AdapterFusion: Non-Destructive Task Composition for Transfer Learning

Jonas Pfeiffer[1], **Aishwarya Kamath**[2], **Andreas Rücklé**[1], **Kyunghyun Cho**[2,3], **Iryna Gurevych**[1]
[1]Ubiquitous Knowledge Processing Lab (UKP Lab), Technical University of Darmstadt
[2]New York University [3]CIFAR Associate Fellow
pfeiffer@ukp.tu-darmstadt.de

Abstract

Sequential fine-tuning and multi-task learning are methods aiming to incorporate knowledge from multiple tasks; however, they suffer from catastrophic forgetting and difficulties in dataset balancing. To address these shortcomings, we propose *AdapterFusion*, a new two stage learning algorithm that leverages knowledge from multiple tasks. First, in the *knowledge extraction* stage we learn task specific parameters called *adapters*, that encapsulate the task-specific information. We then combine the adapters in a separate *knowledge composition* step. We show that by separating the two stages, i.e., knowledge extraction and knowledge composition, the classifier can effectively exploit the representations learned from multiple tasks in a non-destructive manner. We empirically evaluate AdapterFusion on 16 diverse NLU tasks, and find that it effectively combines various types of knowledge at different layers of the model. We show that our approach outperforms traditional strategies such as full fine-tuning as well as multi-task learning. Our code and adapters are available at AdapterHub.ml.

1 Introduction

The most commonly used method for solving NLU tasks is to leverage pretrained models, with the dominant architecture being a transformer (Vaswani et al., 2017), typically trained with a language modelling objective (Devlin et al., 2019; Radford et al., 2018; Liu et al., 2019b). Transfer to a task of interest is achieved by fine-tuning all the weights of the pretrained model on that *single task*, often yielding state-of-the-art results (Zhang and Yang, 2017; Ruder, 2017; Howard and Ruder, 2018; Peters et al., 2019). However, each task of interest requires all the parameters of the network to be fine-tuned, which results in a specialized model for each task.

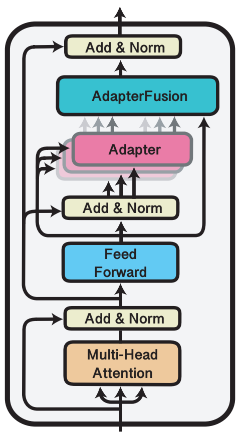


Figure 1: AdapterFusion architecture inside a transformer (Vaswani et al., 2017). The AdapterFusion component takes as input the representations of multiple adapters trained on different tasks and learns a parameterized mixer of the encoded information.

There are two approaches for sharing information across multiple tasks. The first consists of starting from the pretrained language model and sequentially fine-tuning on each of the tasks one by one (Phang et al., 2018). However, as we subsequently fine-tune the model weights on new tasks, the problem of catastrophic forgetting (McCloskey and Cohen, 1989; French, 1999) can arise, which results in loss of knowledge already learned from all previous tasks. This, together with the non-trivial decision of the order of tasks in which to fine-tune the model, hinders the effective transfer of knowledge. Multi-task learning (Caruana, 1997; Zhang and Yang, 2017; Liu et al., 2019a) is another approach for sharing information across multiple tasks. This involves fine-tuning the weights of a pretrained language model using a weighted sum of the objective function of each target task simultaneously. Using this approach, the network captures the common structure underlying all the target tasks. However, multi-task learning requires simultaneous access to all tasks during training. Adding new tasks thus requires complete joint retraining. Further, it is difficult to balance multiple tasks and train a model that solves each task equally well. As has been shown in Lee et al. (2017), these models often overfit on low resource tasks and underfit on high resource tasks. This makes it difficult to effectively transfer knowledge across tasks with all the tasks being solved equally well (Pfeiffer et al., 2020b), thus considerably limiting the applicability of multi-task learning in many scenarios.

Recently, *adapters* (Rebuffi et al., 2017; Houlsby et al., 2019) have emerged as an alternative training strategy. Adapters do not require fine-tuning of all parameters of the pretrained model, and instead introduce a small number of task specific parameters — while keeping the underlying pretrained language model fixed. Thus, we can separately and simultaneously train adapters for multiple tasks, which all share the same underlying pretrained parameters. However, to date, there exists no method for using *multiple* adapters to maximize the transfer of knowledge across tasks without suffering from the same problems as sequential fine-tuning and multi-task learning. For instance, Stickland and Murray (2019) propose a multi-task approach for training adapters, which still suffers from the difficulty of balancing the various target tasks and requiring simultaneous access to all target tasks.

In this paper we address these limitations and propose a new variant of adapters called *AdapterFusion*. We further propose a novel two stage learning algorithm that allows us to effectively share knowledge across multiple tasks while avoiding the issues of catastrophic forgetting and balancing of different tasks. Our AdapterFusion architecture, illustrated in Figure 1, has two components. The first component is an adapter trained on a task without changing the weights of the underlying language model. The second component — our novel Fusion layer — combines the representations from several such task adapters in order to improve the performance on the target task.

Contributions Our main contributions are: (1) We introduce a novel two-stage transfer learning strategy, termed *AdapterFusion*, which combines the knowledge from multiple source tasks to perform better on a target task. (2) We empirically evaluate our proposed approach on a set of 16 diverse NLU tasks such as sentiment analysis, commonsense reasoning, paraphrase detection, and recognizing textual entailment. (3) We compare our approach with Stickland and Murray (2019) where adapters are trained for all tasks in a multi-task manner, finding that AdapterFusion is able to improve this method, even though the model has simultaneous access to all tasks during pretraining. (4) We show that our proposed approach outperforms fully fine-tuning the transformer model on a single target task. Our approach additionally outperforms adapter based models trained both in a Single-Task, as well as Multi-Task setup.

The code of this work is integrated into the AdapterHub.ml (Pfeiffer et al., 2020a).

2 Background

In this section, we formalize our goal of transfer learning (Pan and Yang, 2010; Torrey and Shavlik, 2010; Ruder, 2019), highlight its key challenges, and provide a brief overview of common methods that can be used to address them. This is followed by an introduction to *adapters* (Rebuffi et al., 2017) and a brief formalism of the two approaches to training adapters.

Task Definition. We are given a model that is pretrained on a task with training data D_0 and a loss function L_0. The weights Θ_0 of this model are learned as follows:

$$\begin{aligned} D_0 &:= \text{Large corpus of unlabelled text} \\ L_0 &:= \text{Masked language modelling loss} \\ \Theta_0 &\leftarrow \underset{\Theta}{\operatorname{argmin}}\ L_0(D_0; \Theta) \end{aligned}$$

In the remainder of this paper, we refer to this pretrained model by the tuple (D_0, L_0).

We define C as the set of N classification tasks having labelled data of varying sizes and different loss functions:

$$C = \{(D_1, L_1), \ldots, (D_N, L_N)\}$$

The aim is to be able to leverage a set of N tasks to improve on a target task m with $C_m = (D_m, L_m)$. In this work we focus on the setting where $m \in \{1, \ldots, N\}$.

Desiderata. We wish to learn a parameterization Θ_m that is defined as follows:

$$\Theta_m \leftarrow \underset{\Theta'}{\operatorname{argmin}}\ L_m(D_m; \Theta')$$

where Θ' is expected to have encapsulated relevant information from all the N tasks. The target model for task m is initialized with Θ' for which we learn the optimal parameters Θ_m through minimizing the task's loss on its training data.

2.1 Current Approaches to Transfer Learning

There are two predominant approaches to achieve sharing of information from one task to another.

