Investigating Parameter-Efficient Transfer Learning Model Adaptation through Tuned Parameter Transfer

Alexander To, Wanyu Li and Rathin Bector
University of California Berkeley
Berkeley, CA, United States
{alexander.to,li.wanyu,rathin.bector}@berkeley.edu

Abstract

Parameter efficient transfer learning (PETL) enables the adaptation of large pre-trained language models (PLM's) to a specific task while only tuning a fraction of the total parameters within the base PLM. This reduces the time and the computational cost required during model adaptation as, unlike conventional fine tuning, PETL methods add a relatively small number of tunable parameters while 'freezing' most, if not all, of the original PLM parameters during adaptation. In this project, we adapted the task agnostic source prompt transfer (SPoT) methodology described in Vu et al. (2022) to various PETL architectures, and evaluated the transferability of source tuned PETL parameters across a variety of natural language tasks. We tested the transferability of parameters trained on SQuAD and MNLI as source tasks to each task in the SuperGLUE Benchmark, across four PETL-modified BART-large transformer models derived from the unified framework proposed by He et al. (2021). We found that, like prompt tuning, both prefix tuning (Li and Liang, 2021) and MaM adapters (He et al., 2021) can benefit from parameter transfer, and report an increase in SuperGLUE for almost all source tuned PETL models. At the time of writing, this work is the first to demonstrate the positive transferability of the PETL parameters for prefix tuning and MaM adapters.

1 Introduction and Background

Parameter Efficient Transfer Learning (PETL) has recently emerged as a new model tuning paradigm in Natural Language Processing (NLP) whereby a relatively small set of parameters are added to a frozen pre-trained language model (PLM) and are tuned on a downstream task to attain strong performance (Brown et al., 2020; Houlsby et al., 2019). Within this framework, a distinction is made between the concepts of 'full fine tuning' (FFT) and 'fine tuning' (FT). The former refers to the 'traditional' fine tuning of the full set of model parameters within the PLM, whereas the latter refers to the tuning of additional external parameters while partially or fully freezing the original PLM parameters.

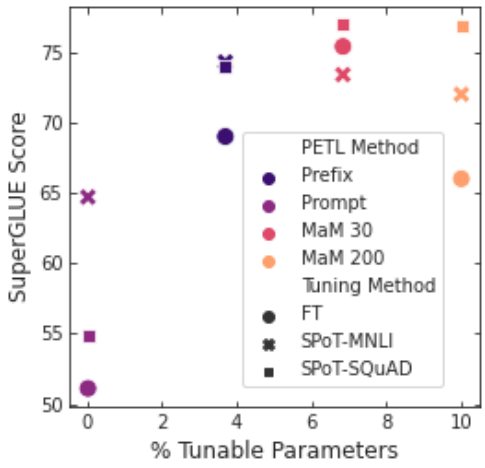

Figure 1: Plot of the percentage of tunable parameters during fine tuning vs SuperGLUE score for each PETL method and fine tuning method. FT indicates traditional fine tuning.

An advantage of the parameter efficient approach is that only a single copy of the original PLM needs to be trained and stored along with a smaller set of added parameters that are tuned for a wide variety of target tasks, rather than needing to train and store a fine-tuned PLM for each task or dataset. Additionally, the time and computational capacity needed for model fitting can be reduced.

The parameter efficient approach was inspired by work on zero/few-shot learning (Brown et al., 2020) and the 'pre-train, prompt, and predict' paradigm (Liu et al., 2021). The success of prompt-based learning, where all tasks are cast as text-generation exercises and the PLM is trained to predict an out-

Method	Modified Component	Δh Function	Composition Function	Key Parameter	% Tunable Parameters
Prompt	Input	concat(*h, P*)	$h \leftarrow [h;P]$	Prompt *l*=110	0.03
Prefix	Attention	softmax($xW_qP_k^T$)P_v	$h \leftarrow (1-\lambda)h + \lambda\Delta h$	Prefix *l*=200	3.7
MaM	Attention	softmax($xW_qP_k^T$)P_v	$h \leftarrow (1-\lambda)h + \lambda\Delta h$	Prefix *l*=[30, 200]	[6.8, 10.0]
	FFN	ReLU(hW_{down})W_{up}	$h \leftarrow h + s.\lambda\Delta h$	Scaling = 4	

Table 1: Summary of PETL transformations and key parameters used in this work. The unified PETL framework applies a modification Δh to the input h. For prompt tuning, a prompt embedding P of dimension $e \times l$, where e is the embedding dimension, is concatenated on the input h. The percentage of model parameters updated during fine tuning is calculated using the methodology from He et al. (2021), relative to the total parameters in BART-large before PETL parameters are added.

put as a function of some text based input (for instance, 'tl;dr' for summarisation), has led to the development and adoption of numerous approaches for PETL.

