A Comprehensive Analysis of Adapter Efficiency

Nandini Mundra [1 2] **Sumanth Doddapaneni** [1 2] **Raj Dabre** [3] **Anoop Kunchukuttan** [1 2 4] **Ratish Puduppully** [5]
Mitesh M. Khapra [1 2]

Abstract

Adapters have been positioned as a parameter-efficient fine-tuning (PEFT) approach. However, adapters have not been sufficiently analyzed to understand if PEFT translates to benefits in training/deployment efficiency and maintainability/extensibility. Through extensive experiments on many adapters, tasks, and languages in supervised and cross-lingual zero-shot settings, we clearly show that for Natural Language Understanding (NLU) tasks, the parameter efficiency in adapters does not translate to efficiency gains compared to full fine-tuning of models. More precisely, adapters are relatively expensive to train and have slightly higher deployment latency. Furthermore, the maintainability/extensibility benefits of adapters can be achieved with simpler approaches like multi-task training via full fine-tuning, which also provide relatively faster training times. We, therefore, recommend that for moderately sized models for NLU tasks, practitioners should rely on full fine-tuning or multi-task training rather than using adapters. Our code is available at https://github.com/AI4Bharat/adapter-efficiency.

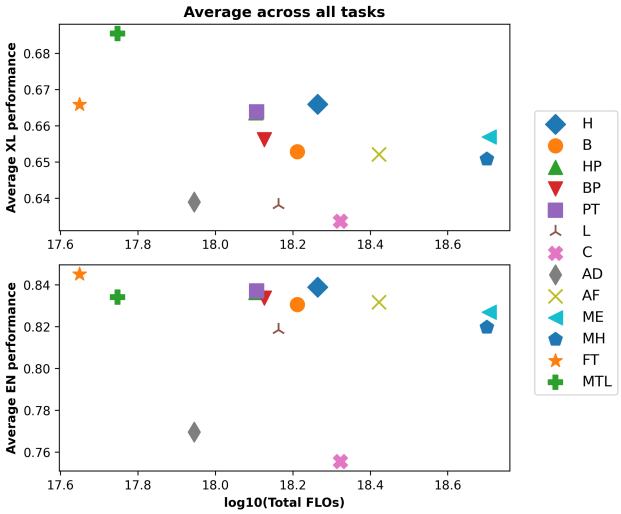

Figure 1: A comparison of 10 different adapters with simpler baselines like full fine-tuning (FT) and multi-task learning (MTL). In the top figure the y-axis shows the zero-shot performance averaged across all tasks and all languages. In the bottom figure, the y-axis shows the En performance averaged across all tasks. The abbreviations used are-'**H**' - Houlsby, '**B**' - Bapna, '**HP**' - Houlsby Parallel[1], '**BP**'- Bapna Parallel, '**PT**'- Prefix Tuning, '**L**'- LoRA, '**C**' - Compacter, '**AD**'- Adapter Drop, '**AF**' - Adapter Fusion, '**ME**' - MADX-en, '**MH**' - MADX-hi, '**FT**' - Fine-tuning, '**MTL**'- Multi-task-learning.

1. Introduction

Pretraining followed by fine-tuning (Devlin et al., 2019; Liu et al., 2019b) is the most commonly used paradigm in NLP, but as pre-trained models grow in size, fine-tuning the entire model (full fine-tuning) becomes costly. Maintaining a copy of the model for each task is costly, and parameter efficient fine-tuning (PEFT) with adapters (Houlsby et al., 2019) has become an active area of research that focuses on fine-tuning a minimal number of parameters while still achieving comparable performance as of full fine-tuning.

Efficiency in the adapters parameter alone is not sufficient to assess overall efficiency. Several other factors, including the number of parameters and convergence steps, significantly impact latency and compute efficiency. Often, the compute efficiency of adapters in handling multiple tasks is overlooked, emphasizing the importance of addressing the following question: ***What are adapters really efficient at?***

We recommend that to answer this question one should look

[1]Department of Computer Science, Indian Institute of Technology Madras, Chennai, India [2]AI4Bharat, Chennai, India [3]National Institute of Information and Communications Technology, Kyoto, Japan [4]Microsoft, Hyderabad, India [5]Institute for Infocomm Research (I^{2}R), A*STAR Singapore, Singapore . Correspondence to: Nandini Mundra <cs21s041@cse.iitm.ac.in>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]HP is overlapped by PT in this figure.

beyond the number of parameters and consider other indicators of efficiency, such as, (i) training time and compute (FLOs 3.4), (ii) deployability via inference latency (iii) and maintainability. Existing studies have looked at one or more of the above metrics but a thorough study comparing multiple popular adapters on different tasks across languages, especially in a cross-lingual setting, is missing. Comparing against multi-task learning (MTL)(Liu et al., 2019a) is crucial as it serves as a simpler baseline, yet most adapter works lack this comparison, hindering a comprehensive understanding of adapter utility.

In this work, we try to build a clearer picture by experimenting with 10 different adapters and 6 Natural Language Understanding (NLU) tasks spanning 11 Indian languages. Our focus is on zero-shot transfer, fine-tuning models solely on English training data. Our work also lays down a framework for evaluating adapters along multiple dimensions. Our key finding (Figure 1) is that while adapters are parameter-efficient, they are not as good as full fine-tuning and multi-task learning, where the latter two have better training and deployment efficiency.

2. Related Work

Parameter Efficient Fine-Tuning (PEFT): Zoph et al. (2016) pioneered PEFT, showcasing its benefits in reducing memory requirements and preventing overfitting. This led to the development of Adapters (Houlsby et al., 2019; Bapna & Firat, 2019; Hu et al., 2022), Learnable prompts (Li & Liang, 2021), compacting methods focused on reducing the size of adapters such as compacters (Mahabadi et al., 2021) and IA^3 (Liu et al., 2022) and transfer-focused approaches like AdapterFusion (Pfeiffer et al., 2021), and MAD-X (Pfeiffer et al., 2020b). However, These works prioritize parameter efficiency while overlooking training time, deployability, maintainability, and cross-lingual transfer effectiveness. AdapterDrop (Rücklé et al., 2021) proposes to reduce adapter training time but ignores other aforementioned aspects, a gap which we fill in this paper. Chen et al. (2022) demonstrated the instability of PEFT across weight initialization, training time, and training data order, while comparing its performance to fine-tuning on various dataset sizes. In addition to focusing on the observation that fine-tuning cannot be fully replaced by PEFT, our study has also demonstrated that multi-task learning can be an alternative to the PEFT method.

Multilingual Pre-trained Models: The introduction of BERT (Devlin et al., 2019) revolutionized downstream NLP tasks, followed by multilingual models like XLM-R (Conneau et al., 2020), and language group specific models IndicBERT (Doddapaneni et al., 2022; Kakwani et al., 2020), IndoBERT (Koto et al., 2020), and AfriBerta (Ogueji et al., 2021), etc.

Multi-Task Learning (MTL): MTL focuses on fully fine-tuning one model for multiple tasks (Caruana, 1993) but has only recently seen significant adoption (Wei et al., 2021; Muennighoff et al., 2022). MTL benefits from cross-task transfer, which we also analyzed in this paper (§4). A general overview of MTL in deep learning can be found in Ruder (2017) and Zhang et al. (2022).

3. Experimental Setup

We now describe the fine-tuning approaches, tasks, datasets, languages, pre-trained models, and training settings.

3.1. Fine-Tuning Methodologies

We experimented with the following approaches Full Fine-Tuning (Devlin et al., 2019), Multi-Task-Learning(MTL) (Liu et al., 2019a) which is similar to full fine-tuning, except that it uses a shared encoder for all tasks, with each task having a task-specific "head", Houlsby Adapter (Houlsby et al., 2019), Bapna Adapter (Bapna & Firat, 2019), LoRA (Hu et al., 2022), Compacter (Mahabadi et al., 2021), Prefix-Tuning (Li & Liang, 2021), MAD-X (Pfeiffer et al., 2020b), AdapterFusion (Pfeiffer et al., 2021), AdapterDrop (Rücklé et al., 2021). While LoRA and prefix-tuning are not originally considered as adapters, He et al. (2022) have shown that they can be reformulated as adapters and thus all PEFT approaches we study in this paper are essentially adapters.

