Transformer Adapters
Team Falcon: CS 7643

Sai Guttikonda, Brandon Stewart, Carlo Cochran
Georgia Institute of Technology
sguttikonda7@gatech.edu, rstewart61@gatech.edu, ccochran37@gatech.edu

Abstract

Fine-tuning masked language models for downstream tasks yield state of the art performance. But this performance has a considerable computational cost, since such fine-tuning retrains 100% of the original model parameters on downstream tasks. We adopt the transformer adapter framework proposed by Houlsby et al. [6] and improved by Pfeiffer et al. [15] to explore if similar performance to fine-tuning can be achieved with massively reduced computational requirements. Our approach combines independently trained single task adapters similar to Pfeiffer et al. [15]. For comparison with full-parameter fine-tuning, we train adapters on top of RoBERTa [11] for two of the same domains and four of the same downstream tasks as Gururangan et al. [5]. We find that transformer adapters can improve model performance without any pre-training, but in contrast to full fine-tuning [5], pre-training with adapters does not provide further benefit.

1. Introduction & Motivation

Fine-tuning [16] transformer [17] based language models for different downstream tasks is common in natural language processing (NLP) to achieve state of the art task performance. However, the computational resources required for this tuning can be prohibitive in terms of cost and time. A base RoBERTa [11] or BERT [3] model contains over 100 million parameters, and fine-tuning for downstream tasks uses 100% of the trainable parameters. Using adapters developed by Houlsby et al. [6] and improved by Pfeiffer et al.[15], **we explore if the performance benefits of pre-training can be replicated with a fraction of the trainable parameters**. We take the fine-tuning results from Gururangan et al. [5] as the baseline to compare against. We find that transformer adapters improve the performance of fine-tuning without pre-training, but do not realize the benefits of low resource "task adaptive pre-training" (TAPT)[5], or high resource "domain adaptive pre-training" (DAPT)[5].

For each classification task, we report results for four different adapter configurations, each of which corresponds to an adaptive pre-training procedure from Gururangan et al. [5]: i) training the classification adapter directly on top of RoBERTa, ii) fine-tuning the base model after pre-training with the task corpus (TAPT Configuration), iii) fine-tuning the base model after pre-training with the domain corpus (DAPT Configuration), and iv) fine-tuning the base model using adapter fusion [15] which fuses both the domain and task masked language models (MLM) and trains a classifier on top for the downstream task (DAPT + TAPT Configuration). In all our configurations, we use transformer adapters to reduce the number of trainable parameters.

Kim et al. [9] similarly analyze the performance and efficiency benefits of adapters as an alternative to fine-tuning. Our analysis goes further in exploring domain MLM adapters as an analog to DAPT, and also exploring fusion of domain and task MLMs as an alternative to sequential pre-training. Similar to Kim et al. [9], we find that directly training an adapter on the base model can achieve performance similar to TAPT. However, we were not able to find much additional benefit to pre-training adapters on the domain or task corpora. So the performance benefits of DAPT still requires full fine-tuning.

Motivation: Our contribution is evaluating the viability of stacking transformer adapters as an alternative to expensive full-model TAPT and DAPT. With a fraction of the trainable parameters, and their potential for reuse in different combinations, adapters could massively reduce the time and energy currently required for the benefits of pre-training.

1.1. Datasets

Following Gururangan et al. [5], we train our adapter configurations on top of RoBERTa [11] for two domains (Computer Science and Biomed from S2ORC [12]) and four downstream classification tasks (ChemProt [10], RCT [2], ACL-ARC [8], and SciERC [14]).

We consider the Computer Science and Biomed domains and tasks studied by Gururangan et. al. [5]. Both domain corpora were obtained from S2ORC by filtering on subject, and extracting only the abstracts [12]. With no clear guid-

ance on how the Biomed domain was constructed in Gururangan et al. [5], we include paper abstracts that were in either Biology **or** Medicine.

The labeled task classification datasets explored were ChemProt [10], RCT [2], ALC-ARC [8], and SciERC [14]. These classification datasets are publicly available from the Github repository provided by Gururangan et. al. [4].

2. Background

The transformer architecture [17] is state of the art for NLP. Masked Language Models (MLM) such as BERT [3] and RoBERTa [11] allow training on massive corpora in a self-supervised manner, by randomly masking or replacing tokens for the model to predict. These pretrained MLMs can then be fine-tuned on downstream tasks for state of the art performance [16].