Adapter tuning was an early approach to PETL that added neural modules between each layer of the PLM. The method is claimed to require two orders of magnitude fewer parameters as compared to full fine tuning (Houlsby et al., 2019). In prompt tuning, a series of 'soft prompts' are prepended onto the inputs, and their embeddings are updated during training (Lester et al., 2021). In prefix tuning, tunable prefixes are prepended only to the keys and values of the multi-head attention layers, and only these parameters are updated during training (Li and Liang, 2021). In Low-Rank Adaptation (LoRA), trainable rank decomposition matrices are inserted into each layer to approximate parameter updates (Hu et al., 2021).

Noting the myriad of PETL approaches, He et al. (2021) recently recast the aforementioned techniques within a single unified PETL framework, where the modifications to the PLM are aligned within a shared set of parameter and design modifications to a base transformer. This is possible as prefix tuning can be cast as a type of adapter, whereby prefix tuning modifies only the keys and values on the input attention head h, whereas the original adapters modify the output of the multi-head attention and the feed forward network (FFN).

This unified approach allows a practitioner to incorporate components and parameters from different PETL frameworks into a single model. For instance, He et al. combined design elements from prefix-tuning and parallel adapters to develop a Mix-and-Match' adapter (MaM), which utilises a prefix tuning architecture with prefix length $l = 30$ along with a scaled parallel adapter on the FFN. This particular architecture was shown to outperform other PETL architectures on MNLI and SST2 benchmarks.

While PETL frameworks focus on architectural and training methodologies to better adapt large PLM's to downstream tasks, recent work has also focused on the transferability of the learned parameters. Working on the transferability of soft prompts, Vu et al. (2022) developed a method known as Source Prompt Transfer (SPoT), which demonstrated that prompts learned during prompt tuning can be used as an initialization when fine tuning on another downstream target task. Applying the concept of SPoT, they showed that training prompts on MNLI and other tasks resulted in improvements in test scores on SuperGLUE tasks, relative to fine tuning prompts on that target task alone. Their work showed that SPoT helps to narrow the performance gap between FFT and prompt tuning for medium sized models.

In this work, we applied the concept of parameter transfer beyond prompt tuning, to several different PETL architectures built on top of a BART-Large PLM (Lewis et al., 2020). We evaluated whether a SPoT-like approach can be successfully applied to other PETL methods, specifically prefix tuning and MaM. Specifically, we explored whether other PETL model architectures can improve performance on a given target task, by first being fine-tuned on a distinct source task. To the best of our knowledge, to date there has not been an evaluation of SPoT's effectiveness beyond prompt tuning on a T5 base PLM.

2 Methods

2.1 Datasets and Preparation

The task agnostic SPoT approach involves first 'source tuning' PETL parameters (e.g. soft prompt embeddings when prompt tuning) on a given dataset, and then initialising with the tuned PETL parameters during 'task tuning' on the target task.

We cast all tasks within the text-to-text framework using the SeqIO python package.[1] We closely

[1]https://github.com/google/seqio

followed the SeqIO implementation of Raffel et al. and Lester et al., however we omitted the task prefix prepended on the input, and used a BART-specific vocabulary for tokenization. [2]

In this work, we used the Multi-Genre Natural Language Inference corpus (MNLI) (Nangia et al., 2017) and the Stanford Question Answering Dataset (SQuAD), (Rajpurkar et al., 2016) as source tasks, since they demonstrated strong downstream performance on SuperGLUE tasks in the original work by Vu et al..

Following from the SPoT authors, we evaluated each PETL method and source-task combination on each SuperGLUE task (Wang et al., 2019). The SuperGLUE Benchmark is a combination of ten different NLP tasks across a range of NLP topics, and is therefore a useful benchmark to evaluate the broad transferability of source tuned parameters. [3]

Each SuperGLUE task was also cast within the text-to-text framework. When transforming ReCoRD and WsC to the text-to-text format, only one answer from the original answer span was used as target.[4] This transformation potentially constrains the model's learning as only part of the correct answers are 'shown' to the model during the training. Furthermore, due to issues with EM calculation with text-to-text datasets, we took the checkpoint with the highest accuracy score for ReCoRD and MultiRC. [5]

2.2 Model Architectures

We utilised the code repository from He et al.[6], which is forked from Huggingface transformers repository, to build the PETL models used in this work (Wolf et al., 2019). We conducted experiments with three distinct PETL architectures built on top of a base BART-Large PLM. They were prefix tuning, prompt tuning and MaM adaptor . For the MaM adapter, we evaluated two variants with prefix length of 30 and 200. A summary of the PETL variants used in this work is shown in Table 1.

Method	MNLI (Acc.)	SQuAD (Acc./F1)
Prefix, *l*=110	89.4	62.9 / 44.8
Prompt, *l*=200	78.7	51.7 / 33.9
MaM, *l*=30	89.5	64.8 / 46.7
MaM, *l*=200	89.8	64.3 / 46.2

Table 2: Summary of top validation scores obtained for each PETL method evaluated on the source tasks validation sets for MNLI validation and SQuAD v1.