3.2. Tasks, Datasets and Languages

We focus on 6 cross-lingual natural language understanding tasks from the IndicXTREME benchmark (Doddapaneni et al., 2022) and XTREME benchmark (Hu et al., 2020b). We give an overview in Appendix Table 5, our training and validation are conducted on English data, while evaluation includes English test sets (supervised/in-language) and Indian language test sets in IndicXTREME (zero-shot). For specific information on tasks and languages, please see Appendix A.1.

3.3. Pre-Trained Models

We mainly experiment with IndicBERT v2 (Doddapaneni et al., 2022). We also perform ablations with the `BASE` and `LARGE` versions of XLM-R (Conneau et al., 2020) on the chosen subset of languages.

3.4. Training Details

We have reported FLOs, which represent the total number of floating-point operations, to assess the computational efficiency when comparing different methods. For training details please refer to the Appendix A.2

Table 1: Comparison on **in-language (train and test on English)** performance of FT and adapters for IndicBERT. We report F1 scores for CoNLL-2003 & SQuAD, and accuracy for the other tasks. The abbreviation "AMR" refers to the Amazon Multilingual Review Dataset. The last three columns show the percent increase in FLOs, inference time, and the number of fine-tuned parameters compared to full fine-tuning respectively. Here, "best method # " reports the best performing row for the respective task and "best adapter # " reports the best performing adapter for the respective task.

#	Method	AMR	XNLI	COPA	PAWS	CoNLL 2003	SQuAD	Avg.	% ↑ FLOs	% ↑ Inference time	% ↑ #Param.
1	Houlsby	**94.0**	82.4	61.5	92.3	91.5	81.7	83.9	311.7	44.0	0.9
2	Bapna	93.3	81.9	59.9	91.4	91.0	80.9	83.1	264.7	28.3	0.5
3	Houlsby Parallel	93.1	82.5	61.4	90.6	92.2	82.0	83.6	185.1	41.5	0.9
4	Bapna Parallel	93.1	82.7	60.5	91.3	91.1	81.4	83.4	199.9	21.2	0.5
5	Prefix Tuning	93.8	82.6	61.1	92.2	91.5	81.0	83.7	186.5	33.8	3.8
6	LoRA	93.4	80.3	57.4	90.2	90.4	79.5	81.8	226.2	23.1	0.3
7	Compacter	92.8	74.8	50.8	72.7	89.2	73.0	75.5	371.4	100.5	0.2
8	Adapter Drop	92.7	80.6	52.3	75.0	90.4	70.7	77.0	97.6	27.5	0.7
9	Adapter Fusion	93.2	79.9	59.9	92.2	92.0	81.9	83.2	492.5	178.1	7.9
10	MAD-X - en	93.6	82.1	56.9	91.0	91.5	81.1	82.7	1042.5	56.6	1.1
11	MAD-X - hi	93.0	79.3	58.4	90.6	91.1	79.4	82.0	1025.7	56.6	1.1
	Best Adapter #	1	4	1	1	3	3	1	8	2	7
12	FT	93.8	**83.0**	**62.3**	**93.0**	**92.8**	**82.1**	**84.5**	-	-	-
13	MTL	93.5	80.9	61.4	91.5	91.0	**82.1**	83.4	20.2	0.0	0.0
	Best method #	1	12	12	12	12	12, 13	12	12	12,13	12,13

4. Results

Tables 1 and 2 respectively show the in-language (train and test on English) and cross-lingual (train on English and test on Indic) results averaged across Indic languages. See Appendix A.3 for additional results.

Adapters are parameter-efficient, but no single adapter is best: It is clear that there is no single adapter that performs best in all the tasks. This observation holds true in both in-language and cross-lingual settings, where one method performs best in the in-language setting but might not be the best in the cross-lingual setting. Compacter and LORA consistently give the lowest performance, possibly due to the small number of parameters they fine-tune (they add only 0.2% - 0.3% tunable parameters to the model). On the other hand, Adapter Fusion, Prefix Tuning, and MADX add between 1.1% to 7.9% tunable parameters but still perform poorly as compared to the Houlsby adapter, which only adds 0.9% parameters. In general, we recommend the Houlsby adapter as it tends to perform well across multiple tasks and languages on average.

Full fine-tuning is the fastest by a significant margin. While adapters, although *parameter* efficient, consume more FLOs to achieve comparable or worse performance comparable to full fine-tuning. AdapterDrop (row 8 in Table 1) shows the least increase in FLOs (97.6%) but suffers from reduced performance, while MAD-X (rows 10, 11) is the costliest (1042.5%-1025.7%) but still under performs compared to full fine-tuning. The best performing adapter (Houlsby, row 1) is also computationally expensive. In the Appendix, we provide the percentage increases in FLOs compared to full fine-tuning (Table 7), as well as task-specific convergence details and absolute FLOs (Tables 8).

MTL is a cost-efficient alternative to adapters, with only 20% more FLOs than full fine-tuning while achieving performance comparable to the best adapter approaches (Houlsby - 83.9% & MTL - 83.4%). Further, MTL exhibits the best average cross-lingual performance with respect to adapters as well as full fine-tuning. It should be noted that MTL significantly benefits the paraphrasing task via cross-task transfer, exhibiting a performance increase of 16.9% accuracy over full fine-tuning in a cross-lingual setting (experiments in further sections show that paraphrasing benefits from the NLI task). Thus, if the full set of tasks to be supported is known *a priori*, MTL is simpler and equivalent to adapters in downstream performance, while being more cost-efficient. Sanh et al. (2022) show that MTL enables zero-shot task generalization, further enhancing the attractiveness of MTL over adapters.

Inference overhead of adapters is shown in Table 1 for different approaches compared to full fine-tuning. MTL does not add any overhead over full fine-tuning since no new parameters are added to the model. On the other hand, adapters have a non-trivial overhead in inference time due to additional parameters. The Bapna parallel and LoRA methods show least increase in inference time (of 21.2% and 23.1%,

Table 2: Comparison on **cross-lingual (train on English test on Indic)** performance of FT and adapters for IndicBERT. We report F1 scores for Naamapadam & IndicQA, and accuracy for the other tasks. Here, "best method # " reports the best performing row for the respective task and "best adapter # " reports the best performing adapter for the respective task.

#	Method	Indic Sentiment	Indic XNLI	Indic COPA	Indic XPara	Naama-padam	IndicQA	Avg.
1	Houlsby	89.7	72.9	64.1	57.4	65.5	**50.0**	66.6
2	Bapna	89.0	72.1	60.9	55.9	65.2	48.6	65.3
3	Houlsby Parallel	90.3	72.4	63.7	55.8	66.7	49.2	66.4
4	Bapna Parallel	89.9	72.5	61.4	56.3	64.7	48.9	65.6
5	Prefix Tuning	88.2	**73.5**	**65.3**	55.8	**67.1**	48.4	66.4
6	Lora	85.7	70.7	60.7	55.0	63.3	47.4	63.8
7	Compacter	88.5	69.9	63.2	50.8	61.3	46.4	63.4
8	Adapter Drop	87.8	72.0	61.8	52.9	64.4	44.4	63.9
9	Adapter Fusion	89.3	70.8	59.3	56.3	66.9	48.7	65.2
10	MAD-X - en	89.6	72.4	62.6	55.9	66.0	47.6	65.7
11	MAD-X - hi	88.6	70.8	63.1	56.5	64.1	47.4	65.1
	Best Adapter #	3	5	5	1	5	1	1
12	FT	**90.9**	72.9	62.5	57.3	66.7	49.3	66.6
13	MTL	90.2	70.7	**65.3**	**74.3**	65.3	45.5	**68.6**
	Best method #	12	5	5, 13	13	5	1	13

respectively), since they are parallel adapters. Bapna parallel has lesser inference time than Houlsby parallel as it has almost half the number of parameters. The adapter fusion method has the highest inference time as it combines all six task adapters and has an additional fused layer. It also has the maximum number of additional parameters. Although Compacter has the least number of parameters, its inference time is 100.5% more than fine-tuning because the compact low-rank hypercomplex weight matrices are converted to high-rank ones via the Kronecker product. These high-rank matrices are actually used during the forward pass and this two-step process slows down inferencing[2].