2.1.1 Sequential Fine-Tuning

This involves sequentially updating all the weights of the model on each task. For a set of N tasks, the order of fine-tuning is defined and at each step the model is initialized with the parameters learned through the previous step. However, this approach does not perform well beyond two sequential tasks (Phang et al., 2018; Pruksachatkun et al., 2020) due to catastrophic forgetting.

2.1.2 Multi-Task Learning (MTL)

All tasks are trained simultaneously with the aim of learning a shared representation that will enable the model to generalize better on each task (Caruana, 1997; Collobert and Weston, 2008; Nam et al., 2014; Liu et al., 2016, 2017; Zhang and Yang, 2017; Ruder, 2017; Ruder et al., 2019; Sanh et al., 2019; Pfeiffer et al., 2020b, *inter alia*).

$$\Theta_{0\rightarrow\{1,...,N\}} \leftarrow \underset{\Theta}{\operatorname{argmin}} \left(\sum_{n=1}^{N} L_n(D_n; \Theta_0) \right)$$

Where $\Theta_{0\rightarrow\{1,...,N\}}$ indicates that we start with Θ_0 and fine-tune on a set of tasks $\{1, ..., N\}$.

However, MTL requires simultaneous access to all tasks, making it difficult to add more tasks on the fly. As the different tasks have varying sizes as well as loss functions, effectively combining them during training is very challenging and requires heuristic approaches as proposed in Stickland and Murray (2019).

2.2 Adapters

While the predominant methodology for transfer learning is to fine-tune all weights of the pretrained model, *adapters* (Houlsby et al., 2019) have recently been introduced as an alternative approach with applications in domain transfer (Rücklé et al., 2020b), machine translation (Bapna and Firat, 2019; Philip et al., 2020) transfer learning (Stickland and Murray, 2019; Wang et al., 2020; Lauscher et al., 2020), and cross-lingual transfer (Pfeiffer et al., 2020c,d; Üstün et al., 2020; Vidoni et al., 2020). Adapters share a large set of parameters Θ across all tasks and introduce a small number of task-specific parameters Φ_n. While Θ represents the weights of a pretrained model (e.g., a transformer), the parameters Φ_n, where $n \in \{1, \dots, N\}$, are used to encode task-specific representations in intermediate layers of the shared model. Current work on adapters focuses either on training adapters for each task separately (Houlsby et al., 2019; Bapna and Firat, 2019; Pfeiffer et al., 2020a) or training them in a multi-task setting to leverage shared representations (Stickland and Murray, 2019). We discuss both variants below.

2.2.1 Single-Task Adapters (ST-A)

For each of the N tasks, the model is initialized with parameters Θ_0. In addition, a set of new and randomly initialized adapter parameters Φ_n are introduced.

The parameters Θ_0 are fixed and only the parameters Φ_n are trained. This makes it possible to efficiently parallelize the training of adapters for all N tasks, and store the corresponding knowledge in designated parts of the model. The objective for each task $n \in \{1, \dots, N\}$ is of the form:

$$\Phi_n \leftarrow \underset{\Phi}{\operatorname{argmin}}\ L_n(D_n; \Theta_0, \Phi)$$

For common adapter architectures, Φ contains considerably fewer parameters than Θ, e.g., only 3.6% of the parameters of the pretrained model in Houlsby et al. (2019).

2.2.2 Multi-Task Adapters (MT-A)

Stickland and Murray (2019) propose to train adapters for N tasks in parallel with a multi-task objective. The underlying parameters Θ_0 are fine-tuned along with the task-specific parameters in Φ_n. The training objective can be defined as:

$$\boldsymbol{\Theta} \leftarrow \underset{\Theta,\Phi}{\operatorname{argmin}} \left(\sum_{n=1}^{N} L_n(D_n; \Theta_0, \Phi_n) \right)$$

where

$$\boldsymbol{\Theta} = \Theta_{0\rightarrow\{1,...,N\}}, \Phi_1, \dots, \Phi_N.$$

2.2.3 Adapters in Practice

Introducing new adapter parameters in different layers of an otherwise fixed pretrained model has been shown to perform on-par with, or only slightly below, full model fine-tuning (Houlsby et al., 2019; Stickland and Murray, 2019; Pfeiffer et al., 2020a).

For NLP tasks, adapters have been introduced for the transformer architecture (Vaswani et al., 2017). At each transformer layer l, a set of adapter parameters Φ_l is introduced. The placement and architecture of adapter parameters Φ within a pretrained model is non-trivial. Houlsby et al. (2019) experiment with different architectures, finding that a two-layer feed-foward neural network with a bottleneck works well. They place two of these components within one layer, one after the multi-head attention (further referred to as *bottom*) and one after the feed-forward layers of the transformer (further referred to as *top*).[1] Bapna and Firat (2019) and Stickland and Murray (2019) only introduce one of these components at the *top* position, however, Bapna and Firat (2019) include an additional *layer norm* (Ba et al., 2016).

Adapters trained in both single-task (ST-A) or multi-task (MT-A) setups have learned the idiosyncratic knowledge of the respective tasks' training data, encapsulated in their designated parameters. This results in a compression of information, which requires less space to store task-specific knowledge. However, the distinct weights of adapters prevent a downstream task from being able to use multiple sources of extracted information. In the next section we describe our two stage algorithm which tackles the sharing of information stored in adapters trained on different tasks.

3 AdapterFusion

Adapters avoid catastrophic forgetting by introducing task-specific parameters; however, current adapter approaches do not allow sharing of information between tasks. To mitigate this we propose AdapterFusion.

3.1 Learning algorithm

In the first stage of our learning algorithm, we train either ST-A or MT-A for each of the N tasks.

In the second stage, we then combine the set of N adapters by using AdapterFusion. While fixing both the parameters Θ as well as all adapters Φ, we introduce parameters Ψ that learn to combine the N task adapters to solve the target task.

$$\Psi_m \leftarrow \underset{\Psi}{\operatorname{argmin}}\ L_m(D_m; \Theta, \Phi_1, \ldots, \Phi_N, \Psi)$$

Ψ_m are the newly learned AdapterFusion parameters for task m. Θ refers to Θ_0 in the ST-A setting or $\Theta_{0\to\{1,\ldots,N,m\}}$ in the MT-A setup. In our experiments we focus on the setting where $m \in \{1, ..., N\}$, which means that the training dataset of m is used twice: *once* for training the adapters Φ_m and *again* for training Fusion parameters Ψ_m, which learn to compose the information stored in the N task adapters.

[1] We illustrate these placements in Appendix Figure 5 (left).

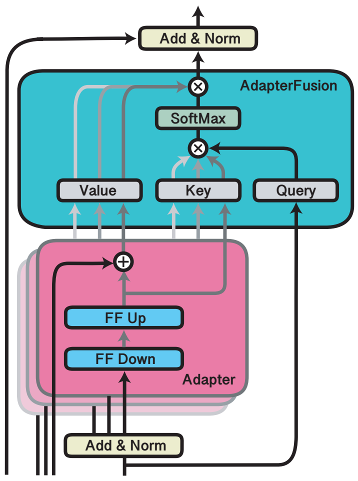


Figure 2: Our AdapterFusion architecture. This includes learnable weights *Query*, *Key*, and *Value*. *Query* takes as input the output of the pretrained transformer weights. Both *Key* and *Value* take as input the output of the respective adapters. The dot product of the *query* with all the *keys* is passed into a softmax function, which learns to weight the adapters with respect to the context.

By separating the two stages — knowledge extraction in the adapters, and knowledge composition with AdapterFusion — we address the issues of catastrophic forgetting, interference between tasks and training instabilities.