3 Experiments

3.1 Baselines

We established two types of baselines in our work. The first was a fine-tuning PETL baseline, where PETL parameters overlaid on a frozen BART-large PLM were trained directly on the target SuperGLUE task. We trained the model for 100K steps on the target task and evaluated performance with the appropriate task-specific metric. This FT baseline served as the primary comparison for the effectiveness of tuned parameter transfer.

We also performed full fine-tuning baseline on each SuperGLUE task, for which the base PLM (without any PETL parameters) was trained on directly on the target SuperGLUE task. Theoretically, this FFT baseline represents the 'upper limit' of model performance on each downstream task. Due to GPU and time constraints, we were only able to perform 100K gradient update steps for each task, which was likely not adequate for model convergence. Nevertheless, we included it as a helpful benchmark to compare results, while noting that it may not represent the true performance limit of the model on each task. The hyperparameters used during FFT are listed in the Appendix Table 8

3.2 Source Tuning

During source tuning, only the PETL parameters were trained. Following from Vu et al., we trained each PETL implementation for 260K steps on MNLI and SQuAD, check-pointing progress on the validation set every 1.5K steps. The checkpoint with the highest accuracy score was used as the source of tuned parameters for downstream target tuning. Each PETL method and source task combination was evaluated only once due to time and

[2] All the text-to-text datasets used in this work are publicly available at: https://huggingface.co/stjokerli. The datasets were originally pulled from Tensorflow datasources (Ten) and accessed via Huggingface datasets (Wolf et al., 2019)

[3] The SuperGLUE benchmark includes tasks CB (Reitinger and Mazurek, 2021), COPA(Roemmele et al., 2011), BoolQ (Clark et al., 2019), WiC (Pilehvar and Camacho-Collados, 2018), WsC (Davis, 2016), RTE (Dagan et al., 2006), MultiRC (Khashabi et al., 2018), ReCoRD (Zhang et al., 2018), AX-g (Rudinger et al., 2018) and AX-b(Wang et al., 2019)

[4] This method was also used by Raffel et al.

[5] In text-to-text form, the reference paragraph is embedded in each example and the dataset shuffled, thus during evaluation not all questions associated with a given paragraph may be shown to the model and hence the EM score may not be properly calculated.

[6] He et al.'s code is publicly available on githib: https://github.com/jxhe/unify-parameter-efficient-tuning

resource constraints. Curves for the train loss, validation loss and validation accuracy during source tuning are shown in the Appendix.

Hyperparameters were chosen based off the settings used by Vu et al. and He et al., unless poor convergence was observed. For all source tuning experiments, a default dropout of 0.1, an AdamW [Reference?] optimiser and polynomial learning rate decay were used. Training was performed using a single NVIDIA RTX 3090 with 24 GB of GPU memory, using a batch size of 16 with gradient step of two, thus enabling an effective batch size of 32. A table outlining relevant parameters used during source tuning for each run is listed in the Appendix Table 7

Source tuning for the prompt tuning method was harder than the prefix tuning and MaM adaptor equivalents. For prompt, the default parameters from He et al. (2021) consistently yielded training cycles with no improvement in validation loss and accuracy metric. Consequently, we experimented with variations in learning rate, optimiser, learning rate decay schedule, label smoothing, effective batch size and warm up steps. Ultimately, we found that a slightly higher initial learning rate of 7×10^{-5} and a small number of warm up steps led to better convergence for prompt tuning.

Evaluation scores from the source tuning of all PETL architectures on MNLI and SQuAD are shown in Table 2. MaM and prefix tuning showed strong performance on MNLI, each with an evaluation accuracy score over 89%. Meanwhile, prompt tuning achieved a lower accuracy of 78.7% on MNLI. Similar trends were observed for source tuning on SQuAD, where a higher F1-Score and Accuracy were obtained by prefix and MaM than by prompt.

In order to investigate this further, a sweep of prompt length between 10 and 400 tokens was performed to investigate whether longer prompts could improve model performance. The plots of validation loss, accuracy and training loss are shown in Appendix, Figure 2. These experiments found that increasing prompt length is a limited method to increase evaluation accuracy. Beyond a value of 110 prompt tokens, there was no significant increase (rather a decrease) in validation score. This was also observed by Lester et al., who found that increasing prompt length can actually lead to worse downstream performance. Lester et al. and Vu et al. both showed that the largest gains in model performance are obtained by increasing the size of the base PLM, and that only modest increases are realised by increases in prompt length.[7] Consequently, we proceeded with a prompt length of 110 for the investigation into SPoT.

3.3 Target Tuning

During target tuning, we instantiated our model by loading checkpoints of the source tuned PETL models and training on the target task for 100K steps. We modified the trainer function of the model to re-start the scheduler and optimiser, giving us full flexibility over these parameters during target tuning. Target tuning was performed using either an A100 GPU with 40 GB GPU memory or using the aforementioned RTX3090 GPUs.