Maintainability and Extensibility The primary advantage of adapters is the ability to 'plug-and-play' modules, thus making it easy to extend a pre-trained model to new tasks without having to make a copy for the new task or impacting performance on other tasks. This reduces memory requirements at inference time and makes the system more modular, maintainable and extensible. We have already seen that MTL models offer the same performance with no additional parameters and at a lower computational cost compared to adapters. To see if they can also be easily extensible, we experiment with the following setup.

We hold out one task (the *target task*) and fine-tune the pre-trained model on the remaining tasks (resulting in model $\mathbf{MTL}_{-1}$). Next, we continue fine-tuning the model on the target task as well as 10% data from the tasks the model has already seen. A sample from the older tasks is included in the fine-tuning mix to avoid catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999). For comparison, we also perform continued fine-tuning on the **target task only** (model: $\mathbf{MTL}_{+tgt}$) as well as fine-tuning on all available tasks (model: **MTL**).

The results of these experiments are shown in Table 3 for cross-lingual settings (and Table 9 in Appendix for in-language settings). We see that the target task's performance is comparable to both full fine-tuning and MTL with all tasks. Thus, new tasks can be added to an existing MTL model while retaining the same performance as full FT or MTL. Moreover, we see that the $\mathbf{MTL}_{+tgt+old}$ model also retains performance for the older tasks. We also see that if sample data from the already supported tasks is not used, the model suffers from catastrophic forgetting (model: $\mathbf{MTL}_{+tgt}$). Thus, a simple adaptation of MTL can support multiple tasks in an extensible manner.

The fine-tuning computational cost for $\mathbf{MTL}_{+tgt+old}$ is the sum of computational costs for (a) fine-tuning $\mathbf{MTL}_{-1}$ and (b) continued fine-tuning required to extend model for the target task. In Table 3, column "%↑FLOs" reports the percentage increase in total FLOs(sum of (a) and (b)) with respect to total fine-tuning FLOs(i.e. Fine-tuning FLOs sum over all task). As observed, holding out sentiment task, and then continual learning of sentiment task along with 10% data of existing tasks takes only 2.3% more relative FLOs. The maximum cost is taken by NER task with 68.2% more relative FLOs. Holding out one task and then adding the held out task on an average takes 27.4% more relative FLOs,

[2]The current implementation does not pre-compute the high-rank matrices and thus there is a possibility of reducing the inference time of Compacter, although it will not be faster than the Houlsby adapter to which it is architecturally similar.

Table 3: This table reports **cross-lingual (train on English test on Indic)** performance for maintainability of MTL. "Target task" is held out task i.e. pre-trained IndicBERT model is fine-tuned on the remaining 5 task representing MTL_{-1} model. $MTL_{+tgt+old}$ represents continual fine-tuning of the MTL_{-1} model on the target task dataset and 10% of the existing task dataset. MTL_{+tgt} represents continual fine-tuning of the MTL_{-1} model on the target task dataset. "Avg -1 " reports the cross-lingual performance averaged over the tasks included in MTL_{-1} step. "Avg" reports the cross-lingual performance averaged over all 6 task. Here, column "%↑FLOs" reports the relative percent increase in the total computation cost for adding all 6 task to the model with respect to the total computation cost of fine-tuning. Here, text bold indicates the best value in the column and colored cell represent MTL is performing better than the Best Adapter method.

Target Task	Step	Indic Sentiment	Indic XNLI	Indic COPA	Indic XPara	Naama-padam	IndicQA	Avg -1	Avg	% ↑ FLOs
Baseline	Full FT	**90.9**	**72.9**	62.5	57.3	66.7	49.3	-	66.6	-
	MTL	88.5	71.2	64.9	74.0	65.8	45.4	-	68.3	20.2
Best Adapter	Houlsby	89.7	**72.9**	64.1	57.4	65.5	**50.0**	-	66.6	311.7
Sentiment	MTL_{-1}	-	71.5	64.8	74.8	65.1	46.9	64.6	-	-
	$MTL_{+tgt+old}$	90.2	70.8	61.9	72.9	66.1	48.8	64.1	**68.4**	2.3
	MTL_{+tgt}	89.1	54.9	52.2	67.3	40.4	34.9	50.0	56.5	1.7
XNLI	MTL_{-1}	90.5	-	**67.8**	56.7	63.6	47.4	65.2	-	-
	$MTL_{+tgt+old}$	90.8	71.2	63.6	68.7	59.6	48.3	66.2	67.0	20.5
	MTL_{+tgt}	86.0	70.5	64.5	73.3	56.2	15.1	59.0	60.9	12.1
COPA	MTL_{-1}	88.8	72.3	-	73.7	65.0	48.4	69.7	-	-
	$MTL_{+tgt+old}$	88.3	69.7	65.6	74.8	65.4	43.9	68.4	67.9	15.3
	MTL_{+tgt}	89.5	66.4	66.0	75.5	62.7	46.4	68.1	67.7	9.3
Paraphrase	MTL_{-1}	86.0	70.2	64.4	-	65.0	45.0	66.1	-	-
	$MTL_{+tgt+old}$	87.4	69.8	64.2	**77.8**	65.0	45.4	66.4	68.3	32.4
	MTL_{+tgt}	81.1	66.0	64.4	73.1	30.1	42.5	56.8	59.5	24.1
NER	MTL_{-1}	88.0	72.5	65.7	77.3	-	47.7	70.3	-	-
	$MTL_{+tgt+old}$	86.7	71.2	64.4	76.3	65.2	45.1	68.7	68.1	68.2
	MTL_{+tgt}	83.8	67.9	62.3	57.3	**68.5**	39.8	62.2	63.2	59.8
QA	MTL_{-1}	89.2	72.3	64.9	74.9	65.4	-	73.3	-	-
	$MTL_{+tgt+old}$	85.9	71.1	63.9	75.7	62.3	46.8	71.8	67.6	25.8
	MTL_{+tgt}	84.9	68.2	65.9	66.9	23.7	46.6	61.9	59.4	21.2

while adding all tasks at once takes 20.2% more relative FLOs. Nonetheless, this is still more cost-effective than the best-performing adapter methods. For instance, the Houlsby adapter requires around 311% more computation compared to full fine-tuning. Thus, we see maintainability of MTL cost-effective. However, average cross-lingual performance for MTL maintainability (as shown in Table 3), is slightly inflated due to the inclusion of the paraphrase task. If the average MTL performance is calculated without the paraphrase task (i.e. only considering the remaining five tasks), a slight decrease in performance is observed.

Effect of model size To further study the effect of model size on different adapters, we experiment with two different pre-trained models trained on the same pretraining data but differing only in model size. Specifically, we compare the XLMR-base and XLMR-large models (Conneau et al., 2020) which have 270M and 550M parameters, respectively. We evaluate the adapters on the XNLI, XQuAD and NER tasks from the XTREME benchmark (Hu et al., 2020a). We use the English dataset for training and test the cross-lingual zero-shot performance on 14 languages for XNLI and WikiANN and 11 languages for XQuAD. The results are shown in Table 4. We can see that as the model size increase, the adaptation time relative to full fine-tuning time reduces. Thus, for large language models, we might see a trend of adapters being increasingly cost-efficient. In fact, recent work on large language models have shown adapters to be promising (Yong et al., 2022). However, larger models still need heavy compute and deploying them is still challenging. In this case, there is a line of work that distills LLMs which can then be fine-tuned (Ganesan et al., 2021). Given that adapters do not have much compute efficiency in smaller models, full fine-tuning or MTL are excellent contenders.

Key takeaway Fig 1 shows a unified summary of task performance and fine-tuning compute required for the various approaches discussed in the paper. Summarizing observations previously discussed, we see that MTL outperforms or is comparable to all adapters in in-language and cross-

Table 4: Comparison on **cross-lingual** performance of FT and adapters for XLMR-Base and XLMR-Large model. "Avg." reports the average cross-lingual perfromance across all task. "%↑FLOs" reports the relative increase in FLOs with respect to fine-tuning.