3.2 Components

AdapterFusion learns to compose the N task adapters Φ_n and the shared pretrained model Θ, by introducing a new set of weights Ψ. These parameters learn to combine the adapters as a dynamic function of the target task data.

As illustrated in Figure 2, we define the AdapterFusion parameters Ψ to consist of *Key, Value* and *Query* matrices at each layer l, denoted by $\mathbf{K}_l$, $\mathbf{V}_l$ and $\mathbf{Q}_l$ respectively. At *each layer* l of the transformer and *each time-step* t, the output of the feed-forward sub-layer of layer l is taken as the query vector. The output of each adapter $\mathbf{z}_{l,t}$ is used as input to both the *value* and *key* transformations. Similar to attention (Bahdanau et al., 2015; Vaswani et al., 2017), we learn a contextual activation of each adapter n using

$$\begin{aligned}
\mathbf{s}_{l,t} &= \text{softmax}(\mathbf{h}_{l,t}^\top \mathbf{Q}_l \otimes \mathbf{z}_{l,t,n}^\top \mathbf{K}_l), n \in \{1, ..., N\} \\
\mathbf{z}'_{l,t,n} &= \mathbf{z}_{l,t,n}^\top \mathbf{V}_l, n \in \{1, ..., N\} \\
\mathbf{Z}'_{l,t} &= [\mathbf{z}'_{l,t,0}, ..., \mathbf{z}'_{l,t,N}] \\
\mathbf{o}_{l,t} &= \mathbf{s}_{l,t}^\top \mathbf{Z}'_{l,t}
\end{aligned}$$

Where $\otimes$ represents the dot product and $[\cdot, \cdot]$ indicates the concatenation of vectors.

Given the context, AdapterFusion learns a parameterized mixer of the available trained adapters. It learns to identify and activate the most useful adapter for a given input.

4 Experiments

In this section we evaluate how effective AdapterFusion is in overcoming the issues faced by other transfer learning methods. We provide a brief description of the 16 diverse datasets that we use for our study, each of which uses accuracy as the scoring metric.

4.1 Experimental Setup

In order to investigate our model's ability to overcome catastrophic forgetting, we compare Fusion using ST-A to only the ST-A for the task. We also compare Fusion using ST-A to MT-A for the task to test whether our two-stage procedure alleviates the problems of interference between tasks. Finally, our experiments to compare MT-A with and without Fusion let us investigate the versatility of our approach. Gains in this setting would show that AdapterFusion is useful even when the base adapters have already been trained jointly.

In all experiments, we use BERT-base-uncased (Devlin et al., 2019) as the pretrained language model. We train ST-A, described in Appendix A.2 and illustrated in Figure 5, for all datasets described in §4.2. We train them with reduction factors[2] $\{2, 16, 64\}$ and learning rate 0.0001 with AdamW and a linear learning rate decay. We train for a maximum of 30 epochs with early stopping. We follow the setup used in Stickland and Murray (2019) for training the MT-A. We use the default hyperparameters[3], and train a MT-A model on all datasets simultaneously.

For AdapterFusion, we empirically find that a learning rate of $5e-5$ works well, and use this in all experiments.[4] We train for a maximum of 10 epochs with early stopping. While we initialize $\mathbf{Q}$ and $\mathbf{K}$ randomly, we initialize $\mathbf{V}$ with a diagonal of ones and the rest of the matrix with random weights having a small norm ($1e-6$). Multiplying the adapter output with this value matrix $\mathbf{V}$ initially adds small amounts of noise, but retains the overall representation. We continue to regularize the *Value* matrix using l_2-norm to avoid introducing additional capacity.

4.2 Tasks and Datasets

We briefly summarize the different types of *tasks* that we include in our experiments, and reference the related datasets accordingly. A detailed descriptions can be found in Appendix A.1.

Commonsense reasoning is used to gauge whether the model can perform basic reasoning skills: *Hellaswag* (Zellers et al., 2018, 2019), *Winogrande* (Sakaguchi et al., 2020), *CosmosQA* (Huang et al., 2019), *CSQA* (Talmor et al., 2019), *SocialIQA* (Sap et al., 2019). **Sentiment analysis** predicts whether a given text has a positive or negative sentiment: *IMDb* (Maas et al., 2011), *SST* (Socher et al., 2013). **Natural language inference** predicts whether one sentence entails, contradicts, or is neutral to another: *MNLI* (Williams et al., 2018), *SciTail* (Khot et al., 2018), *SICK* (Marelli et al., 2014), *RTE* (as combined by Wang et al. (2018)), *CB* (De Marneffe et al., 2019). **Sentence relatedness** captures whether two sentences include similar content: *MRPC* (Dolan and Brockett, 2005), *QQP*[5]. We also use an argument mining *Argument* (Stab et al., 2018) and reading comprehension *BoolQ* (Clark et al., 2019) dataset.

5 Results

We present results for all 16 datasets in Table 1. For reference, we also include the adapter architecture of Houlsby et al. (2019), ST-A$^{\text{Houlsby}}$, which has twice as many parameters compared to ST-A. To provide a fair comparison to Stickland and Murray (2019) we primarily experiment with BERT-base-uncased. We additionally validate our best model configurations — ST-A and Fusion with ST-A — with RoBERTa-base, for which we present our results in Appendix Table 4.

[2]A reduction factor indicates the factor by which the hidden size is reduced such that the bottle-neck size for BERT Base with factor 64 is reduced to 12 ($768/64 = 12$).

[3]We additionally test out batch sizes 16 and 32.

[4]We have experimented with learning rates $\{6e-6, 5e-5, 1e-4, 2e-4\}$

[5]data.quora.com/First-Quora-DatasetReleaseQuestion-Pairs

Dataset	Head	Full	ST-A	MT-A	F. w/ ST-A	F. w/ MT-A	ST-A$^{\text{Houlsby}}$
MNLI	54.59	84.10	**84.32**	82.49 ±0.49	84.28	83.05	84.13
QQP	76.79	**90.87**	90.59	89.47 ±0.60	90.71	90.58	90.63
SST	85.17 ±0.45	92.39 ±0.22	91.85 ±0.41	92.27 ±0.71	92.20 ±0.18	**93.00** ±0.20	92.75 ±0.37
WGrande	51.92 ±0.35	60.01 ±0.08	**61.09** ±0.11	57.70 ±1.40	60.23 ±0.31	59.32 ±0.30	59.32 ±1.33
IMDB	85.05 ±0.22	**94.05** ±0.21	93.85 ±0.07	92.56 ±0.54	93.82 ±0.39	92.66 ±0.32	93.96 ±0.22
HSwag	34.17 ±0.27	**39.25** ±0.76	38.11 ±0.14	36.47 ±0.98	37.98 ±0.01	37.36 ±0.10	38.65 ±0.25
SocIQA	50.33 ±2.50	62.05 ±0.04	62.41 ±0.11	61.21 ±0.89	**63.16** ±0.24	62.56 ±0.10	62.73 ±0.53
CosQA	50.06 ±0.51	60.28 ±0.40	60.01 ±0.02	61.25 ±0.90	60.65 ±0.55	**62.78** ±0.07	61.37 ±0.35
SciTail	85.30 ±2.44	94.32 ±0.11	93.90 ±0.16	94.53 ±0.43	94.04 ±0.23	**94.79** ±0.17	94.07 ±0.39
Argument	70.61 ±0.59	76.87 ±0.32	**77.65** ±0.34	75.70 ±0.60	**77.65** ±0.21	76.08 ±0.27	77.44 ±0.62
CSQA	41.09 ±0.27	58.88 ±0.40	58.91 ±0.57	53.30 ±2.19	59.73 ±0.54	56.73 ±0.14	**60.05** ±0.36
BoolQ	63.07 ±1.27	74.84 ±0.24	75.66 ±1.25	78.76 ±0.76	76.25 ±0.19	**79.18** ±0.45	76.02 ±1.13
MRPC	71.91 ±0.13	85.14 ±0.45	85.16 ±0.52	81.86 ±0.99	**90.29** ±0.84	84.68 ±0.32	86.66 ±0.81
SICK	76.30 ±0.71	87.30 ±0.42	86.20 ±0.00	88.61 ±1.06	87.28 ±0.99	**90.43** ±0.30	86.12 ±0.54
RTE	61.37 ±1.17	65.41 ±0.90	71.04 ±1.62	77.61 ±3.21	76.82 ±1.68	**79.96** ±0.76	69.67 ±1.96
CB	68.93 ±4.82	82.49 ±2.33	86.07 ±3.87	89.09 ±1.15	**92.14** ±0.97	89.81 ±0.99	87.50 ±4.72
Mean	64.17	75.51	76.05	75.80	**77.33**	77.06	76.32

