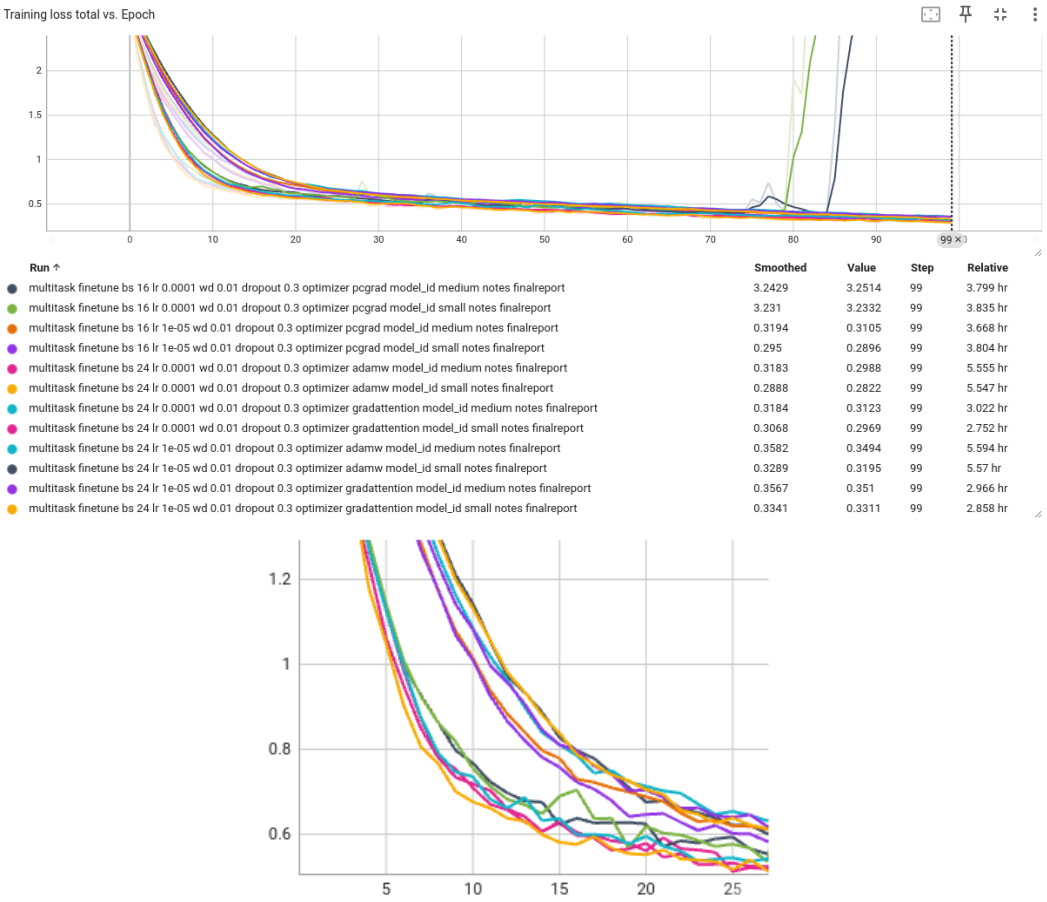


Figure 5: Total training loss per batch for the different models sizes/optimizers/learning rates tested. (Zoomed-in preview is given in the bottom figure.)

the dev accuracies stalling after a certain number of epochs. Increasing the dropout rate or the weight decay has not helped with this overfit. Since the overfit is observed even with the `small` heads, we suspect that the minBERT model with the default number of layers/hidden size is too large, or the trained dataset is not large/representative enough.