XL	XLMR-Base					XLMR-Large				
Method	**NER**	**XNLI**	**QA**	**Avg.**	**%↑FLOs**	**NER**	**XNLI**	**QA**	**Avg.**	**%↑FLOs**
Houlsby	61.0	72.6	**72.5**	**68.7**	484.3	**64.6**	76.2	**78.6**	73.1	200.5
Bapna	58.3	71.3	71.0	66.8	547.1	64.3	76.7	78.0	73.0	139.9
Houlsby parallel	59.2	72.8	71.2	67.8	197.0	65.3	78.7	77.8	73.9	143.3
Bapna parallel	57.1	70.3	69.7	65.7	409.1	63.1	**78.8**	77.6	73.2	168.6
Prefixtuning	58.5	69.9	68.8	65.7	256.5	64.7	78.7	77.6	73.7	287.3
LORA	58.6	70.5	68.4	65.8	734.7	62.3	76.9	77.1	72.1	270.0
Compacter	55.1	66.8	64.1	62.0	805.3	58.5	76.4	75.3	70.1	490.1
Adapter drop	60.5	70.2	71.3	67.3	345.1	64.6	**78.8**	78.5	**74.0**	214.1
FT	**61.7**	**73.7**	70.8	**68.7**	-	63.9	77.0	78.0	73.0	-

language zero-shot settings (particularly for smaller models). Hence, we recommend that MTL should be considered as an alternative to adapters in constrained scenarios where relatively smaller models are preferred, computational budgets are limited and extensibility is important.

5. Conclusion

In this paper, we have conducted a comprehensive analysis of adapters across different languages and tasks to evaluate their advantages in terms of training/deployment efficiency and maintainability/extensibility. We compared adapters with simpler baseline methods, including fine-tuning and multi-task learning, in supervised/in-language as well as zero-shot cross-lingual settings, and found that these simpler methods are more computationally efficient and have better deployment efficiency, while achieving the comparable performance as that of adapters. Additionally, we conducted extensive experiments to show that multi-task learning is a relatively more cost-effective alternative to the adapters in terms of maintainability, as it allows the model to be extended for new tasks at a lower cost than adapters. Therefore, we suggest that simpler baselines be used for moderately sized models, as they are more efficient than adapters.

Acknowledgements

We would like to thank the Ministry of Electronics and Information Technology[3] of the Government of India for their generous grant through the Digital India Bhashini project[4]. We also thank the Centre for Development of Advanced Computing[5] for providing compute time on the Param Siddhi Supercomputer. We also thank Nilekani Philanthropies for their generous grant towards building datasets, models, tools and resources for Indic languages. We also thank Microsoft for their grant to support research on Indic languages.

[3] https://www.meity.gov.in/

[4] https://www.bhashini.gov.in/

[5] https://www.cdac.in/index.aspx?id=pune

Limitations

We identify the following limitations of our work:

- Our study is limited to NLU and some of our observations might not apply in Natural Language Generation (NLG) settings. While for NLU cross-lingual transfer through full fine-tuning is as effective as adapters, in NLG full fine-tuning for zero-shot cross-lingual NLG is unreliable due to the risk of catastrophic forgetting. Therefore, adapters might be more important for NLG (Vu et al., 2022).

- We primarily focus on smaller pre-trained models because larger models require significant computing resources that not everyone may have access to, and therefore, our findings may not be applicable to larger models with billions of parameters. However, active research on compressing pre-trained models indicates that fine-tuning compact pre-trained models will remain a significant area of research.

- Our analysis focus on 6 NLU tasks, which is relatively fewer compared to the total number of tasks in benchmarks such as BIG-Bench (bench authors, 2023). Although focusing on a larger number of tasks will increase the credibility of our studies, our focus on cross-lingual performance means that we are currently limited by the availability of benchmarking data in other languages for these large number of tasks.

Ethics Statement

All of the datasets used in this study were publicly available, and no annotators were employed for data collection. We confirm that the datasets we used did not contain any harmful content. We have cited the datasets and relevant works used in this study.

References

Bapna, A. and Firat, O. 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)*, pp. 1538–1548, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1165. URL https://aclanthology.org/D19-1165.

bench authors, B. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

Caruana, R. Multitask learning: A knowledge-based source of inductive bias. In Utgoff, P. E. (ed.), *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pp. 41–48. Morgan Kaufmann, 1993. doi: 10.1016/b978-1-55860-307-3.50012-5. URL https://doi.org/10.1016/b978-1-55860-307-3.50012-5.

Chen, G., Liu, F., Meng, Z., and Liang, S. Revisiting parameter-efficient tuning: Are we really there yet? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2612–2626, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.168.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Doddapaneni, S., Aralikatte, R., Ramesh, G., Goyal, S., Khapra, M. M., Kunchukuttan, A., and Kumar, P. Towards leaving no indic language behind: Building monolingual corpora, benchmark and models for indic languages. *CoRR*, abs/2212.05409, 2022. doi: 10.48550/arXiv.2212.05409. URL https://doi.org/10.48550/arXiv.2212.05409.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. ISSN 1364-6613. doi: https://doi.org/10.1016/S1364-6613(99)01294-2. URL https://www.sciencedirect.com/science/article/pii/S1364661399012942.

Ganesan, V., Ramesh, G., and Kumar, P. Supershaper: Task-agnostic super pre-training of BERT models with variable hidden dimensions. *CoRR*, abs/2110.04711, 2021. URL https://arxiv.org/abs/2110.04711.

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0RDcd5Axok.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hu, J., Ruder, S., Siddhant, A., Neubig, G., Firat, O., and Johnson, M. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *CoRR*, abs/2003.11080, 2020a. URL https://arxiv.org/abs/2003.11080.

Hu, J., Ruder, S., Siddhant, A., Neubig, G., Firat, O., and Johnson, M. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization, 2020b.

Kakwani, D., Kunchukuttan, A., Golla, S., N.C., G., Bhattacharyya, A., Khapra, M. M., and Kumar, P. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4948–4961, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.445. URL https://aclanthology.org/2020.findings-emnlp.445.

Keung, P., Lu, Y., Szarvas, G., and Smith, N. A. The multilingual amazon reviews corpus. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 4563–4568. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.369. URL https://doi.org/10.18653/v1/2020.emnlp-main.369.

Koto, F., Rahimi, A., Lau, J. H., and Baldwin, T. IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian NLP. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 757–770, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.66. URL https://aclanthology.org/2020.coling-main.66.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *CoRR*, abs/2205.05638, 2022. doi: 10.48550/arXiv.2205.05638. URL https://doi.org/10.48550/arXiv.2205.05638.

Liu, X., He, P., Chen, W., and Gao, J. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4487–4496, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1441. URL https://aclanthology.org/P19-1441.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019b. URL http://arxiv.org/abs/1907.11692.

Mahabadi, R. K., Henderson, J., and Ruder, S. Compacter: Efficient low-rank hypercomplex adapter layers. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 1022–1035, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/081be9fdff07f3bc808f935906ef70c0-Abstract.html.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Mhaske, A., Kedia, H., Doddapaneni, S., Khapra, M. M., Kumar, P., Murthy, V. R., and Kunchukuttan, A. Naamapadam: A large-scale named entity annotated data for indic languages. *CoRR*, abs/2212.10168, 2022. doi: 10.48550/arXiv.2212.10168. URL https://doi.org/10.48550/arXiv.2212.10168.

Muennighoff, N., Wang, T., Sutawika, L., Roberts, A., Biderman, S., Scao, T. L., Bari, M. S., Shen, S., Yong, Z.-X., Schoelkopf, H., Tang, X., Radev, D., Aji, A. F., Almubarak, K., Albanie, S., Alyafeai, Z., Webson, A., Raff, E., and Raffel, C. Crosslingual generalization through multitask finetuning, 2022.

Ogueji, K., Zhu, Y., and Lin, J. Small data? no problem! exploring the viability of pretrained multilingual language models for low-resourced languages. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pp. 116–126, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.mrl-1.11. URL https://aclanthology.org/2021.mrl-1.11.

Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., and Gurevych, I. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods*

in Natural Language Processing: System Demonstrations, pp. 46–54, Online, October 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.7. URL https://aclanthology.org/2020.emnlp-demos.7.