Table 1: Mean and standard deviation results (development sets) for each of the 16 datasets and the different architectural setups. The datasets are ordered by their respective training dataset size. Dashed horizontal lines separate datasizes $\{> 40k, > 10k, > 5k\}$, respectively. Each model is initialized with BERT-base (Devlin et al., 2019) weights. **Head** indicates training only a classification head on top of fixed BERT weights. For **Full** training we fine-tune all weights of BERT. Single-Task Adapters (**ST-A**) is the training of independently trained adapters for each task, using the architecture illustrated in Figure 5. Multi-Task Adapters (**MT-A**) shows results of jointly trained adapters using the default settings of Stickland and Murray (2019). **Fusion w/ ST-A** and **Fusion w/ MT-A** show the results of AdapterFusion using the respective pre-trained Adapters. **ST-A$^{\text{Houlsby}}$** shows the results of ST-Adapters with the architecture proposed by Houlsby et al. (2019). Reported results are accuracy scores.

5.1 Adapters

Training only a prediction-head on the output of a pretrained model can also be considered an adapter. This procedure, commonly referred to as training only the *Head*, performs considerably worse than fine-tuning all weights (Howard and Ruder, 2018; Peters et al., 2019). We show that the performance of only fine-tuning the *Head* compared to *Full* fine-tuning causes on average a drop of 10 points in accuracy. This demonstrates the need for more complex adaptation approaches.

In Table 1 we show the results for MT-A and ST-A with a reduction factor 16 (see the appendix Table 3 for more results) which we find has a good trade-off between the number of newly introduced parameters and the task performance. Interestingly, the ST-A have a regularization effect on some datasets, resulting in better performance on average for certain tasks, even though a much small proportion of weights is trained. On average, we improve 0.66% by training ST-A instead of the *Full* model.

For MT-A we find that there are considerable performance drops of more than 2% for *CSQA* and *MRPC*, despite the heuristic strategies for sampling from the different datasets (Stickland and Murray, 2019). This indicates that these heuristics only partially address common problems of multi-task learning such as catastrophic interference. It also shows that learning a shared representation jointly does not guarantee the best results for all tasks. On average, however, we do see a performance increase of 0.4% using MT-A over *Full* fine-tuning on each task separately, which demonstrates that there are advantages in leveraging information from other tasks with multi-task learning.

5.2 AdapterFusion

AdapterFusion aims to improve performance on a given target task m by transferring task specific knowledge from the set of all N task adapters, where $m \in \{1, \ldots, \mathrm{N}\}$. We hypothesize that if there exists at least one task that supports the target task, AdapterFusion should lead to performance gains. If no such task exists, then the performance should remain the same.

Dependence on the size of training data. In Table 1 we notice that having access to relevant tasks considerably improves the performance for the target task when using AdapterFusion. While datasets with more than 40k training instances perform well without Fusion, smaller datasets with fewer training instances benefit more from our approach. We

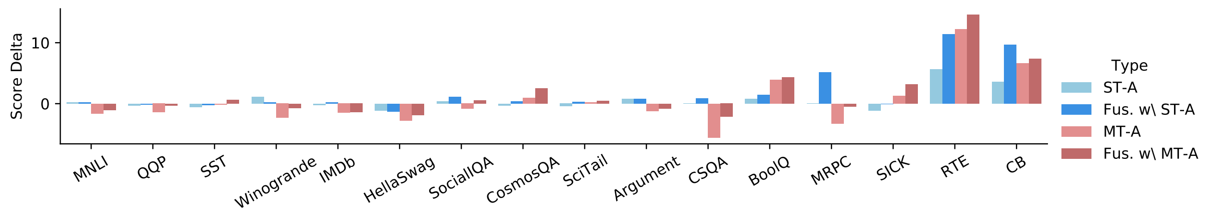


Figure 3: Relative performance difference of the two adapter architectures and the AdapterFusion models over fully fine-tuned BERT. Fusion improves over its corresponding adapters (ST-A and MT-A) for most tasks.

	Fus. w/ ST-A		**Fus. w/ MT-A**	
compared to	ST-A	MT-A	ST-A	MT-A
MNLI	→	↗	↘	↗
QQP	→	↗	→	↗
SST	↗	→	↗	↗
Winogrande	↘	↗	↘	↗
IMDB	↗	↗	↘	→
HellaSwag	→	↗	↘	↗
SocialIQA	↗	↗	→	↗
CosmosQA	↗	↘	↗	↗
SciTail	→	↗	↗	→
Argument	→	↗	↘	↗
CSQA	↗	↗	↘	↗
BoolQ	↗	↘	↗	↗
MRPC	↗	↗	↘	↗
SICK	↗	↘	↗	↗
RTE	↗	↘	↗	↗
CB	↗	↗	↗	↗
Improved	10/16	11/16	7/16	14/16

Table 2: Performance changes of AdapterFusion compared to ST-A and MT-A. Arrows indicate whether there has been an improvement ↗ (> 0.3), decrease ↘ (< -0.3), or whether the results have stayed the same → $[-0.3, 0.3]$.

observe particularly large performance gains for datasets with less than 5k training instances. For example, Fusion with ST-A achieves substantial improvements of 6.5 % for *RTE* and 5.64 % for *MRPC*. In addition, we also see performance gains for moderately sized datasets such as the commonsense tasks *CosmosQA* and *CSQA*. Fusion with MT-A achieves smaller improvements, as the model already includes a shared set of parameters. However, we do see performance gains for *SICK*, *SocialIQA*, *Winogrande* and *MRPC*. On average, we observe improvements of 1.27% and 1.25% when using Fusion with ST-A and MT-A, respectively.

Mitigating catastrophic interference. In order to identify whether our approach is able to mitigate problems faced by multi-task learning, we present the performance differences of adapters and AdapterFusion compared to the fully fine-tuned model in Figure 3. In Table 2, we compare AdapterFusion to ST-A and MT-A. The arrows indicate whether there is an improvement ↗, decrease ↘, or if the the results remain the same →. We compare the performance of both, Fusion with ST-A and Fusion with MT-A, to ST-A and MT-A. We summarize our four most important findings below.