Gururangan et al. [5] demonstrate state of the art performance on four Biomedical and Computer Science classification tasks by further pre-training RoBERTa on a relevant corpora before the target task. This pre-training is broken into two phases, each of which updates all the original model weights: (1) MLM pre-training on a larger relevant corpus, such as Computer Science papers (DAPT), and (2) MLM pre-training on the much smaller corpus for the relevant task (TAPT) [5]. After pre-training, the model is then trained directly on the downstream task, again allowing all weights to be updated [5]. Because such fine-tuning requires updating all the original model parameters, each downstream task requires extensive retraining and storing/sharing these models can be expensive.

Houlsby et al. [6] propose an adapter framework that allows fine-tuning large models with greater parameter efficiency. The base model is frozen, but instead of just training a new head, adapter layers are inserted between all transformer layers in the original network [6]. Similar to autoencoders, each adapter layer projects the d-dimensional features of the transformer output down to a bottleneck of size m, which is then projected back up to the original size d [6]. The efficiency of the adapter can be tuned by adjusting the size of m, but generally each adapter is 0.5% to 5% of the original model's size in terms of parameters [6].

Kim et. al. [9] applied the framework of Houlsby et al. [6] to the experiments in Gururangan et al. [5] to see if similar performance could be achieved, with significantly improved parameter efficiency. Their analysis was limited to MLM pre-training on the task corpus, while the domain corpus pre-training was omitted [9]. They found that pre-training on the task corpus had little benefit over simply training an adapter directly on top of RoBERTa [9].

3. Approach

We build on the results of Kim et al. [9] while considering alternative adapter configurations that include domain pre-training. Similar to Gururangan et al. [5], we trained MLM adapters on task and domain corpora. But each MLM was trained independently rather than successively on top of each other. This training independence allows greater CPU efficiency and reusability as the MLM adapter for a domain can be shared by all downstream tasks.

After training the MLM adapters for each domain and task, these MLM adapters were fused [15] and the classification adapter was stacked on top. The list below shows each of the tested adapter configurations (all trained on top of a frozen RoBERTa base model), and the corresponding pre-training configuration from Gururangan [5] (Figure 1):

1. **Baseline:** Adapter for task classification
2. **DAPT:** Adapter for domain MLM + Adapter for task classification
3. **TAPT:** Adapter for task MLM + Adapter for task classification
4. **DAPT + TAPT:** Fused adapters for domain and task MLMs + Adapter for task classification

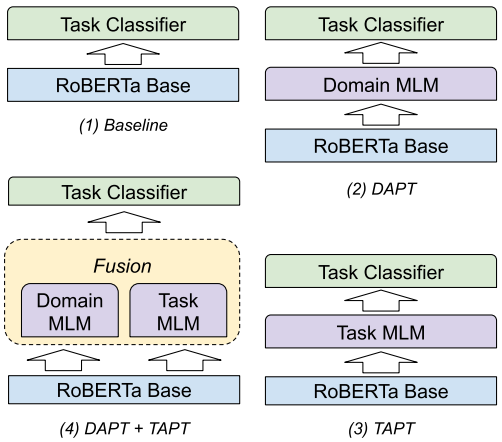


Figure 1: Adapter configurations tested

In all cases, only the last adapter is unfrozen and trained. So for example, when training (2), the adapter for the domain MLM has its weights frozen. This allows for each adapter to be self-contained and independent, while potentially reducing representational power as the trainable parameters are more limited.

The configuration of our adapters were based on Pfeiffer et. al. [15], and the only significant architecture parameter altered was reduction factor (RF). RF is defined as $\frac{1}{m}$. This provides a simple lever to balance the representational power of the adapter against its task performance.

3.1. Implementation

We implement all experiments on top of HuggingFace [7] and Adapter Transformers [1], with a PyTorch backend. The adapters for domain and task MLMs are trained on a single NVIDIA T4. The task classifiers are trained on a mix of NVIDIA T4s and NVIDIA RTX 3070. All training is mixed precision (*fp16=True*) as this nearly halved training time with no detectable loss in performance.

3.1.1 Baseline (Adapter for task classification):

We load a pretrained $RoBERTa_{BASE}$ model, freeze its weights, and then add a trainable adapter [15]. Instead of using 2 adapters [6], we use the architecture with one adapter which is added after the feedforward layer in the transformer architecture [17]. For RoBERTa [11], the input to the adapter layer has a hidden size of 768. We then add a classification head on top of the model. This classification head included as part of HuggingFace consists of a dropout layer with a default probability of 0.1, a 768x768 linear layer, a tanh activation function, and a final 768x<vocab size> linear layer to classify output labels. We use cross-entropy loss as mentioned below.