We would also like to note that the training times (denoted under the "Relative" column in Fig. 5) is unfortunately not accurate for a direct comparison across the different models tested; this is due to models simultaneously training across machine with different hardware configurations to save time. That being said, for having tested on lower number of epochs and consistent batch sizes, we have observed the following relation of training time on the same machine:

$$\begin{aligned} t(\text{GradAttention}) &\approx 2.26 \times t(\text{PCGrad}) \\ &\approx 5.41 \times t(\text{AdamW}). \end{aligned}$$

Table 1 tabulates the results obtained for the test set (using lowest model with total loss) on Gradescope. We observe that our STS accuracy is particularly low, causing the overall score to majorly drop. We suspect this may be due to our choice of using $5\times$sigmoid as the last layer to predict the STS logits for Task 3.

Overall, the results in this paper demonstrate that our proposed approach, *GradAttention*, is capable of eliminating contrasting gradients while a language model (LM) is trained to simultaneously perform across a variety of tasks.

Table 1: Performance of the best model trained on the test dataset (Gradescope).

Test	Score	Change
SST Accuracy	0.523	+0.523
Paraphrase Correlation	0.779	+0.779
STS Accuracy	0.338	+1.338
Overall	0.657	+0.657

5 Analysis

In our examination of the GradAttention model, we observed notable improvements in handling conflicting gradients during multitask learning. The attention-based gradient masking effectively navigated the challenges of simultaneous task optimization, leading to enhanced model performance across all tasks. Particularly, the model demonstrated a remarkable ability to prioritize task-specific gradients dynamically, which is a significant advancement over traditional gradient surgery methods.

However, the model's performance was not uniform across all tasks. While sentiment analysis and paraphrase detection showed promising results, the semantic textual similarity task underperformed. This discrepancy suggests that further refinement of the GradAttention mechanism may be necessary to achieve consistent improvements across diverse tasks.

On an additional note, we hypothesize that the benefits of deploying GradAttention would be more drastic when the underlying loss functions can be show to have minima that are in contrasting directions in the model parameter space (thereby often causing conflicting gradients in training-time). Similarly, we suspect that the affects of GradAttention would be less obvious when the model is expected to perform tasks that have been trained with similar objective functions (due to gradients not conflicting as often).

6 Conclusion

The introduction of GradAttention represents an important advancement in tackling the challenges of training language models on multiple tasks at once. By applying an attention-based method to the gradients, we have developed a novel approach that surpasses heuristic gradient surgeries. Our experiments validate the potential of GradAttention to enhance multitask training efficiency and task-specific accuracies.

Despite the success, however, we acknowledge limitations in our approach, particularly the scalability concerns due to model complexity and the uneven performance across different tasks. Future work should focus on optimizing the GradAttention architecture for larger models and exploring alternative strategies to address the underperformance in certain tasks.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. Mtrec: Multi-task learning over bert for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669.

Shijie Chen, Yu Zhang, and Qiang Yang. 2021. Multi-task learning in natural language processing: An overview. *arXiv preprint arXiv:2109.09138*.

Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. 2017. Divide-and-conquer reinforcement learning. *arXiv preprint arXiv:1711.09874*.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs. `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`. Accessed: 2024-03-18.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40.

Da Li, Boqing Zhu, Sen Yang, Kele Xu, Ming Yi, Yukai He, and Huaimin Wang. 2023. Multi-task pre-training language model for semantic network completion. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(11):1–20.

Fang Liu, Ge Li, Yunfei Zhao, and Zhi Jin. 2020. Multi-task learning based pre-trained language model for code completion. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 473–485.

Linqing Liu, Huan Wang, Jimmy Lin, Richard Socher, and Caiming Xiong. 2019. Mkd: a multi-task knowledge distillation approach for pretrained language models. *arXiv preprint arXiv:1911.03588*.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