Pfeiffer, J., Vulić, I., Gurevych, I., and Ruder, S. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7654–7673, Online, November 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.617. URL https://aclanthology.org/2020.emnlp-main.617.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., and Gurevych, I. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7930–7946, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.626. URL https://aclanthology.org/2021.emnlp-main.626.

Ruder, S. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Scao, T. L., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Sap, M., Rashkin, H., Chen, D., Le Bras, R., and Choi, Y. 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)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454.

Tjong Kim Sang, E. F. and De Meulder, F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003. URL https://aclanthology.org/W03-0419.

Vu, T., Barua, A., Lester, B., Cer, D., Iyyer, M., and Constant, N. Overcoming catastrophic forgetting in zero-shot cross-lingual generation. *CoRR*, abs/2205.12647, 2022. doi: 10.48550/arXiv.2205.12647. URL https://doi.org/10.48550/arXiv.2205.12647.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners, 2021.

Williams, A., Nangia, N., and Bowman, S. 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, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Yang, Y., Zhang, Y., Tar, C., and Baldridge, J. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. 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), pp. 3687–3692, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1382. URL https://aclanthology.org/D19-1382.

Yong, Z. X., Schoelkopf, H., Muennighoff, N., Aji, A. F., Adelani, D. I., Almubarak, K., Bari, M. S., Sutawika, L., Kasai, J., Baruwa, A., Winata, G. I., Biderman, S., Radev, D., and Nikoulina, V. BLOOM+1: adding language support to BLOOM for zero-shot prompting. *CoRR*, abs/2212.09535, 2022. doi: 10.48550/arXiv.2212.09535. URL https://doi.org/10.48550/arXiv.2212.09535.

Zhang, Z., Yu, W., Yu, M., Guo, Z., and Jiang, M. A survey of multi-task learning in natural language processing: Regarding task relatedness and training methods. *CoRR*, abs/2204.03508, 2022. doi: 10.48550/arXiv.2204.03508. URL https://doi.org/10.48550/arXiv.2204.03508.

Zoph, B., Yuret, D., May, J., and Knight, K. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1568–1575, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1163. URL https://aclanthology.org/D16-1163.

Table 5: A summary of the tasks and datasets used. |Test| denotes the size of Test Data. |Train| is the size of English training sets. |Lang| denotes the number of languages for which we have evaluated its cross-lingual performance.

Task Category	Train Data	Test Data	|Train|	|Test|	|Lang|	Metric
Sentence Classification	Amazon Multi Reviews	IndicSentiment	160k	1000	11	Acc.
	MultiNLI	IndicXNLI	392k	5000	11	Acc.
	SocialIQA	IndicCOPA	33k	500	11	Acc.
	PAWS	IndicParaphrase	49k	2002	10	Acc.
Token Classification	CoNLL-2003	Naamapadam	11k	607-1080	11	F1
Question Answering	SQuAD	IndicQA	87k	1517-2017	11	F1

A. Appendices

A.1. Details of Tasks and Languages

Sentence Classification tasks are Natural Language Inference (NLI), sentiment classification, paraphrase detection and Choice Of Plausible Alternatives (COPA). For NLI we use the MultiNLI (Williams et al., 2018) dataset for training and test performance on IndicXNLI for 11 languages. For sentiment classification, we train on the Amazon Multilingual Reviews (AMR) dataset (Keung et al., 2020) and test on IndicSentiment for 11 languages. For paraphrase detection, we train on the PAWS-X (Yang et al., 2019) dataset and test on IndicXParaphrase for 10 languages. For the COPA task, which involves selecting one of two alternatives that more plausibly has a causal relation with a given premise, we train on SocialIQA (Sap et al., 2019) and test on IndicCOPA for 11 languages.

Token Classification task uses the CoNLL-2003 (Tjong Kim Sang & De Meulder, 2003) dataset for training and Naamapadam (Mhaske et al., 2022) for testing for 11 languages.

Question Answering We use the SQuAD (Rajpurkar et al., 2016) data for training and test on the IndicQA benchmark (Doddapaneni et al., 2022) available in 11 Indian languages.

A.2. training details

All models are trained with Adapter-hub (Pfeiffer et al., 2020a). All experiments are performed on Nvidia A100-SXM4 40GB GPUs and the results are reported by doing single run. We use the recommended/default settings in Adapter-hub but wherever possible, we performed hyperparameter tuning on the development set to determine optimal hyperparameters. Table 6 gives the search space and best performing hyperparameters for Houlsby, Bapna, LoRA and Prefix-Tuning.

For MAD-X, we have used the default configuration as in Adapter-hub for both language and task adapters, as shown in Table 6. For Adapter-fusion we have trained each task adapter in ST-A (single task adapter) style (Pfeiffer et al., 2021).

For all the tasks using the IndicBERT model, we train models for a maximum of 50 epochs with an early stopping patience of 3 epochs. We use 2,000 warmup steps for all tasks and settings, except for MTL, where we use 20,000 warmup steps due to the increased size of the training data. For a fair comparison across all settings, we use a batch size of 32 examples with a learning rate of 3e-5 and weight decay of 0.1. For MTL, we found that a weight decay of 0.01 gave the best results. For all the experiments FLOs reported are provided by the HF transformers library (Wolf et al., 2020).

Pretraining MAD-X language adapter is done using the IndicCorp v2 (Doddapaneni et al., 2022) dataset with MLM objective for the 11 Indic languages and English with 6.5M sentences sampled per language.

A.3. Task-level sensitivity

The efficiency of training is also affected by the task, as shown in Table 7, where the QA task requires relatively more FLOs compared to the paraphrase task. However, across all tasks the trend remains the same.

Table 6: This table reports the optimal reduction factor (r), prefix length (l) and LoRA α we have set for adapters. For those not listed in this table, we have used the default AdapterHub configurations.

Method	Hyperparameter	Search Space
Houlsby	$r = 16$	$r = 2, 4, 8, 16$
Bapna	$r = 16$	$r = 2, 4, 8, 16$
LoRA	$r = 8, \alpha = 16$	$r = 2, 4, 8, 16$
Prefix-Tuning	$l = 30$	$l = 10, 20, 30, 40, 50$

Table 7: This table reports percentage increase of FLOs for several adapters across tasks with respect to full fine-tuning on IndicBert model. Column "Total" reports the percentage increase in total FLOs for each method with respect to full fine-tuning (FLOs are added across all tasks). Since, for Adapter Fusion and MAD-X, task adapters and language adapters, respectively, are shared across tasks, training FLOs are also shared across tasks. Thus, for these two approaches, FLOs cannot be reported accurately for individual tasks.

Method	Sentiment	XNLI	COPA	Paraphrase	NER	QA	Total
Houlsby	249.8	208.5	376.6	88.5	19.8	599.0	311.7
Bapna	185.2	246.5	321.0	43.6	77.7	456.7	264.7
Houlsby Parallel	105.3	208.5	274.2	-5.8	88.0	275.0	185.1
Bapna Parallel	62.9	205.4	185.7	52.6	26.9	389.4	199.9
Prefix Tuning	190.9	237.2	179.4	96.2	77.4	198.1	186.5
Lora	223.2	203.1	168.0	93.6	143.6	402.9	226.2
Compacter	363.9	121.7	650.9	25.9	252.4	735.6	371.4
Adapter Drop	124.3	225.6	136.4	-40.6	19.8	1.9	97.6

A.4. MTL maintainability

MTL is maintainable as discussed in sec 4, as the MTL model can be extended to new tasks by continually learning with the new task's data along with 10% of the existing tasks' data. We analyze the impact of performance and computational cost by changing the percentage of an existing task for continual learning of new task as presented in Table 10 and 11. We tested two additional setups: (a) using 5% data from previously seen tasks (**MTL**$_{-1}$) instead of 10%, as reported in the "MTL$_{+tgt+old_5}$" row and (b) using the minimum of either 10% of the existing task dataset or the new task dataset, reported in the row "MTL$_{+tgt+old+min_{10}}$", and similarly, using the minimum of either 5% of the existing task dataset or the new task dataset, reported in the row "MTL$_{+tgt+old+min_5}$". Our findings show that cross-lingual performance is better when using a higher percentage of the existing task dataset, while in-language performance is better when using a lower percentage of the existing task dataset. In terms of computational efficiency, using 5% of the existing dataset requires fewer FLOs compared to using 10%.