3.1.2 DAPT & TAPT:

The MLM adapter is trained on the domain or task corpus, and then placed upstream of a new adapter for fine-tuning task classification. Then a classification head is added as with the Baseline configuration. The framework trains this model by freezing the weights of the $RoBERTa_{BASE}$, but updating the stacked adapters to fine-tune for the downstream task. The output of each RoBERTa layer is passed as input to the MLM adapter and the output of this MLM adapter is then fed into the task classification adapter, which then feeds into the next layer of RoBERTa.

3.1.3 DAPT + TAPT (Fused adapters):

For this part, we use adapter fusion to fuse both the task and domain MLM pre-trained adapters. Adapter fusion [15] attempts to combine the information from multiple adapters. We then stack our classification adapter on top of this fused adapter followed by classification head on top of this.

4. Experiments and Results

4.1. Experiments Setup

Baseline: We use Gururangan et al. [5] as a baseline, and correspondingly train our adapters on $RoBERTa_{Base}$.

Loss function, Optimizer & Accuracy: For all classification and MLM experiments, we use Softmax logits and Cross-Entropy loss, and optimization was AdamW [13]. For classification accuracy, we use F1-micro for ChemProt and RCT and F1-macro for SciERC and ACL-ARC.

Learning Rate Scheduler & Hyper-parameters: We use a linear learning rate decay scheduler for all our experiments. We use the highest batch size possible on available hardware, which was 32 for ChemProt and RCT, 163 for SciERC and ACL-ARC, and 163 for CS, and 142 for Biomed. Learning rates of {1e-4, 3e-4, 5e-4, 6e-4} are used for experiments as they empirically work well for our smaller batch sizes and trainable parameters. We experiment with {1e-4, 3e-4, 5e-4} for RCT, SciERC and ACL-ARC; and additionally 6e-4 for ChemProt. We also experiment with different reduction factors {4, 8, 16, 32, 64} to study the effects on accuracy. The number of epochs is scaled based on the dataset size. This is 10 epochs for RCT, 50 epochs for ChemProt and SciERC, and 100 epochs for ACL-ARC.

For domain and task MLMs, we experiment with learning rates of {1e-3, 3e-3, 6e-3} and a batch size of 142. We also experiment with the same reduction factors as above. We use $RoBERTa_{BASE}$ with frozen weights as our base model, and add a MLM head on top of this model. The MLM head provided by HuggingFace consists of a 768x768 linear layer, followed by a "gelu" non-linearity with normalization, and finally a 768x<vocab size> linear layer.

4.2. MLM Adapter Experiments

4.2.1 Preprocessing corpora for MLM

Since RoBERTa truncates each sample after 80 tokens, we combine sentences to avoid wasteful padding and increase the number of useful tokens trained in each mini-batch. As with RoBERTa, the self-supervised labels are created by masking 15% of tokens.

Domain: The large domain corpora presents a challenge for our available disk and GPU resources, so we limit each domain corpus to 10 million samples. S2ORC contains papers from many different languages. We restrict the training data to papers that were primarily encoded in the Latin subset of UTF-8, as the tasks are primarily English. Further filtering to only English papers would be a more complex task that we did not have time to explore.

Task: For all 4 task specific datasets, we used preprocessed datasets from Gururangan et al. [5] for better comparison to our baseline results. Each MLM was trained only on the input text, ignoring the labels.

4.2.2 MLM Training

For all MLM objectives, the MLM adapter and head are trained directly on a frozen $RoBERTa_{BASE}$ model.

Domain: Hypertuning of learning rate and reduction factor is performed on a subset of 1 million samples. The model with the lowest loss is trained on all 10 million samples, and used for downstream tasks. Based on the hypertuned reduction factor, the batch size is maximized for the GPU capabilities. Computer Science domain is trained with a batch size of 163 and the reduction factor of 16. For Biomed, the batch size and reduction factor are 142 and 8, respectively.

Task: Each task corpora is constructed from the input text, ignoring the labels. Low resource domains (ChemProt, SciERC and ACL-ARC) are trained with 100 epochs, and the high resource domain (RCT) is trained on 10 epochs. For hypertuning, the epochs are reduced by a factor of 10.

The best learning rate and reduction factor is used to train the final adapter, which is then used for downstream tasks. Results are in table [1].