Like with source tuning, we used the training parameters used in He et al. (2021) and Vu et al. (2022) as starting points, deviating from these parameters if poor convergence or over-fitting was observed. In order to make a fair comparison, we aimed to use the same effective batch size for a given task, across all PETL methods. For each task, we chose as large a batch size as possible to maximise GPU memory utilisation. Where possible, we also tried to apply the same scheduler and learning rate settings for a given task. However, occasionally slight variations in these parameters were required in order to ensure good convergence. A full list of the key hyperparameters used during target tuning is provided in the Appendix Table 9. An AdamW optimiser, polynomial learning rate decay rate, model dropout of 0.1 and label smoothing of 0.1 were kept constant for all target tuning runs. The key variables varied across target tasks were the effective batch size (via changes to the gradient update steps and batch size) and the number of warm up steps.

Table 3 lists the validation scores on all SuperGLUE tasks with validation datasets. [8] In general, prompt tuning achieved lower benchmark and SPoT tuned scores compared to the MaM and prefix tuned equivalents. This might be expected given the lower number of tunable parameters in prompt as well as the lower scores observed for prompt during source tuning.

As Vu et al. report detailed SuperGLUE validation results, discussion of the validation scores

[7] We note that SPoT helps 'close the gap' and direct the interested reader to Figure 3a in (Lester et al., 2021)

[8] Note Ax-g and Ax-b provide test datasets only and are therefore omitted

Method	Source Task	CB F1/Acc	RTE Acc	COPA Acc	BoolQ Acc	WiC Acc	WsC Acc	ReCoRD F1/EM	MultiRC F1a/EM
Prefix	None	96.1/94.8	75	56	80.4	66.81	36.44	82.8/82.0	81.3/69.6
	MNLI	**100/100**	89.10	66.0	83.5	69.6	33.5	80.8/80.0	80.2/68.4
	SQuAD	90.7/93.0	82.3	75	82.2	64.69	70.5	80.1/79.1	79.9/42.7
Prompt	None	48.6/69.6	56.3	53	62.1	54	32.69	55.2/53.3	34.9/38.5
	MNLI	70.2/85.8	80.6	67	64.9	59.5	32.75	59.8/57.9	70.9/51.5
	SQuAD	58.6/83.9	68.2	64	62.1	52.19	19	52.2/51.0	21.3/33.3
MaM, *l*=30	None	94.6/92.9	82.9	**79**	**85.1**	**71.5**	33.62	**85.1/84.4**	80.0/69.2
	MNLI	96.3/98.2	89.2	76	83.5	70.3	17.13	80.2/79.4	79.6/69.2
	SQuAD	93.1/94.7	81.8	**79**	83	**71.5**	23	83.0/82.4	81.7/71.8
MaM, *l*=200	None	87.6/81.8	75.5	53	73.4	66.4	32.8	82.8/83.6	62.9/50.2
	MNLI	**100/100**	**90.8**	60	82.4	68.8	29.9	80.3/81.0	80.4/70.4
	SQuAD	96.5/95.1	79.3	**79**	81.5	68.1	**74.7**	81.5/80.8	**81.1/79.1**
FFT	None	96.4/97.3	81.6	85	84.9	74.0	34.6	87.1/86.6	72.0/57.5

Table 3: Detailed task specific breakdown of the SuperGLUE validation scores for each PETL method and source task tested in this work. The highest scores (excluding FT scores) for each task are highlighted in bold. Rows with 'None' are equivalent to fine tuning. A single run was performed for each method/source task/SuperGLUE task combination.

obtained enables a direct comparison between their work and that herein. However, as our work focused on the analysing the effectiveness of SPoT by comparing *test* scores, we provide some comparisons between their work and our validation results in the Appendix. Nevertheless we emphasize that overall, the general trend of task agnostic SPoT improving overall SuperGLUE validation scores is consistent in both works (and not only for prompt tuning), although not every source and target task combination validation score benefit/detriment from SPoT observed in their work aligned with that observed in this work.

Noting that we observed mixed SPoT effectiveness on MaM($l = 30$) we increased the number of prefixes on the attention mechanism to 200, to assess whether the observed decrease in SPoT performance was due to the smaller number of prefixes in the MaM architecture.

Table 3 shows that increasing the MaM prefix length to 200, resulted in an improvement in average validation scores from source tuning with both MNLI and SQuAD. These improvements came primarily through positive transfer to BoolQ, WiC, and MultiRC, and a smaller decrease in the performance for WsC and ReCoRD. Further analysis of the effectiveness of SPoT and the effect of model architecture is given in proceeding section.

