AdapterFusion: Non-Destructive Task Composition for Transfer Learning

Jonas Pfeiffer[1], **Aishwarya Kamath**[2], **Andreas Rücklé**[1],
Kyunghyun Cho[2,3,4], **Iryna Gurevych**[1]
[1]Ubiquitous Knowledge Processing Lab, TU Darmstadt
[2]New York University [3]Facebook AI Research [4]CIFAR Associate Fellow
pfeiffer@ukp.tu-darmstadt.de

Abstract

Current approaches to solving classification tasks in NLP involve fine-tuning a pre-trained language model on a single target task. This paper focuses on sharing knowledge extracted not only from a pre-trained language model, but also from several source tasks in order to achieve better performance on the target task. Sequential fine-tuning and multi-task learning are two methods for sharing information, but suffer from problems such as catastrophic forgetting and difficulties in balancing multiple tasks. Additionally, multi-task learning requires simultaneous access to data used for each of the tasks, which does not allow for easy extensions to new tasks on the fly. We propose a new architecture as well as a two-stage learning algorithm that allows us to effectively share knowledge from multiple tasks while avoiding these crucial problems. In the first stage, we learn task specific parameters that encapsulate the knowledge from each task. We then combine these learned representations in a separate combination step, termed AdapterFusion. We show that by separating the two stages, i.e., knowledge extraction and knowledge combination, the classifier can effectively exploit the representations learned from multiple tasks in a non destructive manner. We empirically evaluate our transfer learning approach on 16 diverse NLP tasks, and show that it outperforms traditional strategies such as full fine-tuning of the model as well as multi-task learning.

1 Introduction

The current go-to method for solving classification tasks in NLP is to leverage deep neural networks pre-trained in a self-supervised fashion on a large amount of text. The dominant architecture is a transformer (Vaswani et al., 2017), which is 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 finetuning 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, finetuning all the weights in this way can lead to learning instabilities on smaller datasets. Additionally, each task of interest requires all the parameters of the network to be fine-tuned, which results in a specialized model for each task. Since these models are separate from each other, the only possible transfer of information is from the underlying pretrained model to the single task of interest — and not among *multiple tasks*.

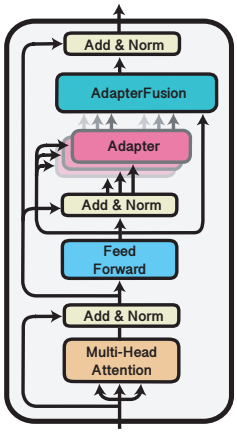


Figure 1: AdapterFusion architecture inside the original Transformer (Vaswani et al., 2017) model. The AdapterFusion component takes as input the representations of multiple adapters which have been trained on different tasks. It learns a parameterized mixer of the information encoded in the various adapters to improve on the target task.

There are two approaches for sharing information across multiple tasks. The first consists of starting from the pretrained language model and sequentially finetuning on each of the tasks one by one. 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 pre-trained 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., 2020a), 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 a solution that overcomes the problems of catastrophic forgetting and imbalanced training set sizes. Adapters do not require fine-tuning of all parameters of the pre-trained model, and instead introduce a small number of task specific parameters — while keeping the underlying pre-trained language model fixed. Thus, we can separately and simultaneously train adapters for multiple tasks, which all share the same underlying pre-trained parameters. However, to date, there exists no method for using *multiple* adapters to maximise the transfer of knowledge across target 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, thus not making use of the aforementioned benefits of adapters.

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, which we illustrate in Figure 1, has two components. The first component is an adapter that is trained to solve a specific target task without changing the weights of the underlying language model. The second component — our novel Fusion layer — combines the representations from several task adapters in order to better solve a specific target task.

Contributions Our main contributions are: (1) We compare multiple adapter architectures at different positions within transformer layers, finding a single more efficient setup which performs best on three diverse tasks. (2) We introduce a novel two-stage transfer learning strategy, termed *AdapterFusion*, which combines the information of multiple source tasks to perform better on the target task. (3) We empirically evaluate our proposed approach on a set of 16 diverse NLP tasks such as sentiment analysis, commonsense reasoning, paraphrase detection, and recognizing entailment. (4) We compare our approach with Stickland and Murray (2019) in which they train all the tasks together in a multi-task manner. (5) We show that our proposed approach increases the performances on tasks using adapters trained in a Single-Task as well as Multi-Task setup, outperforming results obtained by fully fine-tuning the transformer model on the target task.

This paper is structured as follows: In §2 we examine the traditional strategies for transfer learning as well as current adapter approaches. In §3 we introduce *AdapterFusion* as well as our novel two-stage learning algorithm. In §4 we present the details of our experimental setup, the data sets that we use, as well as our findings regarding the best adapter architecture. We present our results in §5, analyze them in §6, and conclude our work in §7.