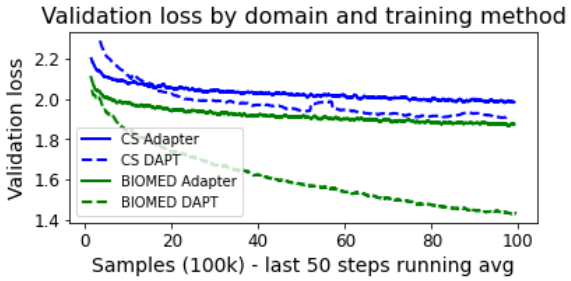


Figure 2: Domain Loss. See table [7] (appendix) for test set loss.

4.2.3 Results and Analysis for MLM

In our experiments, we keep the reduction factor constant and vary the learning rate, then we keep the learning rate constant and vary the reduction factor. As shown in table [1], increased learning rates from Gururangan et al [5] for all tasks decrease the loss. This is expected. Compared to full fine-tuning, higher learning rates are optimal due to the decreased number of parameters.

In contrast to tuning the learning rate, varying the reduction factor shows only minor changes in the validation loss. To minimize trainable parameters, larger reduction factors are preferred when loss is similar. Smaller adapters (larger reduction factors) converged more quickly and even maximizing the reduction factor ($m = 1$) shows only minor increases in loss.

Dataset	Loss		RF	Epochs
	LR=1e-5 [5]	LR=5e-5		
ChemProt	1.141	1.04	8	100
RCT	1.626	1.513	4	10
ACL-ARC	3.587	2.408	8	100
SciERC	2.256	1.450	8	50

Table 1: Validation Loss of the last epoch after pre-training

Domain MLM adapters do not match the loss reported by Gururangan et al. [5] (1.34 for CS, 0.99 for Biomed). However, due to resource constraints we are not able to precisely replicate the domain datasets, so the loss is not exactly comparable. To evaluate the loss difference between MLM adapters and DAPT, we used our dataset with the same hyperparameters given by Gururangan et al. [5]. For example: after 10 million training samples, the CS domain adapter has a loss of 1.898, compared to a loss of 1.558 for full fine-tuning (Figure [2]). With a reduction factor of 16, the CS MLM has 1.23% of the trainable parameters of $RoBERTa_{BASE}$, and this does not seem to have the representational power to match the loss achieved by full fine-tuning.

4.3. Baseline text classification

We study the effects of: 1) number of trainable parameters on the validation accuracy for fixed learning rates; 2) learning rates for fixed reduction factors on the validation accuracy; 3) model overfitting on high (RCT) vs. low resource datasets. For this configuration, only the head and adapter contribute to the number of trainable parameters.

4.3.1 Results & Analysis

Figure [3a] shows how accuracy changes with the number of trainable parameters, for learning rates {1e-4 & 5e-4}. ALC-ARC is the most difficult task, and most clearly shows how decreases in trainable parameters benefit from higher learning rates. SciERC performs best with the most trainable parameters, likely indicating the complexity of this task requires greater representational power. ChemProt seems to be a simpler task, performing better with a lower number of parameters. For the high resource RCT, accuracy only degrades when both learning rate and trainable parameters increase. Based on these results, we empirically choose the best hyperparameters per classification task (table [1]).

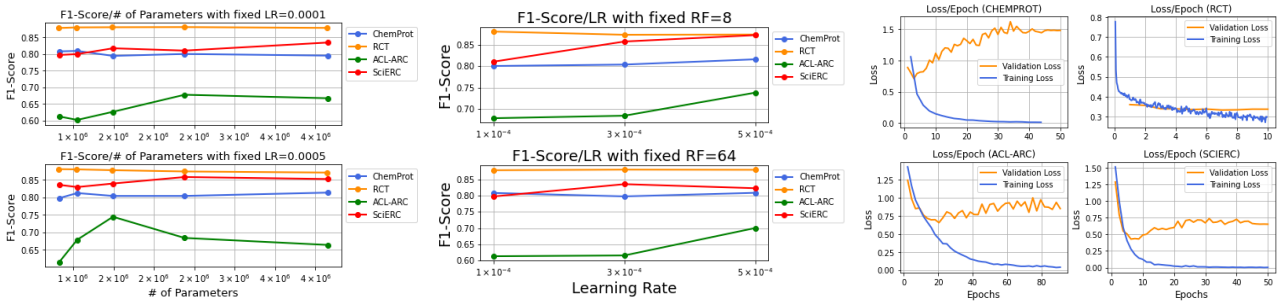
For our 2^{nd} experiment, we see that all three low resource tasks (ChemProt, ACL-ARC & SciERC) have higher accuracy when RF decreases from 64 to 8 (figure [3b]). Higher reduction factors creates smaller information bottlenecks as $RF = \frac{1}{m}$ [6]. When the bottleneck is too small, task performance suffers. RCT does slightly better at a higher reduction factors, indicating that the smaller model has sufficient representational power for this task. In contrast to low resource tasks, RCT loses accuracy at higher learning rates, likely due to catastrophic forgetting.