4 Results and Discussion

4.1 SPoT Effectiveness

Certified test scores on each SuperGLUE task were obtained for each PETL method, benchmark and source tuned checkpoint. 1 plots the SuperGLUE score versus the percentage of tunable parameters, relative to the original number of model parameters before PETL parameters were added. The figure shows almost all PETL methods improved overall SuperGLUE test score when SPoT was used versus the fine-tuned benchmark, with the single exception being MNLI source tuned MaM($l = 30$). This suggests that task agnostic SPoT can be effectively applied beyond prompt tuning, to other PETL methods such as prefix tuning and MaM adapters.

Method	Source Task	SuperGLUE Score	ΔFT %abs.	ΔFFT %abs.
Prefix	-	69.0	-	- 9.0
	MNLI	74.3	+ 7.1	- 2.0
	SQuAD	74.0	+ 6.8	- 2.4
Prompt	-	51.1	-	- 32.6
	MNLI	64.7	+ 21.0	- 14.6
	SQuAD	54.8	+ 6.8	- 22.7
MaM, *l*=30	-	75.4	-	- 0.5
	MNLI	73.4	- 2.7	- 3.2
	SQuAD	77.0	+ 2.1	+ 1.6
MaM, *l*=200	-	66.0	-	- 12.9
	MNLI	72.0	+ 8.3	- 5.0
	SQuAD	76.9	+ 14.2	+ 1.5
FFT	-	75.8	-	-

Table 4: Summary of Super Glue Scores obtained for each PETL method tested, including the absolute percentage difference between task-agnostic SPoT, versus FT (No SPoT) and FFT.

The overall scores and scores broken out by individual task are shown in Table 4 and Table 5 respectively. MNLI was found to be an effective source task for transferring prompt parameters, resulting in a 21% increase in SuperGLUE score, while SQuAD resulted in a more modest increase of 6.8%. These prompt transfer results are somewhat consistent with the results reported by Vu et al.. Tuning prefixes with MNLI and SQuAD source-tuned parameters resulted in an improve-

Method	Source Task	CB F1/Acc	RTE Acc	COPA Acc	BoolQ Acc	WiC Acc	WsC Acc	ReCoRD F1/EM	MultiRC F1a/EM	Ax-b M.Corr	Ax-g G.P./Acc
Prefix	None	81.6/86.0	70.3	46.6	78	65.6	69.2	79.9/78.7	79.3/39.2	17.7	100/50.6
	MNLI	85.1/92.4	83.4	64.6	80.9	65.7	73.3	79.5/78.3	79.1/38.2	48.9	93.8/63.2
	SQuAD	81.8/90.8	77.2	73	80	63.9	70.5	80.1/79.1	79.9/42.7	32.3	97.2/53.1
Prompt	None	49.0/71.2	51.8	50.6	62.3	55.4	61	53.2/51.5	28.8/01.6	5.5	100/50
	MNLI	71.0/88.4	76.2	64.6	65.7	59	67.1	59.0/57.5	69.7/18.7	42.6	**100/51.1**
	SQuAD	55.4/80.4	68.5	62.6	62.4	49	67.1	50.8/49.8	20.7/01.2	25.7	100/50.6
MaM, *l*=30	None	80.3/86.0	77.9	76.2	81.9	68	73.3	**82.5/81.5**	80.0/42.0	30.1	99.4/55.9
	MNLI	63.3/91.2	83.1	75	81	**68.1**	66.4	78.6/77.4	78.6/38.5	**51.5**	96.1/65.4
	SQuAD	86.9/92.4	78.1	**80.2**	**82**	67.6	74	82.2/81.3	80.4/44.8	38.1	97.2/53.7
MaM, *l*=200	None	75.8/81.6	73.8	50.4	74.2	65	66.4	82.1/81.2	60.6/14.5	27.6	100/50.6
	MNLI	84.4/93.6	**83.8**	54.8	79.7	64.9	65.1	78.7/77.7	79.8/41.7	53	92.1/65
	SQuAD	**88.4/93.6**	79.3	79	81.5	**68.1**	**74.7**	80.3/79.4	**80.7/43.4**	34.9	95.5/53.4
FFT	None	84.0/88.4	76.2	80.2	82.9	70.1	69.9	83.4/82.5	78.7/37.1	32.2	97.8/57.9

Table 5: Detailed task specific breakdown of the SuperGLUE test scores for each PETL method and source task tested in this work. The highest scores for each task are highlighted in bold, excluding the FT scores. Rows with 'None' equivalent to fine tuning, since only training on the task test and validation set was performed.

ment in SuperGLUE score by 7.5% and 6.8% respectively. Interestingly, MaM($l = 30$) showed an improvement with SQuAD, but a deterioration with MNLI, in contrast to the trend in validation scores, discussed above, where neither MNLI or SQuAD parameter transfer resulted in improvements.

The decrease in overall SuperGLUE score for MNLI source tuned MaM($l = 30$) is due to the worse performance on CB, WsC and ReCoRD. Noting that CB is a similar task to MNLI and that this decrease in performance was not observed for other PETL methods, we believe that repetition of our experiments with different random seeds is required as a starting point for further investigation.