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 pre-trained 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:

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

In the remainder of this paper, we refer to this pre-trained 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 in an efficient manner to improve on a target task m with $C_m = (D_m, L_m)$. In this work we focus on the setting where $m \in N$.

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

$$\Theta' \leftarrow \{\Theta_0, \Theta_1, \ldots, \Theta_N\}$$
$$\Theta_m \leftarrow \underset{\Theta}{\text{argmin}}\ L_m(D_m; \Theta')$$

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 it's training data.

2.1 Current Approaches to Transfer Learning

There are two predominant approaches to achieve sharing of information from one task to another (Pan and Yang, 2010; Torrey and Shavlik, 2010; Ruder, 2019):

2.1.1 Sequential Finetuning.

This involves sequentially updating all the weights of the model on each task. The following illustrates the N steps in the process of finetuning the model on each of the N tasks, where at each step the model is initialised with the parameters learned through the previous step:

$$\Theta_{0\rightarrow 1} \leftarrow \underset{\Theta}{\text{argmin}}\ L_1(D_1; \Theta_0)$$
$$\Theta' \sim \Theta_{0\rightarrow\ldots\rightarrow N} \leftarrow \underset{\Theta}{\text{argmin}}\ L_n(D_n; \Theta_{0\rightarrow 1\ldots\rightarrow N-1})$$
$$\Theta_{0\rightarrow\ldots\rightarrow N\rightarrow m} \leftarrow \underset{\Theta}{\text{argmin}}\ L_m(D_m; \Theta_{0\rightarrow\ldots\rightarrow N})$$

Here, it is assumed that $\Theta_{0\rightarrow\ldots\rightarrow N}$ is a good approximation of Θ'. However, this approach does not yield good results in practice beyond two sequential tasks (Phang et al., 2018) 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., 2020a).

$$\Theta_{0\rightarrow\{1,\ldots,N,m\}} \leftarrow \underset{\Theta}{\text{argmin}} \left(\sum_{n=1}^{N} L_n(D_n; \Theta_0) + L_m(D_m; \Theta_0) \right)$$

However MTL requires access to all the tasks at the same time 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 quite a challenge and requires heuristic approaches as applied in Stickland and Murray (2019).

2.2 Adapters

While the predominant methodology for transfer learning is to fine-tune all weights of the pre-trained model, *adapters* have recently been introduced as an alternative approach with applications in computer vision as well as the NLP domain (Rebuffi et al., 2017; Houlsby et al., 2019; Bapna and Firat, 2019; Stickland and Murray, 2019; Wang et al., 2020; Pfeiffer et al., 2020b). 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 pre-trained model (e.g., a transformer), the parameters Φ_n, where $n \in 1, \ldots, N$ 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) 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

For each of the N tasks, the model is initialized with parameters Θ_0. In addition, a set of new and randomly initialized parameters Φ_n are introduced (the adapter parameters). To share the same set of parameters Θ_0 across all otherwise independent tasks, the parameters in 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. The objective for each task $n \in 1, \ldots, N$ 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 pre-trained model in (Houlsby et al., 2019).

2.2.2 Multi-Task Adapters.

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:

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

where

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

2.2.3 Adapters in Practice

Introducing new adapter parameters in different layers of an otherwise fixed pre-trained model has been shown to perform on-par with, or only slightly below full model fine-tuning in many setups — e.g., in computer vision (Rebuffi et al., 2017) and in NLP (Houlsby et al., 2019; Bapna and Firat, 2019; Stickland and Murray, 2019). 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 pre-trained model is non-trivial, as has been shown by Houlsby et al. (2019). They experiment with different architectures, finding that a two-layer feed-foward neural network with a bottleneck works well. They place

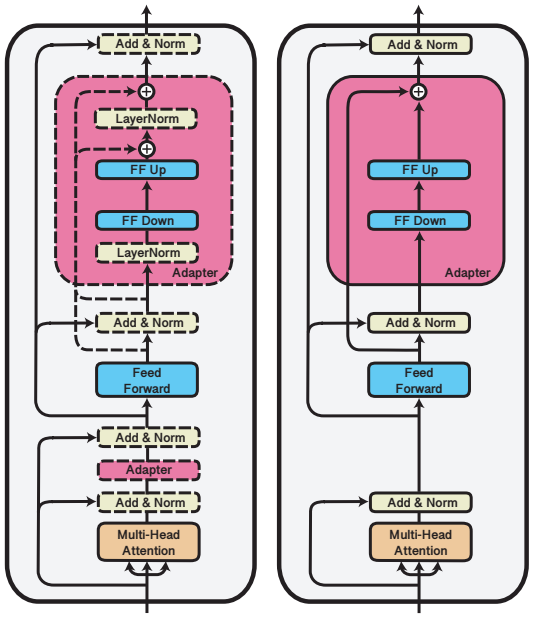


Figure 2: Different architectural components of the adapter. On the left, we show all components for which we conduct an exhaustive search (dashed lines). On the right, we show the Adapter architecture that performs the best across all our tasks.