Table 8: The table reports the total FLOS for FT and various adapters on IndicBERT, across each of the tasks. Total corresponds to the total FLOS summed over all the tasks for a particular fine-tuning method.

Method	Sentiment	XNLI	COPA	Paraphrase	NER	QA	Total
Houlsby	1.8E+17	4.0E+17	3.8E+17	1.5E+17	4.2E+15	7.3E+17	1.8E+18
Bapna	1.5E+17	4.5E+17	3.3E+17	1.1E+17	6.2E+15	5.8E+17	1.6E+18
Houlsby Parallel	1.1E+17	4.0E+17	3.0E+17	7.4E+16	6.6E+15	3.9E+17	1.3E+18
Bapna Parallel	8.6E+16	3.9E+17	2.3E+17	1.2E+17	4.4E+15	5.1E+17	1.3E+18
Prefix Tuning	1.5E+17	4.4E+17	2.2E+17	1.5E+17	6.2E+15	3.1E+17	1.3E+18
Lora	1.7E+17	3.9E+17	2.1E+17	1.5E+17	8.5E+15	5.2E+17	1.5E+18
Compacter	2.4E+17	2.9E+17	5.9E+17	9.8E+16	1.2E+16	8.7E+17	2.1E+18
Adapter Drop	1.2E+17	4.2E+17	1.9E+17	4.6E+16	4.2E+15	1.1E+17	8.8E+17
FT	5.3E+16	1.3E+17	7.9E+16	7.8E+16	3.5E+15	1.0E+17	4.5E+17
Total	1.3E+18	3.3E+18	2.5E+18	9.8E+17	5.6E+16	4.1E+18	1.2E+19

Table 9: This table reports **in-language (train and test on English)** performance for maintainability of MTL. "Target task" is held out task i.e. pre-trained IndicBERT model is fine-tuned on the remaining 5 task representing MTL_{-1} model. $\text{MTL}_{+tgt+old}$ represents continual fine-tuning of the MTL_{-1} model on the target task dataset and 10% of the existing task dataset. MTL_{+tgt} represents continual fine-tuning of the MTL_{-1} model on the target task dataset. "Avg -1 " reports the in-language performance averaged over the task included in MTL_{-1} step. "Avg" reports the in-language performance averaged over all 6 task. Here, text bold indicates the best value in the column and colored cell represent MTL is performing better than the Best Adapter method.

Target Task	Step	Amazon Multi Reviews	XNLI	COPA	PAWS	CoNLL2003	SQuAD	Avg -1	Avg
Baseline	Full FT	93.8	**83.0**	62.3	93.0	**92.8**	82.1	-	**84.5**
	MTL (full)	93.5	80.9	61.4	91.5	91.0	82.1	-	83.4
Best Adapter	Houlsby	94.0	82.4	61.5	92.3	91.5	81.7	-	83.9
Sentiment	MTL_{-1}	-	81.6	**63.0**	91.5	92.5	82.5	82.2	-
	$\text{MTL}_{+tgt+old}$	93.1	79.0	60.7	89.0	91.4	81.3	80.3	82.4
	MTL_{+tgt}	93.5	58.6	47.1	71.4	78.3	71.7	65.4	70.1
XNLI	MTL_{-1}	**94.1**	-	60.5	91.5	91.9	82.4	84.1	-
	$\text{MTL}_{+tgt+old}$	92.9	79.0	58.7	88.3	87.7	78.6	81.2	80.9
	MTL_{+tgt}	90.7	79.6	52.8	56.5	85.4	36.4	64.4	66.9
COPA	MTL_{-1}	93.8	81.9	-	91.6	91.0	81.5	88.0	-
	$\text{MTL}_{+tgt+old}$	93.5	79.0	62.5	90.8	90.9	78.7	86.6	82.6
	MTL_{+tgt}	92.4	73.8	62.2	87.8	89.8	79.2	84.6	80.9
Paraphrase	MTL_{-1}	93.9	80.1	62.4	-	91.9	80.9	81.8	-
	$\text{MTL}_{+tgt+old}$	93.9	79.8	60.2	89.7	91.7	80.9	81.3	82.7
	MTL_{+tgt}	92.9	73.9	59.8	92.2	77.4	73.8	75.5	78.3
NER	MTL_{-1}	94.0	82.1	62.2	92.7	-	**82.6**	82.7	-
	$\text{MTL}_{+tgt+old}$	93.4	80.8	61.0	91.2	91.4	81.0	81.5	83.1
	MTL_{+tgt}	93.1	74.1	59.9	74.5	92.1	71.6	74.6	77.6
QA	MTL_{-1}	**94.1**	81.6	62.9	**93.4**	92.0	-	84.8	-
	$\text{MTL}_{+tgt+old}$	93.5	80.0	59.7	91.4	89.9	81.2	82.9	82.6
	MTL_{+tgt}	92.4	77.5	61.0	73.0	63.5	82.5	73.5	75.0

Table 10: Table reports **cross-lingual performance (train on English test on Indic)**. Row $MTL_{+tgt+old_{10}}$ and $MTL_{+tgt+old_5}$ denotes adding 10% and 5% of existing task data combine with new task dataset respectively. $MTL_{+tgt+old+min_{10}}$ denotes combining the existing task dataset size minimum(10% data , target task dataset size) i.e. to ensure the existing task dataset is less or equal to new task dataset when combined. similarly $MTL_{+tgt+old+min_5}$ denote combining the existing task dataset size as minimum(5% data , target task dataset size). "Avg -1 " reports the cross-lingual perfromance averaged over the task included in MTL_{-1} step. "Avg" reports the cross-lingual performance averaged over all 6 task. Here, column "%↑FLOs" reports the relative percent increase in the total computation cost for adding all 6 task with respect to the total computation cost of fine-tuning. Note, we have $MTL_{+tgt+old+min_{10}}$ and $MTL_{+tgt+old+min_5}$ only for NER and COPA dataset, as dataset size for NER and COPA is less. Here, text bold indicates the best value in the column and colored cell represent MTL is performing better than the Best Adapter method.

Target Task	Step	Indic Sentiment	Indic XNLI	Indic COPA	Indic XPara	Naama-padam	IndicQA	Avg -1	Avg	% ↑ FLOs
Baseline	Full FT	**90.9**	**72.9**	62.5	57.3	66.7	49.3	-	66.6	-
	MTL	88.5	71.2	64.9	74.0	65.8	45.4	-	68.3	20.2
Best Adapter	Houlsby	89.7	**72.9**	64.1	57.4	65.5	**50.0**	-	66.6	311.7
Sentiment	MTL_{-1}	-	71.5	64.8	74.8	65.1	46.9	64.6	-	
	$MTL_{+tgt+old_{10}}$	90.2	70.8	61.9	72.9	66.1	48.8	64.1	**68.4**	2.3
	$MTL_{+tgt+old_5}$	90.5	69.1	62.3	74.2	62.2	47.3	63.0	67.6	-0.4
	MTL_{+tgt}	89.1	54.9	52.2	67.3	40.4	34.9	50.0	56.5	1.7
XNLI	MTL_{-1}	90.5	-	**67.8**	56.7	63.6	47.4	65.2	-	
	$MTL_{+tgt+old_{10}}$	90.8	71.2	63.6	68.7	59.6	48.3	66.2	67.0	20.5
	$MTL_{+tgt+old_5}$	90.5	70.6	64.7	61.9	63.5	47.3	65.6	66.4	-2.6
	MTL_{+tgt}	86.0	70.5	64.5	73.3	56.2	15.1	59.0	60.9	12.1
COPA	MTL_{-1}	88.8	72.3	-	73.7	65.0	48.4	69.7	-	
	$MTL_{+tgt+old_{10}}$	88.3	69.7	65.6	74.8	65.4	43.9	68.4	67.9	15.3
	$MTL_{+tgt+old_5}$	90.5	71.1	64.6	73.0	63.7	44.9	68.6	68.0	3.5
	$MTL_{+tgt+old+min_{10}}$	85.5	69.7	66.5	74.6	63.6	45.7	67.8	67.6	10.4
	$MTL_{+tgt+old+min_5}$	90.0	69.8	65.7	73.9	63.8	46.2	68.7	68.2	10.8
	MTL_{+tgt}	89.5	66.4	66.0	75.5	62.7	46.4	68.1	67.7	9.3
Paraphrase	MTL_{-1}	86.0	70.2	64.4	-	65.0	45.0	66.1	-	
	$MTL_{+tgt+old_{10}}$	87.4	69.8	64.2	77.8	65.0	45.4	66.4	68.3	32.4
	$MTL_{+tgt+old_5}$	86.2	70.0	64.8	**78.0**	64.2	42.7	65.6	67.7	25.3
	MTL_{+tgt}	81.1	66.0	64.4	73.1	30.1	42.5	56.8	59.5	24.1
NER	MTL_{-1}	88.0	72.5	65.7	77.3	-	47.7	70.3	-	
	$MTL_{+tgt+old_{10}}$	86.7	71.2	64.4	76.3	65.2	45.1	68.7	68.1	68.2
	$MTL_{+tgt+old_5}$	88.4	70.4	63.9	73.2	65.7	45.7	68.3	67.9	66.6
	$MTL_{+tgt+old+min_{10}}$	87.6	71.2	64.7	73.5	66.2	43.7	68.1	67.8	66.6
	$MTL_{+tgt+old+min_5}$	87.8	71.2	65.4	71.3	65.4	45.8	68.3	67.8	63.1
	MTL_{+tgt}	83.8	67.9	62.3	57.3	**68.5**	39.8	62.2	63.2	59.8
QA	MTL_{-1}	89.2	72.3	64.9	74.9	65.4	-	73.3	-	
	$MTL_{+tgt+old_{10}}$	85.9	71.1	63.9	75.7	62.3	46.8	71.8	67.6	25.8
	$MTL_{+tgt+old_5}$	87.9	71.6	64.4	75.1	65.6	45.0	72.9	68.3	22.1
	MTL_{+tgt}	84.9	68.2	65.9	66.9	23.7	46.6	61.9	59.4	21.2