Notwithstanding the decrease we observed for MaM($l = 30$), source transfer from MNLI and SQuAD led to increases in SuperGLUE scores relative to the baseline for CB, RTE, COPA and BOOlQ for all PETL methods. SQuAD source parameters appeared to transfer poorly to WiC for all methods, and both MNLI or SQuAD source parameters appeared to transfer poorly to ReCoRD across all PETL methods, other than prompt for which MNLI parameter transfer resulted in a 10.1% increase in accuracy and F1 score relative to the baseline.

While the differences in improvements on individual SuperGLUE tasks were too small to draw inference from, we observed that SQuAD parameter transfer performed slightly worse on WiC, and MNLI parameter transfer performed slightly worse on ReCoRD and MultiRC.

4.2 SPoT vs FT and FFT

Table 4 shows that for prefix and prompt methods, task agnostic SPoT has 'closed the gap' between FT and FFT, since the SuperGLUE performance of MNLI and SQuAD source-tuned models is closer to that of the FFT benchmark.

In the instance of MaM adapters, the SQuAD source-tuned model actually outperformed both the FT and FFT benchmarks, which more likely indicates that our FFT BART benchmark was 'under tuned' with only 100,000 gradient update steps. It should be noted that Lewis et al. reported a higher RTE score in their original work of 87.0 compared to the 76.2 score we were able to achieve in our experiment. [9]. Meanwhile, the MNLI source-tuned MaM($l = 30$) showed slightly worse performance compared to the relevant FT and FFT benchmarks. Nevertheless, our results indicate that task agnostic SPoT can help MaM adapters achieve SuperGLUE. scores close to the FFT baseline.

4.3 Effect of Prefix Length on MaM Adapters

The test scores for source tuned MaM($l = 200$) were slightly lower than the equivalent source tuned MaM($l = 30$). On the other hand, the benchmark MaM($l = 200$) performed much worse than the benchmark MaM($l = 200$). This is counter-intuitive, since the extra parameters should enable better fit on the target task.

The lower scores for the MaM($l = 200$) are possibly due to insufficient/inadequate training. The model, given the extra parameters, may require greater than 100,000 steps to fully train. In addition, Li and Liang and He et al. observed in their

[9]Having cast our tasks as text-to-text may also account for some of this score difference, since the original BART authors did not cast their RTE dataset within the text-to-text framework. A full investigation of this effect however is out of the scope of this work.

work that prefix models did not perform better with increased prefix length (l=30 was the optimal value in He et al's work). While He et al. did not perform a similar analysis to find the optimal prefix length for the MaM adapter, a similar effect may be evident in our analysis. While the overall SuperGLUE are worse for the MaM($l = 200$) variant, the scores on a few individual SuperGLUE tasks are better for that variant.

Despite these open questions, it is worth noting that the highest SuperGLUE score in this work was achieved for a MaM($l = 30$) trained with SQuAD parameter transfer.

4.4 Task Similarity

We observed a similar pattern to Vu et al. (2022) with regards source-target pairs that displayed favorable parameter transfer. For instance, Vu et al. found that prompts trained on MNLI were similar to those of CB and RTE, and transferred well to those tasks as well. Our results showed similar trends, not only for the prompt, but also for Prefix and MaM, where MNLI transferred well to both CB and RTE for all architectures. Additionally, like Vu et al., we observed that SQuAD transferred well to WsC, COPA and BOOlQ. However, we did not find consistent evidence of SQuAD transferring well to ReCoRD as was found by Vu et al..

5 Conclusion

Our work provides evidence that the principles and application of task agnostic SPoT can apply not only to prompt tuning, but also to other parameter efficient transfer learning methods, namely prefix tuning and MaM adapters.

We were able to replicate the results of Vu et al., and demonstrate that prompts transferred from MNLI and SQuAD can outperform a standard initialization baseline, when evaluated on the SuperGLUE benchmark, using a BART base PLM. Further, we were able to demonstrate that prefixes source tuned on MNLI and SQuAD can also lead to improved SuperGLUE scores of 6.8 and 7.1% respectively.

In the case of MaM adapters, we observe that, consistent with prefix and prompt tuning, the addition of extra prefixes does not necessarily result in better model performance, or improve task transferability of the learnt PETL parameters. For MaM adapters, SQuAD appears to transfer well to SuperGLUE tasks, while MNLI transfer showed mixed results depending on the prefix length.

Future work involves further repetition of the existing experimental runs, and expanding the study to a wider array of source tasks and base model sizes.

We also reported the first results of a text-to-text BART model evaluated on SuperGLUE, and showed that task-agnostic spot can improve on fine tuning, increasing test scores close to the level achieved with full fine tuning for prompt tuning, prefix tuning and MaM adapters.