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*). We illustrate these placements in Figure 2 (left). 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*.

Adapters trained in both single-task (ST-A) or multi-task (MT-A) setups consist of parameters that represent the information and specific traits of the respective tasks' training data. This results in a compression of information, which requires less space to store task-specific information. However, the distinct weights of adapters limit the sharing of information for a single downstream task, which might benefit from the learned representations of other tasks. In the next section we will describe our two stage algorithm which tackles the sharing of information stored in adapters that are trained on different tasks.

3 AdapterFusion

While Adapters avoid catastrophic forgetting by introducing task-specific parameters for each task independently, they do not provide a way to leverage information across tasks unless trained in a multi-

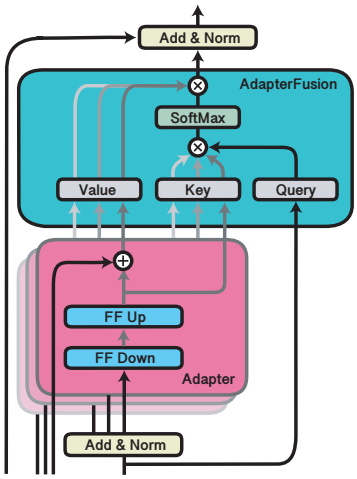


Figure 3: Our AdapterFusion architecture. This includes learnable weights *Query*, *Key*, and *Value*. *Query* takes as input the output of the pre-trained transformer weights. Both *Key* and *Value* take as input the output of the respective Adapters. The dot-product of the *query* with all the *keys* is passed into a SoftMax function, which learns to weight the Adapters with respect to the context of the data point.

task setting, which brings its own challenges. In the following, we present the learning algorithm used to train AdapterFusion for a given set of adapters (§3.1) and the architectural components of AdapterFusion (§3.2).

3.1 Learning algorithm

In the first stage of our learning algorithm, we train the adapters for each of the N tasks. This can be done either using the independently trained single-task adapters, or the jointly trained multi-task adapters — both of which have been discussed in §2.2.

In the second stage, we then combine the set of N adapters by using AdapterFusion. Here, the only tunable weights Ψ learn to leverage the N task adapters for the current 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 $\Theta_{0\rightarrow\{1,\ldots,N,m\}}$ in the MT-A setup.

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

3.2 Components

AdapterFusion learns to combine the information stored in each of the N task adapters Φ_n and the shared pre-trained model Θ, by introducing a new a set of weights Ψ. These parameters should take into account the information of each adapter and learn to combine these representation as a dynamic function of the data.

As illustrated in Figure 3, 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

$$\mathbf{s}_{l,t} = \text{softmax}(\mathbf{h}_{l,t}^\top \mathbf{Q}_l \otimes \mathbf{z}_{l,t,n}^\top \mathbf{K}_l), n \in \{1, \ldots, N\}$$

where $\otimes$ represents the dot product. $\mathbf{s}$ thus learns linear combination of each adapter, conditioned on the input and output of each adapter:

$$\mathbf{z}'_{l,t,n} = \mathbf{z}_{l,t,n}^\top \mathbf{V}_l, n \in \{1, \ldots, N\}$$

$$\mathbf{Z}'_{l,t} = [\mathbf{z}'_{l,t,0}, \cdots, \mathbf{z}'_{l,t,N}]$$

$$\mathbf{o}_{l,t} = \mathbf{s}_{l,t}^\top \mathbf{Z}'_{l,t}$$

Where $[\cdot, \cdot]$ indicates the concatenation of vectors.

Given the context, AdapterFusion thus learns to identify and activate the most useful adapter for a given data point, thus learning a parameterized mixer of information stored in the task specific adapters. With the residual connection around the AdapterFusion layer, this structure can also learn to act as a no-op during target task training.

4 Experiments

In this section, we describe the experiments that we use to validate the effectiveness of our approach in addressing the issues faced by other transfer learning methods. We provide a brief description of the 16 diverse data sets that we base our findings on, each of which uses accuracy as the scoring metric. We also give insights drawn from our search for the best adapter architecture.