Table 11: The table reports performance score on in-language (en). Row $MTL_{+tgt+old_{10}}$ and$MTL_{+tgt+old_5}$ denotes adding 10% and 5% of existing task data combine with new task dataset respectively.$MTL_{+tgt+old+min_{10}}$ denotes combining the existing task dataset size minimum(10% data , target task dataset size) i.e. to ensure the existing task dataset is less or equal to new task dataset when combined. similarly $MTL_{+tgt+old+min_5}$ denote combining the existing task dataset size as minimum(5% data , target task dataset size). Here, column "%↑FLOs" reports the relative percent increase in the total computation cost for adding all 6 task with respect to the total computation cost of fine-tuning. "Avg -1 " reports the in-language performance averaged over the task included in MTL_{-1} step. "Avg" reports the cross-lingual performance averaged over all 6 task. Note, we have $MTL_{+tgt+old+min_{10}}$ and $MTL_{+tgt+old+min_5}$ only for NER and COPA dataset, as dataset size for NER and COPA is less. Here, text bold indicates the best value in the column and colored cell represent MTL is performing better than the Best Adapter method.

Target Task	Step	Amazon Multi Reviews	XNLI	COPA	PAWS	CoNLL2003	SQuAD	Avg -1	Avg
Baseline	Full FT	93.8	**83.0**	62.3	93.0	**92.8**	82.1	-	**84.5**
	MTL (full)	93.5	80.9	61.4	91.5	91.0	82.1	-	83.4
Best Adapter	Houlsby	94.0	82.4	61.5	92.3	91.5	81.7	-	83.9
Sentiment	MTL_{-1}	-	81.6	63.0	91.5	92.5	82.5	82.2	-
	$MTL_{+tgt+old_{10}}$	93.1	79.0	60.7	89.0	91.4	81.3	80.3	82.4
	$MTL_{+tgt+old_5}$	93.0	78.7	60.2	91.0	91.7	81.0	80.5	82.6
	MTL_{+tgt}	93.5	58.6	47.1	71.4	78.3	71.7	65.4	70.1
XNLI	MTL_{-1}	**94.1**	-	60.5	91.5	91.9	82.4	84.1	-
	$MTL_{+tgt+old_{10}}$	92.9	79.0	58.7	88.3	87.7	78.6	81.2	80.9
	$MTL_{+tgt+old_5}$	93.1	77.1	59.0	86.1	89.1	79.9	81.4	80.7
	MTL_{+tgt}	90.7	79.6	52.8	56.5	85.4	36.4	64.4	66.9
COPA	MTL_{-1}	93.8	81.9	-	91.6	91.0	81.5	88.0	-
	$MTL_{+tgt+old_{10}}$	93.5	79.0	62.5	90.8	90.9	78.7	86.6	82.6
	$MTL_{+tgt+old_5}$	93.7	80.8	58.6	90.8	91.7	81.1	87.6	82.8
	$MTL_{+tgt+old+min_{10}}$	93.2	79.3	62.2	91.9	91.0	81.1	87.3	83.1
	$MTL_{+tgt+old+min_5}$	93.8	79.6	**63.2**	90.8	90.9	80.6	87.1	83.1
	MTL_{+tgt}	92.4	73.8	62.2	87.8	89.8	79.2	84.6	80.9
Paraphrase	MTL_{-1}	93.9	80.1	62.4	-	91.9	80.9	81.8	-
	$MTL_{+tgt+old_{10}}$	93.9	79.8	60.2	89.7	91.7	80.9	81.3	82.7
	$MTL_{+tgt+old_5}$	94.2	80.3	61.5	92.4	90.8	80.7	81.5	83.3
	MTL_{+tgt}	92.9	73.9	59.8	92.2	77.4	73.8	75.5	78.3
NER	MTL_{-1}	94.0	82.1	62.2	92.7	-	**82.6**	82.7	-
	$MTL_{+tgt+old_{10}}$	93.4	80.8	61.0	91.2	91.4	81.0	81.5	83.1
	$MTL_{+tgt+old_5}$	93.8	80.5	62.3	92.1	91.6	80.4	81.8	83.4
	$MTL_{+tgt+old+min_{10}}$	93.7	79.8	60.5	91.3	92.0	81.8	81.4	83.2
	$MTL_{+tgt+old+min_5}$	93.6	80.3	62.2	91.0	90.8	81.6	81.7	83.2
	MTL_{+tgt}	93.1	74.1	59.9	74.5	92.1	71.6	74.6	77.6
QA	MTL_{-1}	**94.1**	81.6	62.9	**93.4**	92.0	-	84.8	-
	$MTL_{+tgt+old_{10}}$	93.5	80.0	59.7	91.4	89.9	81.2	82.9	82.6
	$MTL_{+tgt+old_5}$	94.0	81.1	62.1	91.2	90.8	81.6	83.8	83.5
	MTL_{+tgt}	92.4	77.5	61.0	73.0	63.5	82.5	73.5	75.0

Table 12: Results on IndicSentiment with IndicBERT. Metric: Accuracy. Column "Avg.XL" reports average cross-lingual zero-shot performance.

Method	en	as	bn	gu	hi	kn	ml	mr	or	pa	ta	te	Avg.XL
Houlsby	**94.0**	87.7	90.7	89.5	92.0	90.5	88.3	89.4	89.7	92.0	88.0	89.0	89.7
Bapna	93.3	87.0	90.1	89.2	91.8	89.1	87.0	88.6	88.5	90.2	88.1	88.9	89.0
Houlsby Parallel	93.1	88.1	**91.8**	90.5	93.0	91.0	88.7	90.3	90.3	91.7	88.2	90.0	90.3
Bapna Parallel	93.1	87.1	91.1	90.1	92.7	90.0	88.5	89.3	90.2	91.2	88.3	90.2	89.9
Prefix Tuning	93.8	85.1	88.9	88.4	91.7	88.7	86.2	89.0	87.7	90.3	85.7	88.9	88.2
Lora	93.4	83.6	86.3	85.5	85.3	85.6	82.3	86.8	87.1	86.2	86.1	88.4	85.7
Compacter	92.8	87.0	90.1	89.0	89.2	89.2	88.1	86.2	88.0	89.6	87.8	89.2	88.5
Adapter Drop	92.7	85.5	89.3	88.2	89.1	87.0	86.2	87.3	88.4	90.3	86.8	88.1	87.8
Adapter Fusion	93.2	87.4	91.0	89.6	91.3	89.9	87.1	88.7	89.2	91.4	87.7	88.7	89.3
MADX - en	93.6	88.2	90.4	89.0	91.4	90.3	88.4	89.5	88.9	91.8	89.1	89.0	89.6
MADX - hi	93.0	86.2	88.1	88.2	89.0	87.8	87.9	89.2	90.1	91.0	**88.6**	88.1	88.6
FT	93.8	**89.3**	91.7	**91.8**	**93.2**	**91.7**	**89.1**	**91.4**	90.3	**92.4**	88.2	**91.1**	**90.9**
MTL	93.5	87.2	90.2	90.5	92.9	89.4	87.8	90.6	**90.6**	91.3	86.0	88.9	90.2

Table 13: Results on IndicXNLI task with IndicBERT. Metric: Accuracy. Column "Avg.XL" reports average cross-lingual zero-shot performance.