For our 3^{rd} experiment, we explore overfitting. The low resource tasks are clearly impacted by overfitting (figure [3c]), while the high resource RCT is not. Curiously, however, model accuracy continues to increase with loss, when typically these are inversely correlated. We attempt to mitigate overfitting in the most impacted task (ChemProt). However, warmup & weight decay only delay overfitting by a few epochs. After 50 epochs, the loss and accuracy are the same with or without these parameters. More specifically, ChemProt has a loss of 1.475 & accuracy of 81.6 without warmup or weight decay. With warmup steps of 10000 & 15000, we receive a loss of 1.431, 1.315 with accuracies of 81.05 & 80.1 respectively. With weight decay of 0.01 & 0.05, we receive a loss of 1.575 and 1.428 with accuracy of 80.51 and 82.07 respectively. This is a slight improvement for a weight decay of 0.05, but the model still overfits. Overall, we see little benefit to these techniques and do not further explore them.

Dataset	RoBERTa	F1	LR	RF
ChemProt	$81.9_{1.0}$	$80.91_{0.7}$	5e-4	8
RCT	$87.20_{0.1}$	$87.27_{0.09}$	3e-4	64
ACL-ARC	$63.0_{5.8}$	$67.0_{2.6}$	1e-4	4
SciERC	$77.3_{1.9}$	$81.8_{1.2}$	5e-4	4

Table 2: Average F1 scores over 5 runs with the suffix indicating the standard deviation (Text Classification before pre-training)

Given these observations, our hyper-tuned model scores



(a) F1 score vs # of Parameters

(b) F1 score vs Learning rate

(c) Loss Curves (Biomed & CS)

Figure 3: F1 score vs # of Parameters & Learning curves

are in table [2]. Each F1 score represents the mean and standard deviation of five runs. Other than ChemProt, all adapter models outperform the baseline from Gururangan et al. [5]. So adapters can beat fine-tuning without DAPT or TAPT. The ChemProt adapter is the exception, though it still is within 1% of baseline accuracy. Adapter improvements are most dramatic with the CS domain tasks ACL-ARC and SciERC. The number of parameters for all models are in appendix section (table [6]). One reason for lower accuracies on the baseline is because of the need to tune 100% of the parameters, which can cause catastrophic forgetting. Adapter models train fewer parameters, which retain most of the learnt information in the frozen base model.

4.4. Classification after TAPT & DAPT

For this and the next subsections, We perform the same experiments as for baseline classification, but show plots for only the 1^{st} experiment. The analysis for the 2^{nd} and 3^{rd} experiment are similar to the previous sections.

Results & Analysis (TAPT) Similar to the baseline configuration, the high resource task (RCT) performs best with higher reduction factors (figure [4a]), while low resource tasks (ACL-ARC, SciERC and ChemProt) perform best with smaller reduction factors. SciERC needs a higher learning rate compared to ACL-ARC. These align with the observations and reasons mentioned previously. The optimal learning rate for ChemProt is 6e-4, which is even higher than the other low resource tasks.

Table [3] lists the optimal LR and RF for each task, as well as the resulting F1 score. In contrast to training classification adapters directly on $RoBERTa_{BASE}$, accuracy reduces when training the classification adapter on top of a pre-trained task MLM adapter. We hypothesize that this is caused by the increased depth from stacking adapters. Each stacked adapter adds an additional adapter layer between all 12 layers of the $RoBERTa_{BASE}$ model, causing the network to grow in depth. Together with the information bottleneck introduced by each adapter, this makes the overall network deeper and narrower, increasing the difficulty of training. Such effects could be mitigated by additional residual connections or layer norms, but the adapter architecture does not currently support this. Finally, at a fraction of the original number of parameters, adapters simply have less representational power. The information bottleneck introduced by the MLM adapter reduces the signals available for the downstream classification adapter.