We train both ST-A as well as MT-A in stage one of AdapterFusion, followed by their combination using our Fusion step. 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 check whether our two stage procedure alleviates the problems of interference between tasks. Finally, our experiments to compare MT-A with and without Fusion lets us evaluate the versatility of our approach wherein 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 (Devlin et al., 2019) as the pre-trained language model. We train ST-A, described in §4.2 and illustrated in Figure 2, for all datasets described in §4.1. We train them with reduction factors[1] $\{2, 16, 64\}$ and learning rate 0.0001. The results in Table 1 correspond to the reduction factor of 16. Comprehensive results with other reduction factors are presented in the Appendix. We follow the setup used in Stickland and Murray (2019) for training the MT-A. We use the default hyperparameters[2], and train a MT-A model on all datasets simultaneously.

For AdapterFusion, we empirically find that a learning rate of 0.00005 works well and is therefore used in all experiments. While we initialize **Q** and **K** randomly, we initialize **V** with a diagonal of ones and the rest of the matrix with random weights having a small norm $(1e-6)$. This has the effect of passing through the original output representation of each adapter through to the output of the *Value* matrix. We continue to regularize[3] the *Value* matrix to avoid introducing additional capacity to the AdapterFusion module.

4.1 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., 2019) 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 data set which requires reasoning over larger text passages. *Social IQA* (Sap et al., 2019) is a multiple choice data set which requires reasoning over social interactions between humans. *Commonsense QA* (Talmor et al., 2019) is a multiple choice data set 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[4].

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 data set, *SciTail* (Khot et al., 2018) a NLI data set on scientific text, *SICK* (Marelli et al., 2014) a NLI data set with relatedness scores, the composition of *Recognizing Textual Entailment (RTE)* data sets 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 data set.

Sentence Relatedness We include two semantic relatedness data sets 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.[5]

Misc We include additional datasets that do not fall in the aforementioned categories. 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.

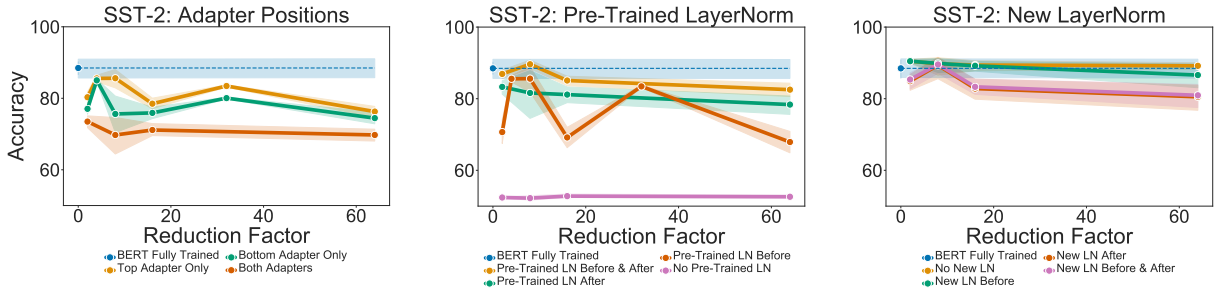
[1] A reduction factor indicates the factor by which the hidden size is reduced such that the bottle-neck size for BERT Base with factor 64 is reduced to 12 ($768/64 = 12$).

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

[3] We use $L1$ or $L2$ regularization.

[4] www.rottentomatoes.com

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



(a) Adapter Positions in Layer

(b) Position of Pre-Trained LayerNorm

(c) Position of newly trained LayerNorm

Figure 4: Results of the grid search on the SST-2 dataset over the architecture settings illustrated on the left of Figure 2. 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 *Pre-Trained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 2.

4.2 What Is The Best Adapter Setup?

As described in §2.2.3, the placement of adapter parameters Φ within a pre-trained model is non-trivial, and thus requires extensive experiments. In order to identify the best single-task adapter 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 pre-trained or task dependent layer norms, the position of residual connections, the bottleneck reduction factors $\{2, 8, 16, 64\}$, and the non linearity {ReLU, LeakyReLU, Swish} used within the adapter. We illustrate this in Figure 2. 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[6] and find that across all three tasks, the same setup obtains best results. We present our results on the SST-2[7] data set in Figure 4 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 2 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.

5 Results

In Table 1 we present the mean and standard deviation of our results on the development set[8] based on 5 random seeds[9].

Adapters Training only a prediction-head on the output of a pre-trained 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 highlights the necessity of more complex adaptation approaches. In Table 1 we present the results for reduction factor 16 which we find to have a good trade-off between number of newly introduced parameters and task performance. Interestingly, the ST-A have a regularization effect on some datasets, resulting in better performance on average for certain tasks, even though only a small percentage of weights are trained. On average we improve 0.66% by training ST-A instead of the *Full* model.

For multi-task adapters we find that despite the heuristic strategies introduced by Stickland and

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

[7] The results for Commonsense QA and ArgumentMining are in the Appendix, which is consistent with our findings on SST-2.