(1) In the case of Fusion with ST-A, for $15/16$ tasks, the performance remains the same or improves as compared to the task's pretrained adapter. For $10/16$ tasks we see performance gains. This shows that having access to adapters from other tasks is beneficial and in the majority of cases leads to better results on the target task. **(2)** We find that for $11/16$ tasks, Fusion with ST-A improves the performance compared to MT-A. This demonstrates the ability of Fusion with ST-A to share information between tasks while avoiding the interference that multi-task training suffers from. **(3)** For only $7/16$ tasks, we see an improvement of Fusion with MT-A over the ST-A. Training of MT-A in the first stage of our algorithm suffers from all the problems of multi-task learning and results in less effective adapters than our ST-A on average. Fusion helps bridge some of this gap but is not able to mitigate the entire performance drop. **(4)** In the case of AdapterFusion with MT-A, we see that the performances on *all 16 tasks* improves or stays the same. This demonstrates that AdapterFusion can successfully combine the specific adapter weights, even if the adapters were trained in a multi-task setting, confirming that our method is versatile.

Summary. Our findings demonstrate that Fusion with ST-A is the most promising approach to sharing information across tasks. Our approach allows us to train adapters in parallel and it requires no heuristic sampling strategies to deal with imbalanced datasets. It also allows researchers to easily add more tasks as they become available, without requiring complete model retraining.

While Fusion with MT-A does provide gains over simply using MT-A, the effort required to train

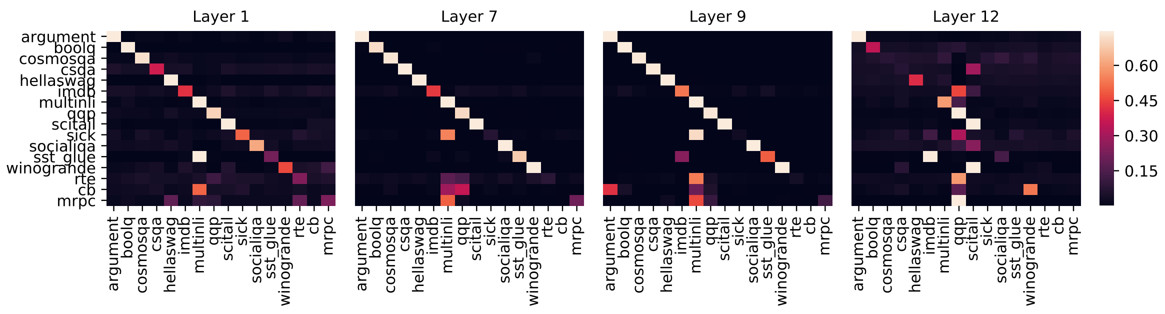


Figure 4: AdapterFusion activations of pretrained **ST-Adapters**. Rows indicate the target task m, columns indicate adapters n. We assume that the softmax activation for $\Phi_{n,l}$ is high if the information of adapter n is useful for task m. For our analysis, we calculate the softmax activation for each adapter $\Phi_{n,l}$, where $n \in \{1, \ldots, N\}$, and average over all activations within the same layer l calculated over all instances in the development set.

these in a multi-task setting followed by the Fusion step are not warranted by the limited gains in performance. On the other hand, we find that Fusion with ST-A is an efficient and versatile approach to transfer learning.

6 Analysis of Fusion Activation

We analyze the weighting patterns that are learned by AdapterFusion to better understand which tasks impact the model predictions, and whether there exist differences across BERT layers.

We plot the results for layers 1, 7, 9, and 12 and ST-A in Figure 4 (see Appendix Figure 6 for the remaining layers). We find that tasks which do not benefit from AdapterFusion tend to more strongly activate their own adapter at every layer (e.g. *Argument, HellaSwag, MNLI, QQP, SciTail*). This confirms that AdapterFusion only extracts information from adapters if they are beneficial for the target task m. We further find that *MNLI* is a useful intermediate task that benefits a large number of target tasks, e.g. *BoolQ, SICK, CSQA, SST-2, CB, MRPC, RTE*, which is in line with previous work (Phang et al., 2018; Conneau and Kiela, 2018; Reimers and Gurevych, 2019). Similarly, *QQP* is utilized by a large number of tasks, e.g. *SICK, IMDB, RTE, CB, MRPC, SST-2*. Most importantly, tasks with small datasets such as *CB, RTE,* and *MRPC* often strongly rely on adapters trained on large datasets such as *MNLI* and *QQP*.

Interestingly, we find that the activations in layer 12 are considerably more distributed across multiple tasks than adapters in earlier layers. The potential reason for this is that the last adapters are not encapsulated between frozen pretrained layers, and can thus be considered as an extension of the prediction head. The representations of the adapters in the 12th layer might thus not be as comparable, resulting in more distributed activations. This is in line with Pfeiffer et al. (2020d) who are able to improve zero-shot cross-lingual performance considerably by dropping the adapters in the last layer.

7 Contemporary Work

In contemporaneous work, other approaches for parameter efficient fine-tuning have been proposed. Guo et al. (2020) train sparse "diff" vectors which are applied on top of pretrained frozen parameter vectors. Ravfogel and Goldberg (2021) only finetune bias terms of the pretrained language models, achieving similar results as full model finetuning. Li and Liang (2021) propose prefix-tuning for natural language generation tasks. Here, continuous task-specific vectors are trained while the remaining model is kept frozen. These alternative, parameter-efficient fine-tuning strategies all encapsulate the idiosyncratic task-specific information in designated parameters, creating the potential for new composition approaches of multiple tasks.

Rücklé et al. (2020a) analyse the training and inference efficiency of adapters and AdapterFusion. For AdapterFusion, they find that adding more tasks to the set of adapters results in a linear increase of computational cost, both for training and inference. They further propose approaches to mitigate this overhead.

8 Conclusion and Outlook

8.1 Conclusion

We propose a novel approach to transfer learning called AdapterFusion which provides a simple and effective way to combine information from several tasks. By separating the extraction of knowledge from its composition, we are able to effectively avoid the common pitfalls of multi-task learning, such as catastrophic forgetting and interference between tasks. Further, AdapterFusion mitigates the problem of traditional multi-task learning in which complete re-training is required, when new tasks are added to the pool of datasets.

We have shown that AdapterFusion is compatible with adapters trained in both single-task as well as multi-task setups. AdapterFusion consistently outperforms fully fine-tuned models on the target task, demonstrating the value in having access to information from other tasks. While we observe gains using both ST-A as well as MT-A, we find that composing ST-A using AdapterFusion is the more efficient strategy, as adapters can be trained in parallel and re-used.

Finally, we analyze the weighting patterns of individual adapters in AdapterFusion which reveal that tasks with small datasets more often rely on information from tasks with large datasets, thereby achieving the largest performance gains in our experiments. We show that AdapterFusion is able to identify and select adapters that contain knowledge relevant to task of interest, while ignoring the remaining ones. This provides an implicit no-op option and makes AdapterFusion a suitable and versatile transfer learning approach for any NLU setting.

8.2 Outlook

Rücklé et al. (2020a) have studied pruning a large portion of adapters after Fusion training. Their results show that removing the less activated adapters results in almost no performance drop at inference time while considerably improving the inference speed. They also provide some initial evidence that it is possible to train Fusion with a subset of the available adapters in each minibatch, potentially enabling us to scale our approach to large adapter sets — which would otherwise be computationally infeasible. We believe that such extensions are a promising direction for future work.

Pfeiffer et al. (2020d) have achieved considerable improvements in the zero-shot cross-lingual transfer performance by dropping the adapters in the last layer. In preliminary results, we have observed similar trends with AdapterFusion when the adapters in the last layer are not used. We will investigate this further in future work.

Acknowledgments

Jonas is supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY center. Aishwarya was supported in part by a DeepMind PhD Fellowship during the time which this project was carried out. Andreas is supported by the German Research Foundation within the project “Open Argument Mining” (GU 798/25-1), associated with the Priority Program “Robust Argumentation Machines (RATIO)” (SPP-1999). This work was partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI) and Samsung Research (Improving Deep Learning using Latent Structure). Kyunghyun was a research scientist at Facebook AI Research part-time during which this project was carried out.