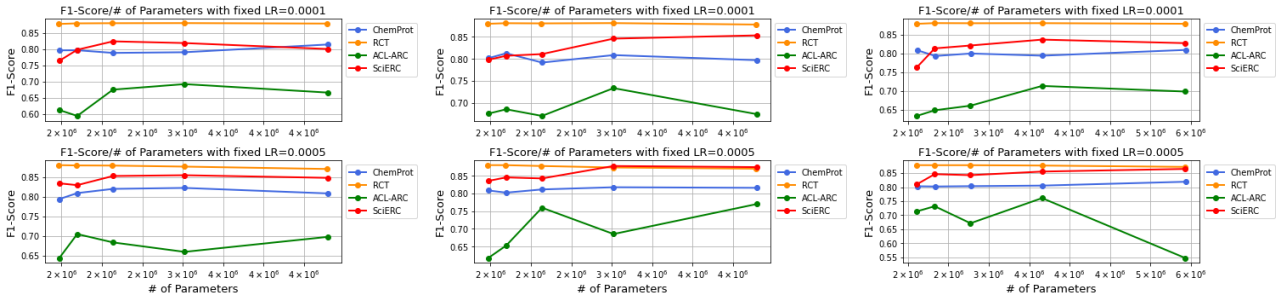
Dataset	Baseline	F1	LR	RF
ChemProt	$82.6_{0.4}$	$80.14_{0.5}$	6e-4	8
RCT	$87.7_{0.1}$	$87.25_{0.06}$	1e-4	8
ACL-ARC	$67.4_{1.8}$	$66.3_{2.4}$	1e-4	4
SciERC	$79.3_{1.5}$	$81.5_{1.0}$	5e-4	4

Table 3: Average F1 scores over 5 runs with the suffix indicating the standard deviation (Task Adaptive Pre-training)

Results & Analysis (DAPT) As shown in figure [4b], RCT accuracy is highest with higher RF, but only degrades slightly as RF decreases. As with other adapter configurations, SciERC and ChemProt do better with a higher learning rate and a smaller reduction factor. The low resource ACL-ARC goes against this trend, performing better with a higher learning rate. But this result is not robust due to the high variance in accuracy for this small dataset. Due to this variance, we favor hyperparameters that provide consistently high accuracy. The resulting hyperparameters and accuracy scores are shown in table [4]. As shown in table [4], DAPT performs similarly to TAPT. In other words, adding a domain MLM adapter before the classification adapter does not improve accuracy, and sometimes reduces it. In contrast, Gururangan et al. [5] see more dramatic improvements with DAPT when fine-tuning the full $RoBERTa_{BASE}$ model on the CS domain. In addition to the challenges of stacking adapters discussed above, the

Dataset	Baseline	F1	LR	RF
ChemProt	$84.2_{0.2}$	$80.90_{0.7}$	6e-4	8
RCT	$87.6_{0.1}$	$87.28_{0.04}$	3e-4	32
ACL-ARC	$75.4_{2.5}$	$65.8_{4.8}$	1e-4	4
SciERC	$80.8_{1.5}$	$81.3_{0.8}$	5e-4	4

Table 4: Average F1 scores over 5 runs with the suffix indicating the standard deviation (Domain Adaptive Pre-training)



(a) F1 score vs # of Parameters (Task) (b) F1 score # of Parameters (Domain) (c) F1 score vs # of Params (Domain + Task)

Figure 4: F1 score vs # of Parameters

representational power of an adapter is not enough to absorb the relevant information from large corpora such as S2ORC [12]. We see this in figure [2], where the loss is worse for adapters compared to full fine-tuning.

4.5. Classification after DAPT + TAPT (Results & Analysis)

In figure [4c], we show the accuracy versus number of parameters. Consistent with other adapter configurations, RCT performs best with a higher reduction factor. ChemProt and SciERC do better with a smaller reduction factor and higher learning rate. ACL-ARC does better with a lower learning rate and smaller reduction factor. Similar to previous experiments, low-resource datasets perform better with smaller reduction factors. Though low-resource datasets overfit with large number of parameters, they don't perform well with a small number of parameters due to lack of sufficient training data. The resulting hyperparameters and accuracy scores are shown in table [5].

Dataset	Baseline	F1	LR	RF
ChemProt	$84.4_{0.4}$	$80.3_{0.6}$	6e-4	4
RCT	$87.8_{0.1}$	$87.24_{0.02}$	1e-4	4
ACL-ARC	$75.6_{3.8}$	$68.4_{4.9}$	1e-4	4
SciERC	$81.3_{1.8}$	$81.5_{1.2}$	5e-4	4

Table 5: Average F1 scores over 5 runs with the suffix indicating the standard deviation (Domain + Task Adaptive Pre-training)

As shown in table [5], the fusion of Domain and Task pre-trained adapters do not substantially improve accuracy, with the exception of ALC-ARC. This shows that even though TAPT and DAPT each reduced the accuracy of this task over a single unstacked adapter, fusing these MLM adapters together can provide enough useful signal to improve performance. This benefit was seen on only one task, due to the same challenges discussed above related to adapter stacking. Full fine-tuning, in contrast, can update all the original parameters without increasing the model depth.