[8] For almost all data sets the test set is not provided.

[9] Except for MNLI and QQP as these data sets are very large

Dataset	Head		Full		ST-A		MT-A		Fusion w/ ST-A		Fusion w/ MT-A	
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
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
CosmosQA	50.06	±0.51	60.28	±0.40	60.01	±0.02	61.25	±0.90	60.65	±0.55	**62.78**	±0.07
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
HellaSwag	34.17	±0.27	**39.25**	±0.76	38.11	±0.14	36.47	±0.98	37.98	±0.01	37.36	±0.10
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
MultiNLI	54.59		83.17		**84.32**		82.49	±0.49	84.28		83.05	
QQP	76.79		**90.87**		90.59		89.47	±0.60	90.71		90.58	
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
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
SocialIQA	50.33	±2.50	62.05	±0.04	62.41	±0.11	61.21	±0.89	**63.16**	±0.24	62.56	±0.10
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
Winogrande	51.92	±0.35	60.01	±0.08	**61.09**	±0.11	57.70	±1.40	60.23	±0.31	59.32	±0.30
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
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
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
Mean	64.17		75.46		76.05		75.80		**77.33**		77.06	

Table 1: Mean and standard deviation results of the development sets of the 16 data sets for the different architectural setups. 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 2. 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.

Murray (2019) for sampling from the different datasets, there are considerable performance drops of more than 2 percentage points for *CSQA* and *MRPC*. This indicates that the common problem of multi-task learning is only partially addressed by MT-A and that learning a shared representation jointly still does not guarantee an optimal solution for all tasks. On average however, we do see a performance increase of 0.4% as compared to *Full* fine-tuning, proving that there are advantages in leveraging information from other tasks.

AdapterFusion The purpose of combining N different adapters is to transfer the task specific information from these adapters to a target task m (where $m \in \{1, \ldots, N\}$) that might benefit from this information. This is based on the hypothesis that if there exists at least one task that is helpful to the target task, using AdapterFusion should lead to performance gains on the target task. If no such task exists, then the performance should stay the same as when only the adapter for the task itself is utilized (assuming $m \in \{1, \ldots, N\}$). We compare this to the MT-A that are trained over the same set of tasks.

We can see in Table 1 that for both ST-A and MT-A there exist performance gains as well as performance drops. In the case where we employ AdapterFusion, we notice that having access to relevant tasks provides a boost in performance for the target task. For Fusion with ST-A we can see large performance gains of 6.5 percentage points for *RTE* and 5.64 percentage points for *MRPC*. We also see performance gains for commonsense tasks such as *CosmosQA* and *CSQA*. For Fusion with MT-A, the gains are smaller, 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 that both Fusion with ST-A and Fusion with MT-A models obtain performance gains compared to their original representation by 1.27 and 1.25 percentage points respectively. Fusion with ST-A performs the best across our tasks with an average accuracy of 77.33%.

In Table 2, we present the absolute performance gains of Adapters and AdapterFusion compared to the fully fine-tuned model. In Table 3, we compare AdapterFusion to ST-A and MT-A. The arrows indicate whether there has been an improvement ↗, decrease ↘, or if the the results remained the same →. For both, Fusion with ST-A and Fusion with MT-A, the left column compares to ST-A and the right column compares to MT-A. The most important results are summarized as follows:

In the case of Fusion with ST-A, for 15 out of 16 tasks, the performance remains the same or im-

Dataset	ST-A	MT-A	Fus. w/ ST-A	Fus. w/ MT-A
Argument	0.78	−1.18	0.78	−0.79
BoolQ	0.81	3.92	1.41	4.34
CosmosQA	−0.27	0.97	0.37	2.50
CSQA	0.04	−5.58	0.85	−2.15
HellaSwag	−1.14	−2.78	−1.27	−1.89
IMDB	−0.20	−1.49	0.23	−1.40
MNLI	1.15	−0.69	1.11	−0.12
QQP	−0.28	−1.40	−0.16	−0.29
SciTail	−0.42	0.21	0.28	0.47
SICK	−1.10	1.31	−0.02	3.13
SocialIQA	0.36	−0.84	1.12	0.51
SST	−0.53	−0.11	−0.19	0.61
Winogrande	1.08	−2.31	0.22	−0.69
RTE	5.63	12.20	11.41	14.55
CB	3.57	6.59	9.64	7.32
MRPC	0.01	−3.28	5.15	−0.46
Improved	9/16	6/16	12/16	8/16

Table 2: Performance gains of the two Adapter architectures and the Adapter Fusion models with respect to fully fine-tuned BERT are given in absolute numbers.