Acknowledgments

The authors would like to thank their course instructors Daniel Cer, Mark Butler and Joachim Rahmfeld for their guidance. They are also grateful to Tu Vu and Jinxuan He for their assistance explaining their respective work in this area. They would also like to acknowledge the support from Moreton Bay Technology for access to compute resources.

References

TensorFlow Datasets, A collection of ready-to-use datasets.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 2020-December.

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. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:2924–2936.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3944 LNAI:177–190.

Ernest Davis. 2016. Winograd Schemas and Machine Translation.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a Unified View of Parameter-Efficient Transfer Learning.

Neil Houlsby, Andrei Giurgiu, Stanisraw Jastrzçbski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:4944–4953.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking Beyond the Surface: A Challenge Set for Reading Comprehension over Multiple Sentences. Technical report.

Brian Lester, Rami Al-Rfou, Noah Constant, and Google Research. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. pages 3045–3059.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. pages 7871–7880.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 4582–4597.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. Technical report.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R. Bowman. 2017. The RepEval 2017 Shared Task: Multi-Genre Natural Language Inference with Sentence Representations. *RepEval 2017 - 2nd Workshop on Evaluating Vector-Space Representations for NLP, Proceedings of the Workshop*, pages 1–10.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 2383–2392.

Nathan Reitinger and Michelle L Mazurek. 2021. Replication Data for: ML-CB.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. Technical report.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender Bias in Coreference Resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, Louisiana. Association for Computational Linguistics.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, Daniel Cer, and Google Research. 2022. SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer. *To appear in ACL 2022*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *Advances in Neural Information Processing Systems*, 32.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. ReCoRD: Bridging the Gap between Human and Machine Commonsense Reading Comprehension.

Appendix

A Validation Score Analysis

Our results show that SPoT was effective for prompt tuning and source tasks, showing an improvement in validation scores relative to the corresponding baseline. However, there were differences observed in the effectiveness of SPoT dependent on the source task, and these differences did not always align with that reported by Vu et al.. The MNLI source tuned prompt model showed a relative improvement compared to the benchmark for all SuperGLUE tasks, while the SQuAD equivalent showed results varied by SuperGLUE task. CB, RTE and COPA scores improved relative to the baseline, however no improvement was observed for WiC, WsC, ReCoRD and MultiRC. It should be noted, that this is inconsistent with the observations of Vu et al, whom observed positive improvements of source tuned SQuAD on WsC and WiC. [10]

Given resource and time constraints, we were not able to perform replication of this work to further investigate this discrepancy, however the aforementioned nuances in MultiRC and ReCoRD metrics or the differences in base model used may be responsible for some of this variance. On the other hand, SPoT as applied to MaM($l = 30$) did not show an improvement in average validation score relative to the baseline[11]. SPoT for MaM($l = 30$) was particularly ineffective on WsC (which showed high variance), as well as on WiC, BoolQ, ReCoRD and MultiRC.

B Supplementary Figures and Tables

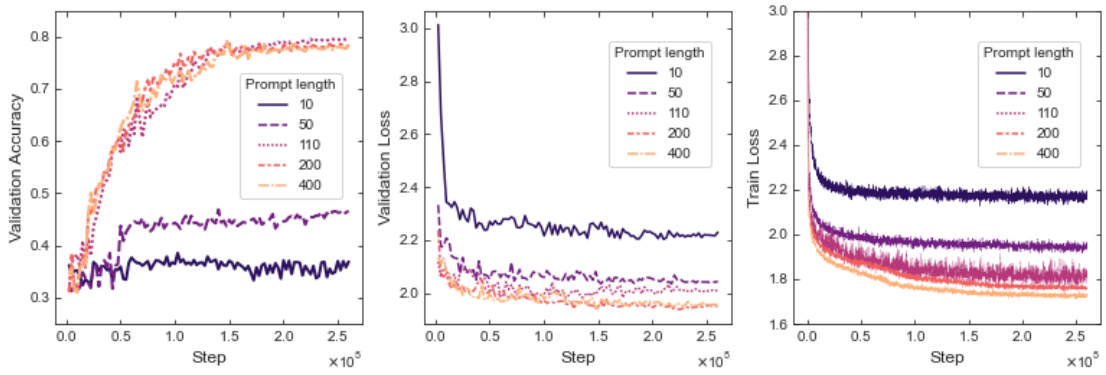

Figure 2: Plot of Validation Accuracy (left), validation loss (centre) and Training loss (right) versus prompt length for a BART-Large model adapted for Prompt Tuning on the MNLI Task.

Dataset	Usage	Train	Development	Test
MNLI	Source Tuning	392,702	19,647	N/A
SQuAD		87,599	10,570	N/A
SuperGLUE	Target tuning			
- CB		250	56	250
- AX-b		N/A	N/A	1,104
- AX-g		N/A	N/A	356
- COPA		400	100	500
- RTE		2,490	3,000	277
- ReCoRD		138,854	15,176	10,000
- BoolQ		9,427	3,270	3,245
- WsC		259	146	104
- WiC		5,428	638	1,400
- MultiRC		27,243	4,848	9,693

Table 6: Summary of all datasets used in this work.