We thank Sebastian Ruder, Max Glockner, Jason Phang, Alex Wang, Katrina Evtimova and Sam Bowman for insightful feedback and suggestions on drafts of this paper.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. 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 1538–1548.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 160–167.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. *arXiv preprint*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

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.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: machine reading comprehension with contextual commonsense reasoning. 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 2391–2401.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5189–5197.

Anne Lauscher, Olga Majewska, Leonardo F. R. Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020. Common Sense or World Knowledge? Investigating Adapter-Based Knowledge Injection into Pretrained Transformers. *arXiv preprint*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics 2017*, 5:365–378.

Hector J. Levesque. 2011. The winograd schema challenge. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2873–2879.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4487–4496.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 216–223, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Jinseok Nam, Jungi Kim, Eneldo Loza Menc'ia, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classification - revisiting neural networks. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, pages 437–452.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*, pages 7–14.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Edwin Simpson, and Iryna Gurevych. 2020b. Low resource multi-task sequence tagging - revisiting dynamic conditional random fields. *arXiv preprint*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020c. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020d. UNKs Everywhere: Adapting Multilingual Language Models to New Scripts. *arXiv preprint*.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint*.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4465–4470.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Elad Ben-Zaken1 Shauli Ravfogel and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint*.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 506–516.

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)*, pages 3980–3990, Hong Kong, China. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020a. AdapterDrop: On the Efficiency of Adapters in Transformers. *arXiv preprint*.

Andreas Rücklé, Jonas Pfeiffer, and Iryna Gurevych. 2020b. MultiCQA: Zero-shot transfer of self-supervised text matching models on a massive scale. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2471–2486, Online. Association for Computational Linguistics.

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

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, National University of Ireland, Galway.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4822–4829.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6949–6956.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. 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 4462–4472.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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, Seattle, Washington, USA. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451.

Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai, and Iryna Gurevych. 2018. Cross-topic argument mining from heterogeneous sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3664–3674.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5986–5995.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. 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 4149–4158.

Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

M. Vidoni, Ivan Vulić, and Goran Glavaš. 2020. Orthogonal language and task adapters in zero-shot cross-lingual transfer. In *arXiv preprint*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 93–104.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800.

Yu Zhang and Qiang Yang. 2017. A survey on multitask learning. *arXiv preprint*.

A Appendices

A.1 Datasets

Commonsense Reasoning We work with a large number of datasets, all of which have emerged recently in this domain, ranging from sentence level and document level classification to multiple choice questions. The next sentence prediction task *HellaSWAG* (Zellers et al., 2019) is a more difficult version of the previously released *SWAG* dataset (Zellers et al., 2018). *Winogrande* (Sakaguchi et al., 2020) is a large scale and adversarially filtered (Zellers et al., 2018) adaptation of the *Winograd Schema Challenge* (Levesque, 2011). *Cosmos QA* (Huang et al., 2019) is a commonsense reading comprehension dataset which requires reasoning over larger text passages. *Social IQA* (Sap et al., 2019) is a multiple choice dataset which requires reasoning over social interactions between humans. *Commonsense QA* (Talmor et al., 2019) is a multiple choice dataset based on ConceptNet (Speer et al., 2017), which requires reasoning over general knowledge.

Sentiment Analysis We conduct experiments on two binary sentiment classification tasks on long and short text passages. *IMDb* (Maas et al., 2011) consists of long movie reviews and *SST-2* (Socher et al., 2013) consists of short movie reviews from Rotten Tomatoes[6].

Natural Language Inference (NLI) The goal is to classify whether two sentences entail, contradict, or are neutral to each other. For this we conduct experiments on *MultiNLI* (Williams et al., 2018), a multi-genre dataset, *SciTail* (Khot et al., 2018) a NLI dataset on scientific text, *SICK* (Marelli et al., 2014) a NLI dataset with relatedness scores, the composition of *Recognizing Textual Entailment (RTE)* datasets provided by Wang, Singh, Michael, Hill, Levy, and Bowman (2018), as well as the *Commitment Bank (CB)* (De Marneffe et al., 2019) three-class textual entailment dataset.

Sentence Relatedness We include two semantic relatedness datasets which capture whether or not two text samples include similar content. *Microsoft Research Paraphrase Corpus (MRPC)* (Dolan and Brockett, 2005) consists of sentence pairs which capture a paraphrase/semantic equivalence relationship. *Quora Question Pairs (QQP)* targets duplicate question detection.[7]

Misc The Argument Aspect corpus (Stab et al., 2018) is a three-way classification task to predict whether a document provides arguments *for*, *against* or *none* for a given topic (Nuclear Energy, Abortion, Gun-Control, etc). BoolQ (Clark et al., 2019) is a binary reading comprehension classification task for simple *yes, no* questions.

A.2 What Is The Best Adapter Setup?

As described in §2.2.3, the placement of adapter parameters Φ within a pretrained model is non-trivial, and thus requires extensive experiments. In order to identify the best ST-A setting, we run an exhaustive architecture search on the hyperparameters — including the position and number of adapters in each transformer layer, the position and number of pretrained or task dependent layer norms, the position of residual connections, the bottleneck reduction factors $\{2, 8, 16, 64\}$, and the non linearity $\{\text{ReLU, LeakyReLU, Swish}\}$ used within the adapter. We illustrate this in Figure 5. This grid search includes the settings introduced by Houlsby et al. (2019) and Bapna and Firat (2019). We perform this search on three diverse tasks[8] and find

[6] www.rottentomatoes.com

[7] data.quora.com/First-Quora-DatasetReleaseQuestion-Pairs

[8] SST-2, Commonsense QA, and Argument.

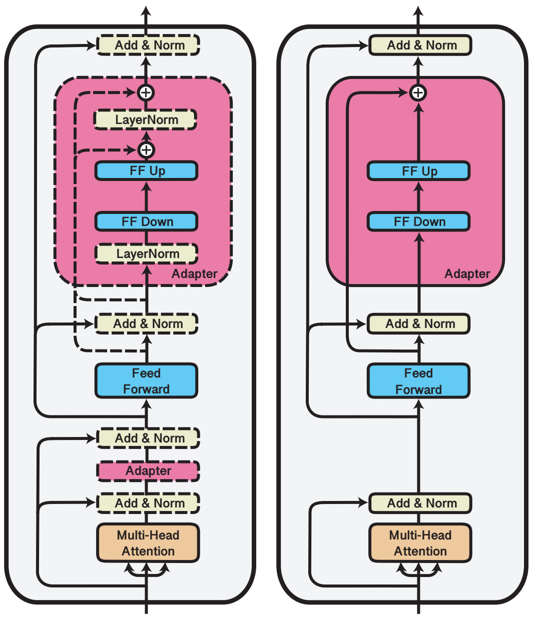


Figure 5: Different architectural components of the adapter. On the left, we show all components for which we conduct an exhaustive search (dashed lines). On the right, we show the adapter architecture that performs the best across all our tasks.

that across all three tasks, the same setup obtains best results. We present our results on the SST-2, Argument, and CSQA datasets in Figures 7, 8, and 9 respectively, at different granularity levels. We find that in contrast to Houlsby et al. (2019), but in line with Bapna and Firat (2019), a single adapter after the feed-forward layer outperforms other settings. While we find that this setting performs on-par with that of Houlsby et al. (2019), it requires only half the number of newly introduced adapters as compared to them, resulting in a more efficient setting in terms of number of operations.