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

Table 3: Performance changes of AdapterFusion compared to ST-Adapters and MT-Adapters. Arrows indicate whether there has been an improvement ↗ (>0.3), decrease ↘ (<-0.3), or whether the results have stayed the same → $[-0.3, 0.3]$.

proves as compared to the task's pre-trained adapter. For 10 out of 16 tasks we see performance gains. This shows that having access to adapters from other tasks is valuable, and that we can leverage this to obtain better results on the target task.

We find that for 11 out of 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.

For only 7 out of 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 we have mentioned earlier, 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.

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, thus confirming that our method is very versatile.

Our findings together indicate that Fusion with ST-A is the most promising approach to sharing information across tasks. It allows for training adapters in parallel without requiring techniques to balance tasks with varying data set sizes and heuristic sampling strategies. It also allows for easily adding in more tasks as they become available without complete retraining as would be required in the multi-task setting. We also conduct experiments with multi-task adapters and Fusion with multi-task adapters, but as we can see from Table 2, these clearly underperform Fusion with ST-A.

In conclusion, while Fusion with MT-A does provide gains over simply using MT-A, the effort required to train these in a multi-task setting followed by the Fusion step are not warranted by the limited gains in performance as seen in Table 2. On the other hand, we find that Fusion with ST-A is a novel and highly efficient approach to transfer learning.

6 Analysis

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

We plot the results for layers 1, 7, 9, and 12 and

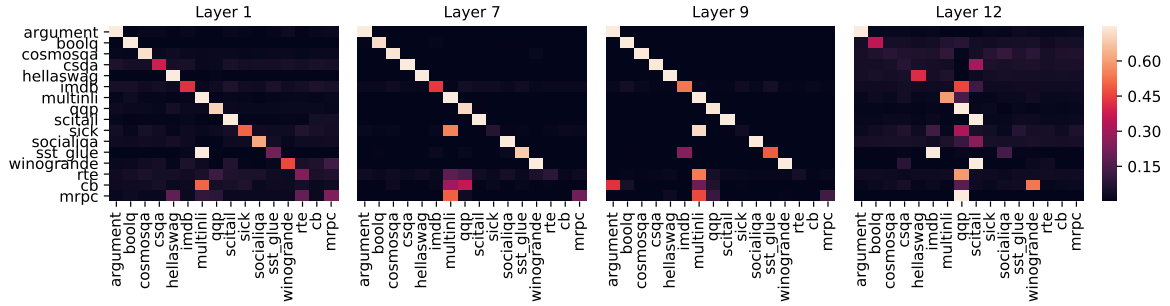


Figure 5: AdapterFusion activations of pre-trained **ST-Adapters**. Rows indicate the target task m, columns indicate Adapters n.

ST-A in Figure 5 (a plot that includes all layers is included in the Appendix). Rows indicate the target task m and columns indicate all available auxiliary tasks n. 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 is able to identify helpful and thus complementary adapters, while at the same time only extracting information from adapters if they are beneficial for the target task m. We further find that in line with previous work (Phang et al., 2018; Conneau and Kiela, 2018; Reimers and Gurevych, 2019), *MultiNLI* is a useful intermediate task that benefits a large number of target tasks, e.g. *BoolQ, SICK, CSQA, SST-2, CB, MRPC, RTE*. Further, *QQP* is a helpful general adapter, which is utilized by a large number of tasks, e.g. *SICK, IMDB, RTE, CB, MRPC, SST-2*. In particular, tasks with small data sets such as *CB, RTE, MRPC* often strongly rely on the *MultiNLI* and *QQP* adapters, which have been trained on significantly larger training sets. The performance on these tasks improves by up to 6.5 percentage points, indicating that the information from other task adapters can be highly beneficial for target tasks with smaller training sets.

7 Conclusion

We propose a novel method for transfer learning called AdapterFusion, in which we first propose a new adapter architecture for use with transformer based models, as well as a method to effectively combine the adapters to solve a wide range of target tasks. This combination is learnt for every time step and layer of the underlying transformer architecture and enables our model to identify and leverage adapters of other tasks as and when required. Through our experiments we show that AdapterFusion is compatible with adapters trained in both single-task as well as multi-task setups. AdapterFusion consistently outperforms the results that are obtained with the underlying transformer model that is fully fine-tuned on only the target task, proving the value in sharing information from other tasks. While we find gains in both single-task as well as multi-task settings, we observed that independently training adapters and then combining them using AdapterFusion provides the best results. By this approach, we are able to circumvent the issues of sequential fine-tuning and multi-task learning such as catastrophic forgetting and interference between tasks, respectively. We also conduct an analysis of adapters that are activated during AdapterFusion, supporting previous results that highlight *MultiNLI* and *QQP* as being beneficial intermediate tasks.