[10] Scores for MultiRC and ReCoRD were not reported, we refer the reader to table 6 in (Vu et al., 2022)

[11] Note here this average is calculated as the average across all validation tasks, and is not a proxy for the overall SuperGLUE score

PETL Variant	Optimizer	LR Decay Schedule	Max. Learning Rate	Warm-up Steps	Batch Size	Gradient Steps
MaM, $l = 30, 200$	AdamW	Cosine	5x10^{-5}	0	16	2
Prefix, $l = 200$			7x10^{-5}	0		
Prompt, $l = 10, 50, 110, 200, 400$			5x10^{-5}	200		

Table 7: Summary of parameters used during source tuning for each PETL method tested.

Target Task	Optimizer	LR Decay Schedule	Max. Learning Rate	Warm-up Steps	Effective Batch Size
CB	AdamW	Polynomial	7×10^{-6}	10,000	32
RTE			7×10^{-6}		
COPA			5×10^{-7}		
BOOlQ			1×10^{-6}		
WiC			5×10^{-7}		
WsC			5×10^{-7}		
ReCoRD			1×10^{-5}		
MultiRC			1×10^{-6}		

Table 8: Summary of parameters used during full fine tuning of a BART-large model on each SuperGLUE task.

PETL Method	Source Task	Target Task	Max. Learning Rate	Warm-up Steps	Effective Batch Size
Prefix	Baseline	CB, RTE, BoolQ, MultiRC	$1x10^{-5}$		32
		COPA	$1x10^{-5}$	5000	92
		WiC	$1x10^{-5}$	5000	64
		WsC	$2x10^{-6}$	5000	120
		ReCoRD	$1x10^{-5}$		80
	MNLI	CB, RTE, BoolQ, WiC, MultiRC	$1x10^{-5}$		32
		COPA	$1x10^{-5}$	5000	92
		WsC	$2x10^{-6}$	5000	120
		ReCoRD	$1x10^{-5}$		80
	SQuAD	CB, WiC	$5x10^{-7}$	5000	32
		RTE, COPA, BoolQ	$2x10^{-6}$		32
		WsC	$2x10^{-6}$	5000	120
		ReCoRD	$1x10^{-5}$		80
		MultiRC	$1x10^{-5}$		32
Prompt	Baseline	CB	$1.4x10^{-5}$	200	32
		RTE, BoolQ, MultiRC	$1.4x10^{-5}$	200	32
		COPA, WiC	$1.4x10^{-5}$	200	92
		WsC	$2x10^{-6}$	5000	120
		ReCoRD	$1.4x10^{-5}$	200	80
	MNLI	CB, WiC	$1.4x10^{-5}$	200	32
		RTE, BoolQ, MultiRC	$1.4x10^{-5}$	200	32
		COPA	$1.4x10^{-5}$	5000	92
		WsC	$2x10^{-6}$	5000	120
		ReCoRD	$1.4x10^{-5}$	200	80
	SQuAD	CB, RTE, COPA, BoolQ, WiC	$5x10^{-5}$	5000	32
		WsC	$1x10^{-5}$	5000	60
		ReCoRD	$1x10^{-5}$		80
		MultiRC	$2x10^{-6}$		32
Mam - 30	Baseline	CB, RTE, BoolQ, MultiRC	$1x10^{-5}$		32
		COPA	$1x10^{-5}$	5000	92
		WiC	$1x10^{-5}$	5000	64
		WsC	$2x10^{-6}$	5000	120
		ReCoRD	$1x10^{-5}$		80
	MNLI	CB, RTE, BoolQ, MultiRC	$1x10^{-5}$		32
		COPA	$1x10^{-5}$	5000	92
		WiC	$1x10^{-5}$	5000	64
		WsC	$2x10^{-6}$	5000	120
		ReCoRD	$1x10^{-5}$		80
	SQuAD	CB, RTE, COPA, BoolQ	$1x10^{-6}$		32
		WiC	$5x10^{-7}$	5000	32
		WsC	$2x10^{-6}$	5000	120
		ReCoRD	$1x10^{-5}$		80
		MultiRC	$1x10^{-5}$		32
Mam - 200	Baseline	CB, RTE, COPA, BoolQ, WiC, WsC, ReCoRD, MultiRC	$2x10^{-6}$		32
	MNLI	CB, RTE, COPA, BoolQ, WiC, WsC, ReCoRD, MultiRC	$2x10^{-6}$		32
	SQuAD	CB, RTE, COPA, BoolQ, WiC, WsC, ReCoRD, MultiRC	$1x10^{-6}$		32

Table 9: Summary of parameters used during target tuning for each PETL method tested.