For the single-task adapter setting, we thus perform all subsequent experiments with the best architecture illustrated in Figure 5 on the right and a learning rate of $1e-4$. In order to reproduce the multi-task results in Stickland and Murray (2019) and build upon them, for experiments involving multi-task training, we adopt their architecture as described in §2.2.3.

A.3 AdapterFusion Activations of all Layers

We present the cross-product of activations of AdapterFusion of all layers for BERT-Base and ST-A_{16} in Figure 6, as an extension to Figure 4.

A.4 BERT-base ST-A with Reduction Factors $\{2, 16, 64\}$

We present the ST-A results with different capacity leveraging BERT-base weights in Table 3. Reduction factors 2, 16, and 64 amount to dense adapter dimensions 384, 48, and 12 respectively.

A.5 ST-A and Fusion with ST-A Results with RoBERTa-base

In order to validate our findings of our best setup—ST-A—we re-evaluate our results leveraging RoBERTa-base weights. We present our results in Table 4. Similar to our findigs with BERT-base, especially datasets with less data profit from AdapterFusion. We find that, in contrast to BERT-base, RoBERTa-base does not perform well with high capacity adapters with reduction factor 2.

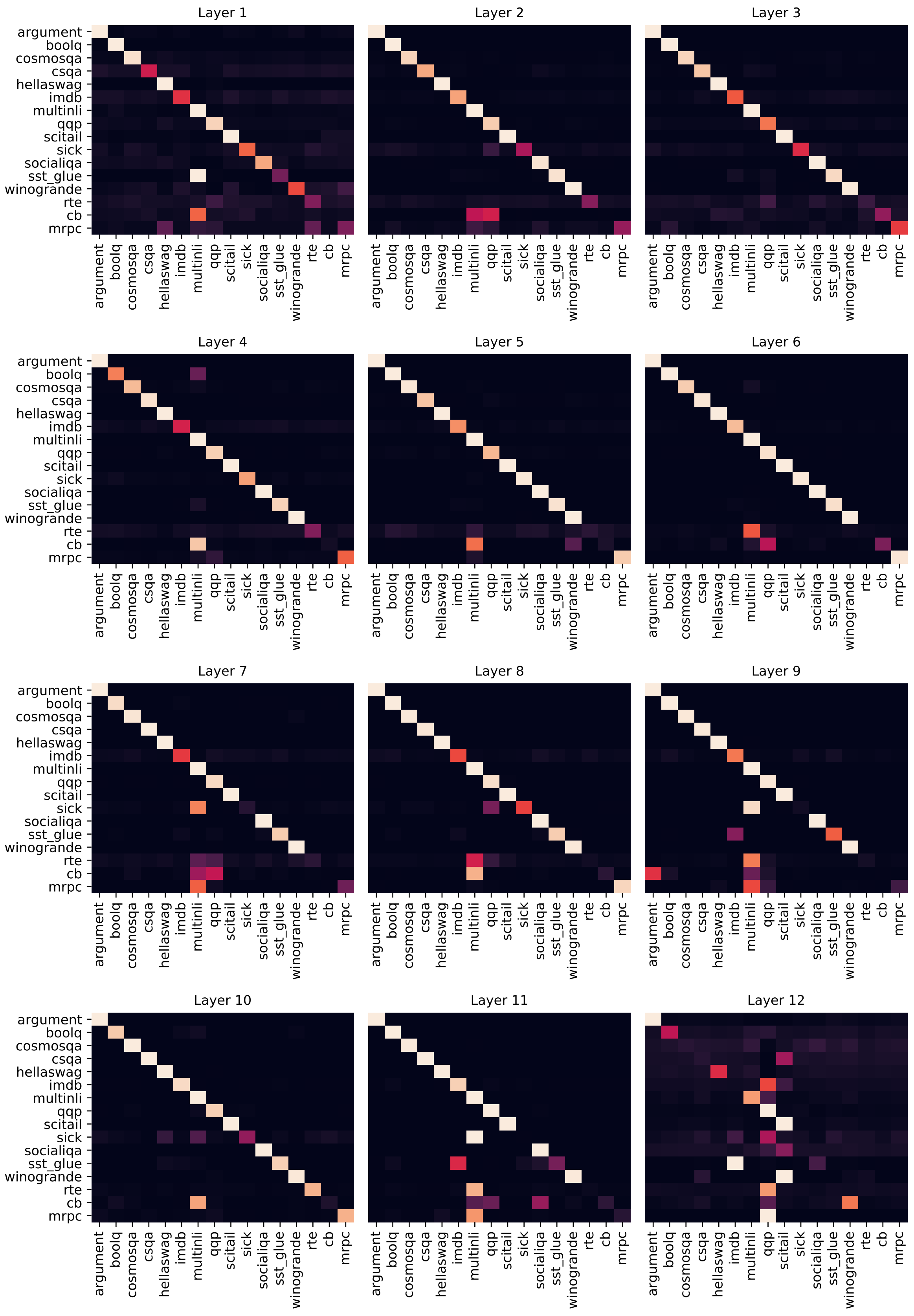
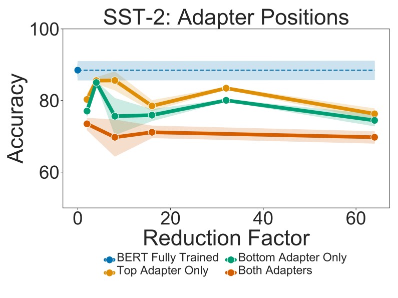
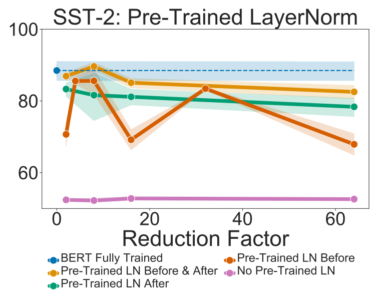


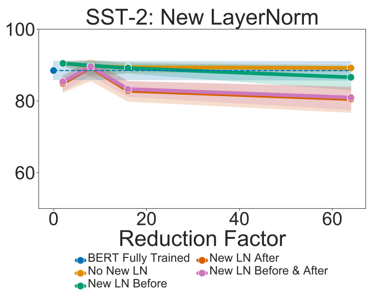
Figure 6: AdapterFusion activations in the 12 BERT-base layers. Target tasks are presented in rows, whereas the set of adapters are displayed in columns. Black squares indicate that an adapter has not been activated, whereas white cells indicate full activation.



(a) Adapter Positions in Layer

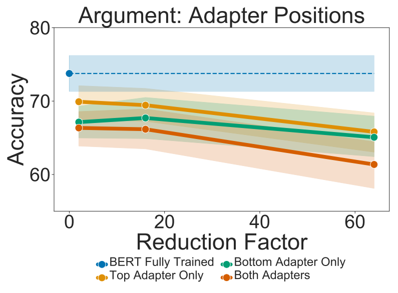


(b) Position of pretrained LayerNorm

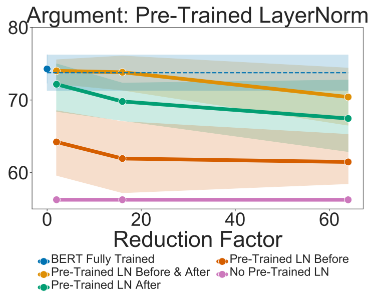


(c) Position of newly trained Layer-Norm

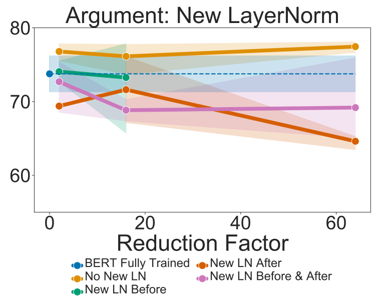
Figure 7: Results of the grid search on the SST-2 dataset over the architecture settings illustrated on the left of Figure 5. As we go from (a) to (c), the best performing setting is used for further search over other hyperparameters. We find that the best performing architecture is *Top Adapter Only* with *Pretrained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 5.