5. Challenges

The unfiltered, uncompressed S2ORC [12] was over 1TB, which is a challenge for our limited disk and GPU resources. As a result, we limit the filtered CS and Biomed domains to 10M samples. Training the RCT classifier is also challenging. While much smaller at 20K samples, the full RCT [2] is trained with 10 epochs (instead of just 1 for each domain). This is 2M samples for every combination of adapter configuration and hyperparameter setting that we experiment on for task classification. Another challenge is overfitting on low resource datasets. We try various methods to reduce overfitting but the final trained models remain suboptimal. Based on the best validation accuracy manually found, this does not fundamentally change our analysis.

6. Future work & Improvements

Since DAPT proved to be the most impactful pre-training [5], a hybrid approach could be pursued where the benefits of full fine-tuning DAPT is combined with the success of a single transformer adapter.

7. Conclusion

Based on our experiments, transformer adapters improve classification accuracy compared to the baseline $RoBERTa_{BASE}$ model fine-tuned directly on the downstream task. However, training a classification adapter on top of domain or task MLMs provide little benefit. We hypothesize several reasons: 1) stacking adapters creates a double information bottleneck that actually reduces the training performance of the classification adapter; 2) stacking adapters makes the model deeper and narrower, which makes training more difficult; 3) the number of parameters in a typical adapter are an order of magnitude less than the original model, and as a result simply lack the representational power required to match the performance of full fine-tuning.

8. Team Contribution:

Please refer to table [8] for team contributions.

References

[1] Huggingface Adapters. Huggingface adapters documentaiton, 2021. https://adapterhub.ml/. 2

[2] Franck Dernoncourt and Ji Young Lee. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts, 2017. 1, 2, 6

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. BERT architecture. 1, 2

[4] DontStopPretraining. Dont stop pretraining repo, 2020. https://github.com/allenai/dont-stop-pretraining. 2

[5] Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks, 2020. Baseline (RoBERTa with pre-training). 1, 2, 3, 4, 5, 6

[6] Neil Houlsby, Andrei Giurgiu, Stanis Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019. Transformer Adapters (Paper 1). 1, 2, 3, 4

[7] Huggingface. Huggingface documentaiton, 2021. https://huggingface.co/. 2

[8] David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Dan Jurafsky. Measuring the evolution of a scientific field through citation frames., 2018. 1, 2

[9] Seungwon Kim, Alex Shum, Nathan Susanj, and Jonathan Hilgart. Revisiting pretraining with adapters, 2021. Revisiting Pretraining with Adapters. 1, 2

[10] Jens Kringelum, Sonny Kim Kjærulff, Søren Brunak, Ole Lund, Tudor I. Oprea, and Olivier Taboureau. Chemprot-3.0: a global chemical biology diseases mapping, 2016. 1, 2

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. RoBERTa architecture. 1, 2, 3

[12] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online, July 2020. Association for Computational Linguistics. 1, 6

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. AdamW. 3

[14] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction., 2018. 1, 2

[15] Jonas Pfeiffer, Aishwarya Kamath, Andreas Ruckle, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning, 2021. AdapterFusion (Transformer Adapters - Paper 2). 1, 2, 3

[16] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskeve. Improving language understanding by generative pre-training, 2018. Introduction of Pre-training and fine-tuning. 1, 2

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Łukasz Kaiser. Attention is all you need, 2017. Beginning of Transformers. 1, 2, 3

A. Appendix

A.1. Number of parameters for our chosen models

Dataset	Method	RF	# of params
ChemProt	Baseline	8	2,380,429
RCT	Baseline	64	831,133
ACL-ARC, SciERC	Baseline	4	4,151,053
ChemProt, RCT	Task	8	302,822
RCT	Domain	32	1,694,854
ACL-ARC, SciERC	Task	4	4,793,446
ChemProt, RCT, ACL-ARC, SCI-ERC	Domain + Task	4	5,435,839

Table 6: Number of parameters for all models

A.2. Domain loss on test set

Domain Corpus	Full fine-tuning	Adapter
Computer Science	1.558	1.898
Biomed	1.332	1.724

Table 7: Domain loss on test set

A.3. Project Code Repository

We zipped our code and put it on google drive for you to access. Click to view code

A.4. Work Division

See table [8].