We believe that AdapterFusion establishes important ground-work for a wide variety of extensions including (1) developing effective combinations of heterogeneous tasks without suffering from common problems such as catastrophic forgetting, and (2) enabling in-depth analyses on what kind of information is stored in different layers of pre-trained language models.

Acknowledgments

Jonas Pfeiffer is supported by the Hessian research excellence program “Landes-Offensive zur Entwicklung Wissenschaftlich-Ökonomischer Exzellenz” (LOEWE) as part of the research center EmergenCITY. Aishwarya Kamath is supported in part by a DeepMind PhD Fellowship. Andreas Rücklé 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).

References

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.

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.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Trans. Assoc. Comput. Linguistics*, 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*.

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 arXiv:1907.11692*.

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

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer. *arXiv preprint*.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *CoRR*, abs/1811.01088.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

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.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098.

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. 2019. WINOGRANDE: an adversarial winograd schema challenge at scale. *CoRR*, abs/1907.10641.

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.

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.

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. *CoRR*, abs/2002.01808.

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 multi-task learning. *CoRR*, abs/1707.08114.

A Appendices

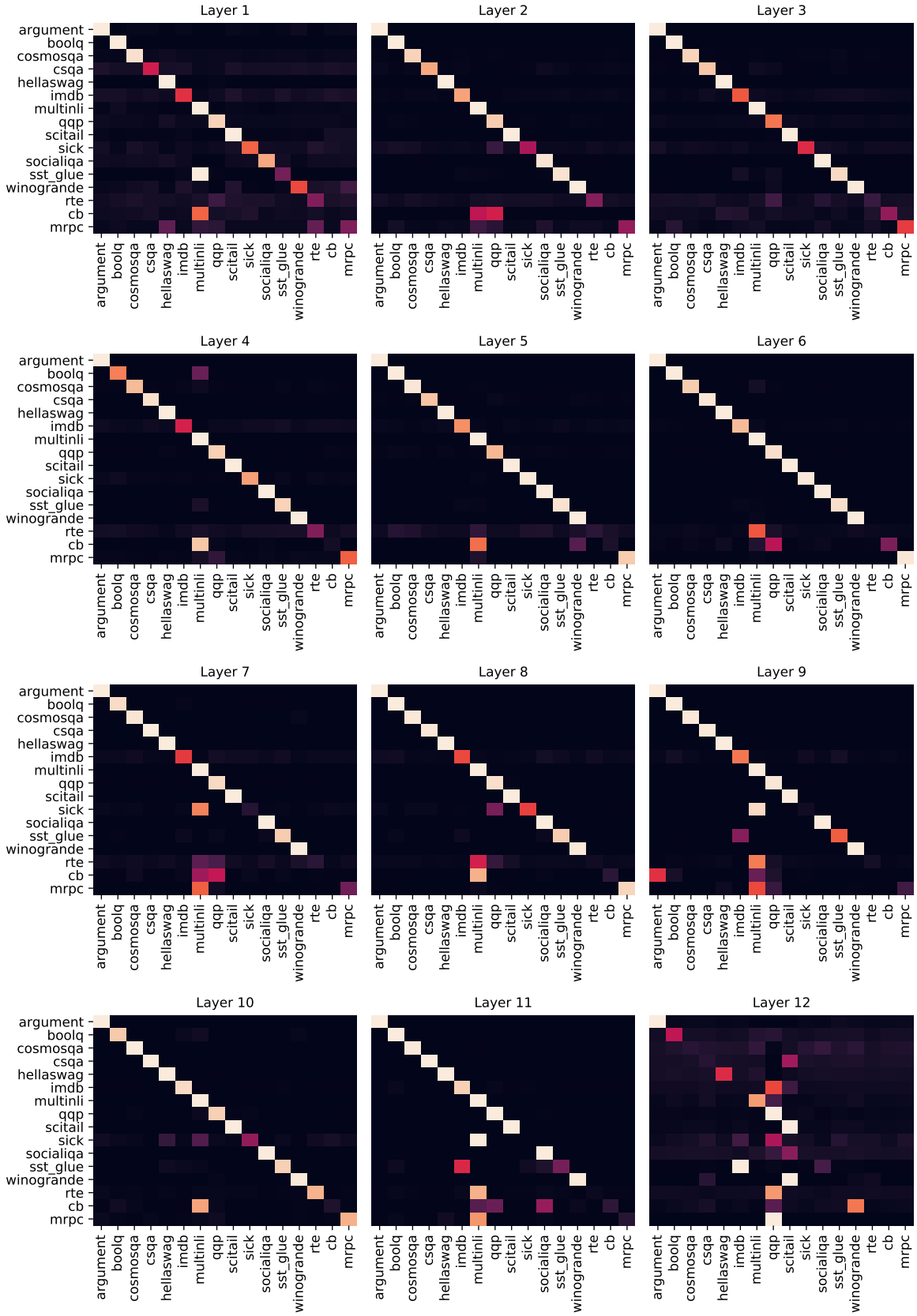
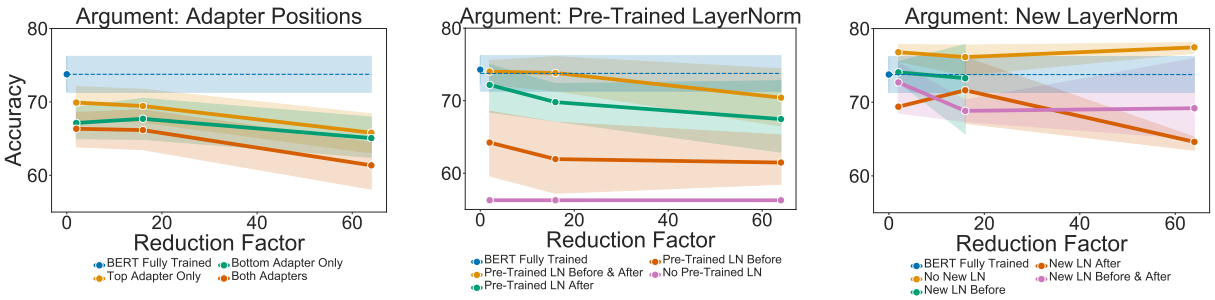