(a) Adapter Positions in Layer

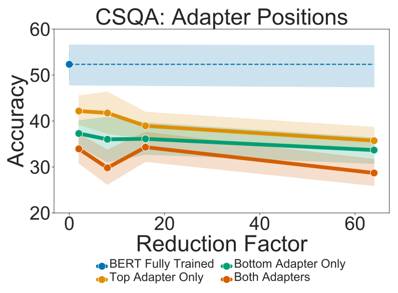


(b) Position of Pretrained LayerNorm

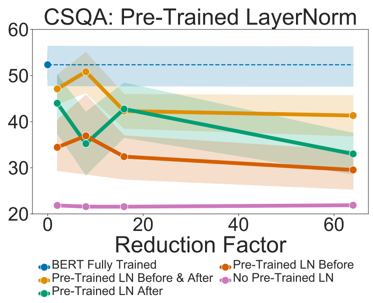


(c) Position of newly trained Layer-Norm

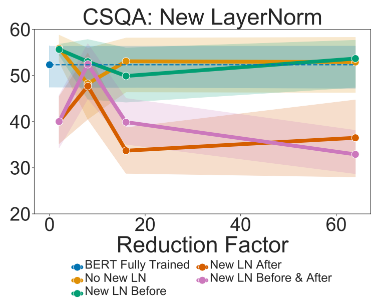
Figure 8: Results of the grid search on the Argument dataset over the architecture settings illustrated on the left of Figure 5. As we go from (a) to (c), the best performing setting is used for further search over other hyperparameters. We find that the best performing architecture is *Top Adapter Only* with *Pretrained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 5.



(a) Adapter Positions in Layer



(b) Position of Pretrained LayerNorm



(c) Position of newly trained Layer-Norm

Figure 9: Results of the grid search on the CSQA dataset over the architecture settings illustrated on the left of Figure 5. As we go from (a) to (c), the best performing setting is used for further search over other hyperparameters. We find that the best performing architecture is *Top Adapter Only* with *Pretrained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 5.

Dataset	ST-A$_2$	ST-A$_{16}$	ST-A$_{64}$
MultiNLI	84.60	84.32	84.08
QQP	90.57	90.59	89.73
SST	92.66 ±0.32	91.85 ±0.41	92.01 ±0.33
Winogrande	62.11 ±0.09	61.09 ±0.11	59.70 ±0.06
IMDB	94.20 ±0.28	93.85 ±0.07	93.90 ±0.14
HellaSwag	39.45 ±0.20	38.11 ±0.14	38.28 ±0.37
SocialIQA	60.95 ±0.15	62.41 ±0.11	62.23 ±0.73
CosmosQA	59.32 ±0.24	60.01 ±0.02	60.65 ±0.34
SciTail	94.44 ±0.81	93.90 ±0.16	93.82 ±0.49
Argument	76.83 ±0.21	77.65 ±0.34	77.64 ±0.56
CSQA	57.83 ±0.23	58.91 ±0.57	58.88 ±0.40
BoolQ	77.14 ±1.10	75.66 ±1.25	76.07 ±0.54
MRPC	86.13 ±1.59	85.16 ±0.52	85.58 ±0.32
SICK	87.50 ±0.14	86.20 ±0.00	85.70 ±0.42
RTE	70.68 ±4.57	71.04 ±1.62	69.16 ±1.59
CB	87.85 ±2.94	86.07 ±3.87	84.28 ±4.79
Mean	76.39	76.05	75.73

Table 3: Mean and standard deviation results (development sets) for each of the 16 datasets and reduction factors {2, 16, 64} for ST-A. Each model is initialized with BERT-base (Devlin et al., 2019) weights. The datasets are ordered by their respective training dataset size. Dashed horizontal lines separates datasizes $\{>40k, >10k, >5k\}$ respectively.

Dataset	Head	Full	ST-A$_2$	ST-A$_{16}$	ST-A$_{64}$	F. w/ ST-A$_{16}$	ST-A$_{16}^{\text{Houlsby}}$
MultiNLI	56.84	86.42	85.56	86.06	85.86	86.20	86.57
QQP	71.40	91.07	90.88 ±0.07	90.27	89.39 ±0.63	90.28	90.66
SST	81.86 ±0.21	94.29 ±0.22	93.71 ±0.29	93.80 ±0.23	93.35 ±0.43	93.67 ±0.13	94.17 ±0.15
Winogrande	51.93	66.77	51.27 ±0.78	65.58 ±0.53	62.43	66.01 ±0.47	63.46 ±6.38
IMDB	85.40	96.00	95.70	95.78 ±0.13	95.80	95.78 ±0.19	95.68 ±0.26
HellaSwag	41.16	63.53	61.09 ±0.08	61.57 ±0.14	61.18 ±0.21	61.52 ±0.07	61.21 ±0.37
SocialIQA	46.87	69.44	69.24	70.14 ±0.40	70.21	70.13 ±0.11	70.78 ±0.17
CosmosQA	41.88 ±0.29	68.52 ±0.49	68.01 ±0.94	68.76 ±0.53	68.62 ±0.55	68.64 ±0.04	69.18 ±0.34
SciTail	49.57	94.47	94.24	94.59 ±0.64	94.32	94.44 ±0.09	94.09 ±0.39
Argument	66.22 ±0.62	78.04 ±0.42	78.60 ±0.34	78.50 ±0.45	78.53 ±0.59	77.98 ±0.24	78.42 ±0.44
CSQA	41.37 ±0.34	65.81 ±0.59	66.11 ±0.60	66.30 ±0.38	64.03 ±0.27	66.52 ±0.18	67.53 ±0.70
BoolQ	62.17	81.89	80.86 ±0.86	80.83 ±0.27	80.17 ±0.25	80.86 ±0.15	81.11 ±0.54
MRPC	68.38 ±0.00	89.11 ±0.93	89.11 ±0.51	88.72 ±0.71	87.10 ±1.67	89.65 ±0.50	89.17 ±1.06
SICK	56.40	86.60	84.80	85.40 ±0.32	85.40	85.76 ±0.26	85.88 ±0.46
RTE	55.81 ±2.92	72.34 ±11.02	61.80 ±12.47	75.30 ±0.61	73.86 ±1.55	78.79 ±1.12	78.56 ±1.54
CB	59.64 ±11.05	90.00 ±1.60	87.14 ±6.85	89.28 ±2.82	81.07 ±4.82	92.86 ±3.79	89.64 ±3.87
Mean	58.05	81.08	78.63	80.83	79.52	**81.41**	81.18

Table 4: Mean and standard deviation results of models initialized with RoBERTa-base (Liu et al., 2019b) weights. Performances are measured on the development sets of the 16 datasets for the different architectural setups. The datasets are ordered by their respective training dataset size. Dashed horizontal lines separate datasizes $\{>40k, >10k, >5k\}$ respectively. **Head** indicates training only a classification head on top of fixed RoBERTa weights. For **Full** training we fine-tune all weights of RoBERTa. Single-Task adapters (**ST-A**) is the training of independently trained adapters for each task, using the architecture illustrated in Figure 5, indices {2, 16, 64} indicate the reduction factor. **Fusion w/ ST-A** show the results of AdapterFusion using the respective pretrained adapters. **ST-A$_{16}^{\text{Houlsby}}$** shows the results of ST-A with with architecture proposed by Houlsby et al. (2019).