A.5. Text classification

A.5.1 Accuracy versus number of parameters

As you can see from the figure [5], as the learning rate increase slightly from 0.0001 [5], the accuracy improves for low-resource datasets as the number of parameters increases. For high resource dataset like RCT, the accuracy degrades as the learning rate and the number of parameters increase. This aligns with our observations and selections in the paper.

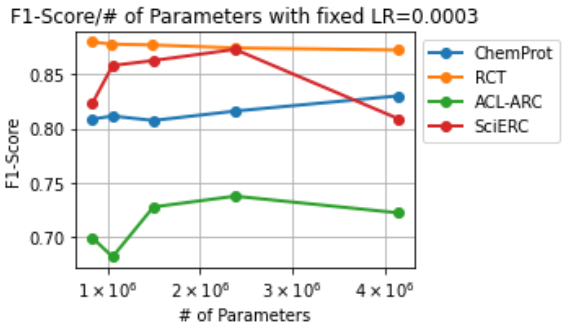


Figure 5: Accuracy/Number of parameters for learning rate 0.0003

A.5.2 Overfitting in low-resource datasets

As you can see in the plots [6], warmup and weight decay didn't really help prevent overfitting in our low-resource datasets.

Student Name	Contributed Aspects	Details
Sai Guttikonda	Implementation and Analysis	Wrote code for base, task, domain + task configurations for ChemProt and RCT tasks. Wrote code for post processing data and making plots [see code under tasks/biomed] and performed analysis on all 4 tasks for baseline, task and domain, task + domain configuration sections [see sections under experiments]. Contributed to the report for the Abstract, Introduction, Approach, Experiments and other sections.
Brandon Stewart	Implementation and Analysis	Preprocessed domain datasets. Wrote initial domain MLM adapters models, trained and ran experiments for domain and task MLM adapters. Wrote initial prototype for ACL-ARC task classification adapter on top of domain adapter. Proposed initial project plan to organize separate work streams. Contributed to the report for Introduction, Approach, MLM training, and various other sections.
Carlo Cochran	Implementation and Analysis	Setup of AWS computing resource. Wrote code for SciERC and ACL-ARC classification tasks and MLM adapter. Run training on SciERC and ACL-ARC classification tasks and MLM adapter. Contributed to plot generation and final report formatting.

Table 8: Contributions of team members.

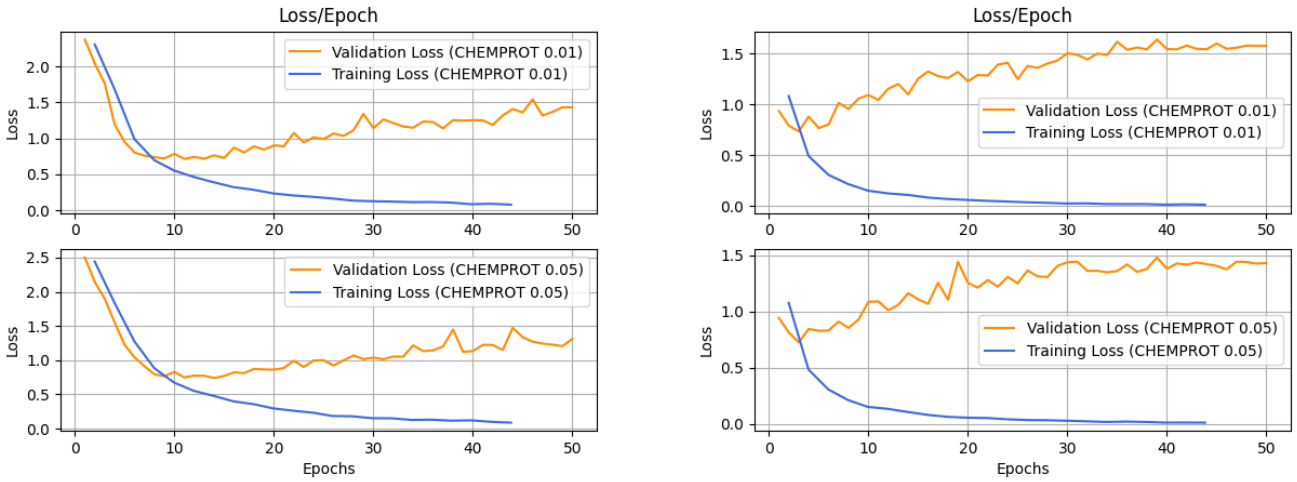


Figure 6: Loss curves with warmup and weight decay for ChemProt