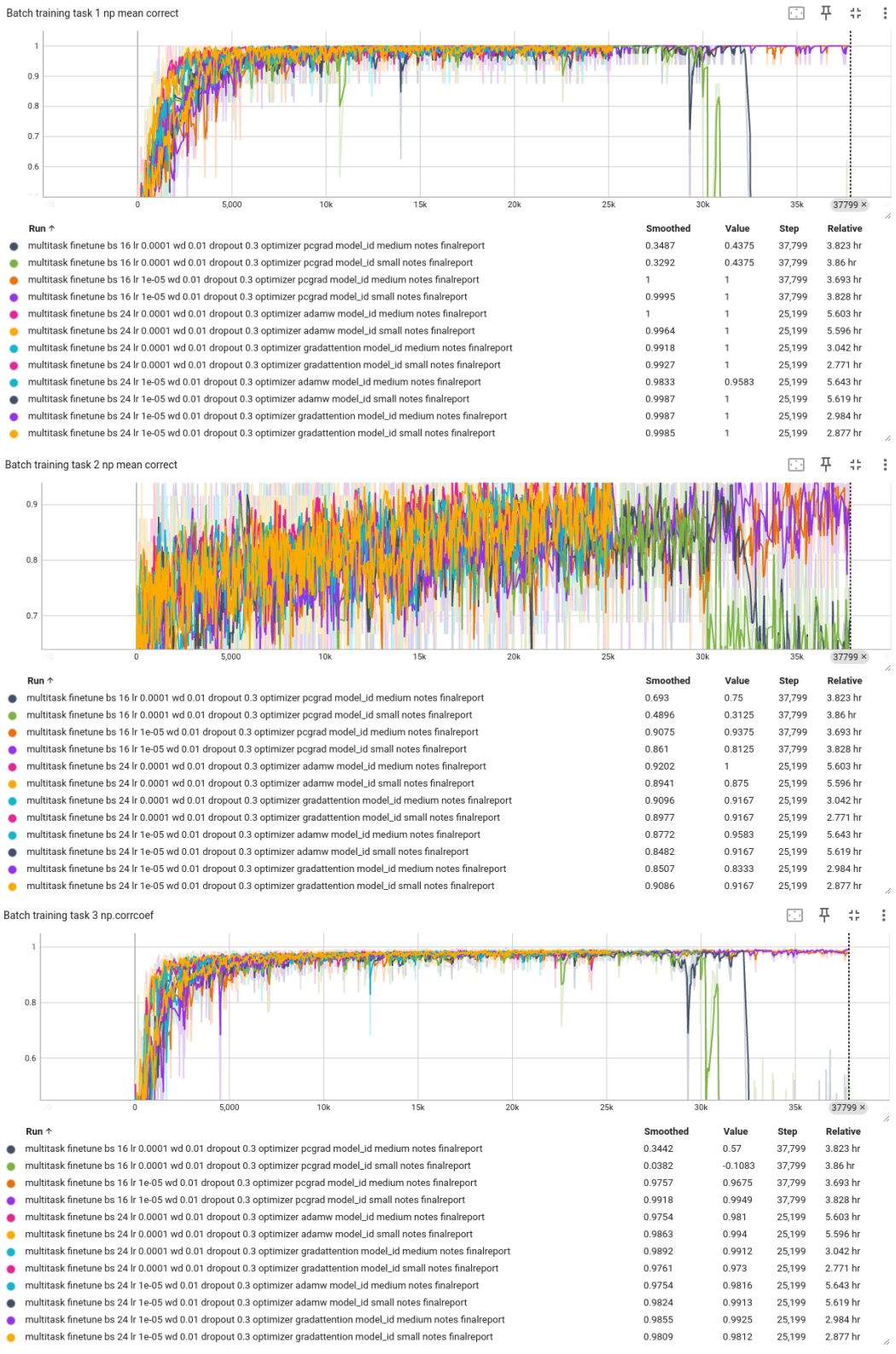
Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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 5986–5995. PMLR.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30.