Figure 6: AdapterFusion

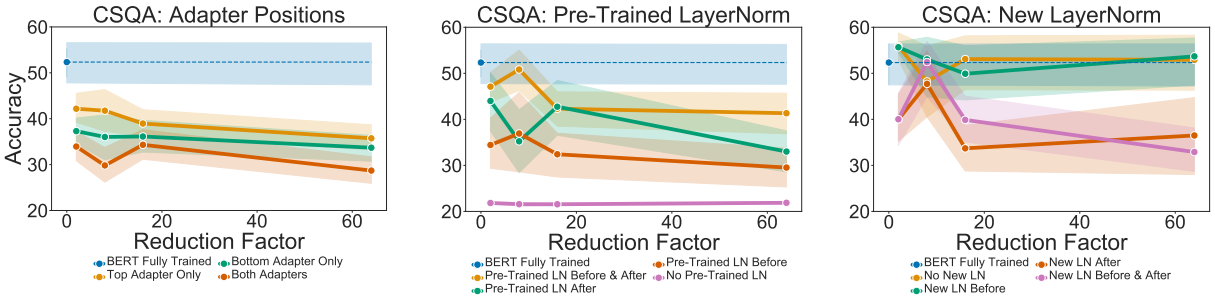


(a) Adapter Positions in Layer

(b) Position of Pre-Trained LayerNorm

(c) Position of newly trained LayerNorm

Figure 7: Results of the grid search on the Argument dataset over the architecture settings illustrated on the left of Figure 2. 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 *Pre-Trained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 2.



(a) Adapter Positions in Layer

(b) Position of Pre-Trained LayerNorm

(c) Position of newly trained LayerNorm

Figure 8: Results of the grid search on the CSQA dataset over the architecture settings illustrated on the left of Figure 2. 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 *Pre-Trained LayerNorm Before & After* including *No New LayerNorm*. This Architecture is illustrated on the right of Figure 2.

Dataset	**ST-A** Red. Fac. 2	**ST-A** Red. Fac. 16	**ST-A** Red. Fac. 64
Argument	76.83 ±0.21	77.65 ±0.34	77.64 ±0.56
BoolQ	77.14 ±1.10	75.66 ±1.25	76.07 ±0.54
CosmosQA	59.32 ±0.24	60.01 ±0.02	60.65 ±0.34
CSQA	57.83 ±0.23	58.91 ±0.57	58.88 ±0.40
HellaSwag	39.45 ±0.20	38.11 ±0.14	38.28 ±0.37
IMDB	94.20 ±0.28	93.85 ±0.07	93.90 ±0.14
MultiNLI	84.60	84.32	84.08
QQP	90.57	90.59	89.73
SciTail	94.44 ±0.81	93.90 ±0.16	93.82 ±0.49
SICK	87.50 ±0.14	86.20 ±0.00	85.70 ±0.42
SocialIQA	60.95 ±0.15	62.41 ±0.11	62.23 ±0.73
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
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
MRPC	86.13 ±1.59	85.16 ±0.52	85.58 ±0.32
Mean	76.39	76.05	75.73

Table 4: Mean and standard deviation results of the development sets of the 16 data sets for reduction factors {2, 16, 64} for ST-A. Each model is initialized with BERT-Base (Devlin et al., 2019) weights.