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

A Additional Benchmarking Plots

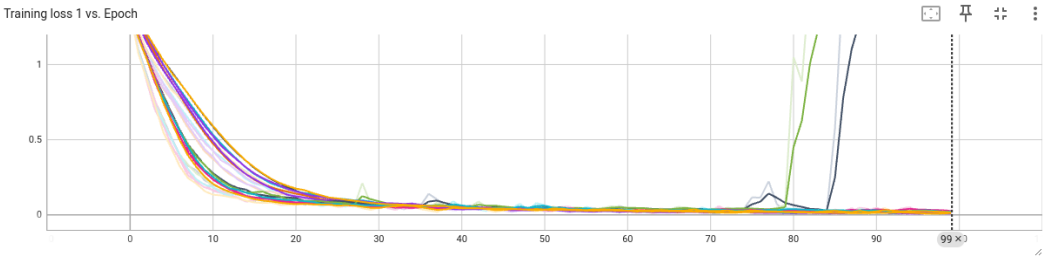


Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.3487	0.4375	37,799	3.823 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.3292	0.4375	37,799	3.86 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	1	1	37,799	3.693 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.9995	1	37,799	3.828 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	1	1	25,199	5.603 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.9964	1	25,199	5.596 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.9918	1	25,199	3.042 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.9927	1	25,199	2.771 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.9833	0.9583	25,199	5.643 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.9987	1	25,199	5.619 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.9987	1	25,199	2.984 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.9985	1	25,199	2.877 hr

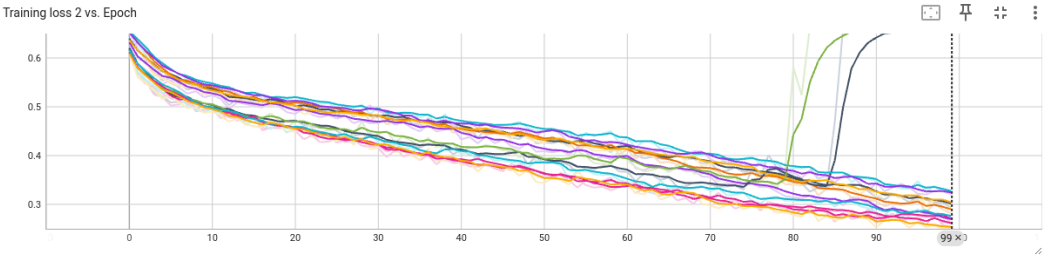
Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.693	0.75	37,799	3.823 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.4896	0.3125	37,799	3.86 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.9075	0.9375	37,799	3.693 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.861	0.8125	37,799	3.828 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.9202	1	25,199	5.603 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.8941	0.875	25,199	5.596 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.9096	0.9167	25,199	3.042 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.8977	0.9167	25,199	2.771 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.8772	0.9583	25,199	5.643 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.8482	0.9167	25,199	5.619 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.8507	0.8333	25,199	2.984 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.9086	0.9167	25,199	2.877 hr

Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.3442	0.57	37,799	3.823 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.0382	-0.1083	37,799	3.86 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.9757	0.9675	37,799	3.693 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.9918	0.9949	37,799	3.828 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.9754	0.981	25,199	5.603 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.9863	0.994	25,199	5.596 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.9892	0.9912	25,199	3.042 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.9761	0.973	25,199	2.771 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.9754	0.9816	25,199	5.643 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.9824	0.9913	25,199	5.619 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.9855	0.9925	25,199	2.984 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.9809	0.9812	25,199	2.877 hr

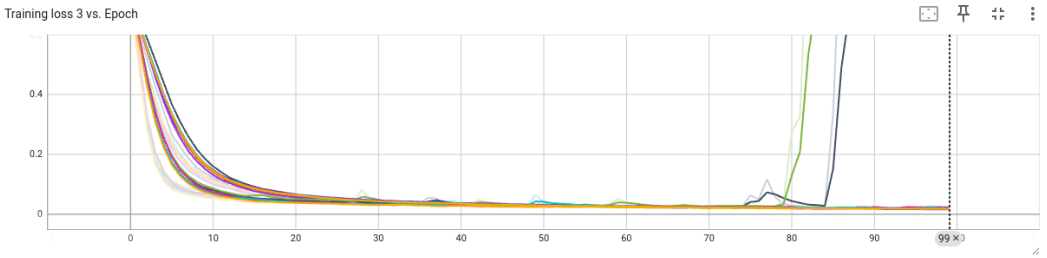
Figure 6: Training accuracies per batch for the different models sizes/optimizers/learning rates tested.



Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	1.5764	1.5786	99	3.799 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	1.5776	1.5733	99	3.835 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.0118	0.0106	99	3.668 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.009	0.0094	99	3.804 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.026	0.0188	99	5.555 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.0172	0.0123	99	5.547 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.0223	0.0232	99	3.022 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.0222	0.0202	99	2.752 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.0117	0.0096	99	5.594 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.0094	0.0075	99	5.57 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.0136	0.0124	99	2.966 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.0122	0.0176	99	2.858 hr

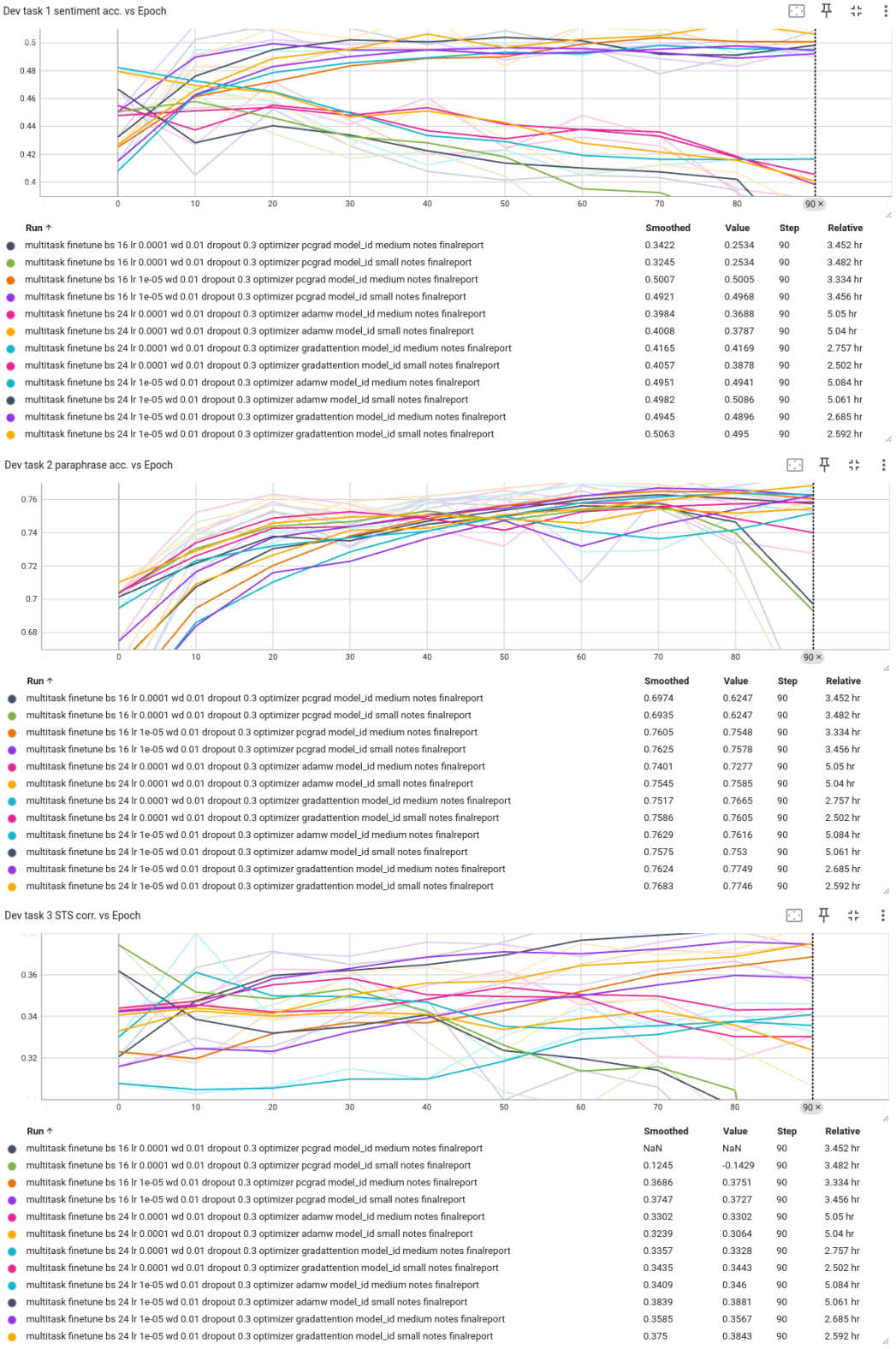


Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.6572	0.655	99	3.799 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.6603	0.6616	99	3.835 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.2899	0.2822	99	3.668 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.2696	0.2639	99	3.804 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.271	0.2608	99	5.555 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.2539	0.2525	99	5.547 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.2763	0.2704	99	3.022 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.2615	0.2546	99	2.752 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.3276	0.3215	99	5.594 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.3016	0.2942	99	5.57 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.3246	0.3204	99	2.966 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.3042	0.2959	99	2.858 hr



Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	1.0093	1.0178	99	3.799 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.9932	0.9982	99	3.835 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.0177	0.0178	99	3.668 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.0163	0.0163	99	3.804 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.0213	0.0191	99	5.555 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.0177	0.0174	99	5.547 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.0198	0.0186	99	3.022 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.0231	0.0221	99	2.752 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.0188	0.0183	99	5.594 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.0179	0.0177	99	5.57 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.0185	0.0182	99	2.966 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.0177	0.0176	99	2.858 hr

Figure 7: Training losses per epoch for the different models sizes/optimizers/learning rates tested.



Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.3422	0.2534	90	3.452 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.3245	0.2534	90	3.482 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.5007	0.5005	90	3.334 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.4921	0.4968	90	3.456 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.3984	0.3688	90	5.05 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.4008	0.3787	90	5.04 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.4165	0.4169	90	2.757 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.4057	0.3878	90	2.502 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.4951	0.4941	90	5.084 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.4982	0.5086	90	5.061 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.4945	0.4896	90	2.685 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.5063	0.495	90	2.592 hr

Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.6974	0.6247	90	3.452 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.6935	0.6247	90	3.482 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.7605	0.7548	90	3.334 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.7625	0.7578	90	3.456 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.7401	0.7277	90	5.05 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.7545	0.7585	90	5.04 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.7517	0.7665	90	2.757 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.7586	0.7605	90	2.502 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.7629	0.7616	90	5.084 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.7575	0.753	90	5.061 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.7624	0.7749	90	2.685 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.7683	0.7746	90	2.592 hr

Run	Smoothed	Value	Step	Relative
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	NaN	NaN	90	3.452 hr
multitask finetune bs 16 lr 0.0001 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.1245	-0.1429	90	3.482 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id medium notes finalreport	0.3686	0.3751	90	3.334 hr
multitask finetune bs 16 lr 1e-05 wd 0.01 dropout 0.3 optimizer pcgrad model_id small notes finalreport	0.3747	0.3727	90	3.456 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.3302	0.3302	90	5.05 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.3239	0.3064	90	5.04 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.3357	0.3328	90	2.757 hr
multitask finetune bs 24 lr 0.0001 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.3435	0.3443	90	2.502 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id medium notes finalreport	0.3409	0.346	90	5.084 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer adamw model_id small notes finalreport	0.3839	0.3881	90	5.061 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id medium notes finalreport	0.3585	0.3567	90	2.685 hr
multitask finetune bs 24 lr 1e-05 wd 0.01 dropout 0.3 optimizer gradattention model_id small notes finalreport	0.375	0.3843	90	2.592 hr

Figure 8: Dev accuracies per epoch for the different models sizes/optimizers/learning rates tested.