Method	en	as	bn	gu	hi	kn	ml	mr	or	pa	ta	te	Avg.XL
Houlsby	82.4	69.3	74.0	73.3	75.2	74.1	72.9	70.6	71.5	74.6	73.4	72.8	**72.9**
Bapna	81.9	68.7	73.3	71.1	73.4	73.3	73.0	69.3	71.5	73.9	72.9	72.8	72.1
Houlsby Parallel	82.5	69.2	73.0	72.7	73.8	73.6	72.6	70.2	71.7	74.4	72.6	72.7	72.4
Bapna Parallel	82.7	69.7	73.7	71.9	73.2	73.5	73.0	69.7	**72.2**	74.2	**73.5**	73.0	72.5
Prefix Tuning	82.6	**70.9**	**74.3**	**73.7**	**75.6**	74.0	**73.5**	72.1	72.6	**75.2**	73.1	**73.4**	73.5
Lora	80.3	68.1	71.7	69.8	72.5	72.2	70.5	68.5	69.5	72.7	70.7	71.3	70.7
Compacter	74.8	68.1	71.0	70.5	72.1	70.2	69.1	66.7	69.6	71.8	69.9	69.7	69.9
Adapter Drop	80.6	69.7	71.8	71.7	74.4	73.1	72.1	70.0	71.1	73.7	72.5	72.4	72.0
Adapter Fusion	79.9	68.0	71.6	70.2	72.8	71.9	71.8	68.2	70.0	73.0	70.9	69.9	70.8
MADX -en	82.1	69.9	73.3	72.9	73.8	73.0	72.4	69.5	70.7	74.2	73.2	73.0	72.4
MADX - hi	79.3	68.3	72.1	70.1	72.5	71.5	70.5	68.7	70.3	73.2	70.9	71.1	70.8
FT	**83.0**	69.4	73.1	73.5	75.1	**74.4**	72.6	**71.0**	71.4	75.0	72.8	73.0	**72.9**
MTL	80.9	67.4	71.9	71.5	72.9	71.7	69.9	68.9	69.8	72.9	70.0	70.6	70.7

Table 14: Results on IndicCOPA with IndicBERT. Metric: Accuracy. Column "Avg.XL" reports average cross-lingual zero-shot performance.

Method	en	as	bn	gu	hi	kn	ml	mr	or	pa	ta	te	Avg.XL
Houlsby	61.5	63.0	66.4	64.7	66.4	63.8	62.2	63.3	59.2	63.2	66.2	66.6	64.1
Bapna	59.9	61.4	66.4	60.5	58.6	60.2	59.2	62.1	59.6	59.6	61.4	60.4	60.9
Houlsby Parallel	61.4	61.2	65.6	63.2	61.7	62.2	62.6	65.5	60.8	63.6	68.0	**66.8**	63.7
Bapna Parallel	60.5	60.4	63.0	61.6	59.0	60.4	61.4	64.4	59.8	60.2	64.2	61.4	61.4
Prefix Tuning	61.1	62.2	65.6	**67.4**	**66.8**	66.6	61.8	61.9	65.2	64.2	**69.6**	67.2	65.3
Lora	57.4	60.0	64.2	58.7	62.1	64.6	60.0	61.5	58.0	58.4	59.2	60.8	60.7
Compacter	50.8	59.8	66.6	62.9	63.5	64.0	63.0	63.3	58.2	62.4	66.0	66.0	63.2
Adapter Drop	52.3	59.6	64.0	61.2	60.4	64.6	62.4	61.7	57.2	61.6	64.0	63.0	61.8
Adapter Fusion	59.9	57.6	65.2	58.5	58.4	58.8	57.8	61.0	58.8	59.4	58.6	58.4	59.3
MADX -en	56.9	60.8	65.8	61.8	63.3	62.8	57.8	64.8	60.0	62.8	63.6	64.8	62.6
MADX - hi	58.4	62.2	**67.2**	62.3	63.5	63.0	59.4	63.0	60.2	64.0	66.0	63.4	63.1
FT	**62.3**	61.2	65.2	60.5	59.5	61.8	62.0	60.8	61.0	63.4	68.0	63.8	62.5
MTL	61.4	**64.6**	66.6	62.5	64.4	**66.6**	**64.4**	**67.3**	**65.8**	**64.6**	65.4	66.6	**65.3**

Table 15: Results on IndicXParaphrase with IndicBERT. Metric: Accuracy.Column "Avg.XL" reports average cross-lingual zero-shot performance.

Method	en	as	bn	gu	hi	kn	ml	mr	or	pa	te	Avg.XL
Houlsby	92.3	57.8	50.8	75.7	51.2	59.7	57.4	54.6	57.6	53.8	55.5	57.4
Bapna	91.4	56.5	49.6	72.6	49.9	57.2	56.0	53.0	55.8	54.0	54.6	55.9
Houlsby Parallel	90.6	56.6	49.8	71.2	50.3	57.2	55.8	53.1	55.9	53.9	54.2	55.8
Bapna Parallel	91.3	56.7	50.0	72.8	50.7	57.6	56.4	53.0	56.6	53.8	55.1	56.3
Prefix Tuning	92.2	55.3	49.1	73.8	49.7	55.5	54.8	53.6	55.2	57.1	53.7	55.8
Lora	90.2	54.8	50.0	70.0	50.0	55.8	54.6	51.8	53.8	54.9	53.9	55.0
Compacter	72.7	49.6	47.0	63.9	48.3	45.1	46.3	48.9	47.1	59.8	52.5	50.8
Adapter Drop	75.0	50.8	49.5	68.1	50.2	47.7	49.6	50.2	49.6	58.6	54.6	52.9
Adapter Fusion	92.2	57.1	49.8	73.5	50.4	57.0	56.6	52.9	56.6	54.2	55.0	56.3
MADX -en	91.0	56.5	49.9	72.5	50.4	56.5	55.2	53.2	55.2	54.9	54.9	55.9
MADX - hi	90.6	57.1	49.6	73.4	50.3	58.4	55.7	53.5	56.7	54.9	54.9	56.5
FT	**93.0**	56.8	50.9	76.5	51.1	57.8	56.5	55.0	56.7	56.2	55.0	57.3
MTL	91.5	**70.7**	**88.3**	**81.3**	**81.7**	**74.7**	**73.6**	**75.9**	**66.2**	**58.7**	**71.6**	**74.3**

Method	en	as	bn	gu	hi	kn	ml	mr	or	pa	ta	te	Avg.XL
Houlsby	91.5	41.7	69.2	77.5	**78.3**	71.8	77.6	76.5	16.4	63.9	68.8	79.1	65.5
Bapna	91.0	37.5	70.0	78.3	76.2	70.9	78.3	77.9	16.1	65.1	67.9	79.3	65.2
Houlsby Parallel	92.2	**46.2**	72.0	77.2	75.9	74.1	**79.7**	77.9	17.3	63.1	69.1	**81.2**	**66.7**
Bapna Parallel	91.1	34.6	70.5	76.9	75.7	72.3	78.2	74.8	16.9	62.6	69.1	79.6	64.7
Prefix Tuning	91.5	42.6	**72.1**	77.7	76.2	75.0	**79.7**	78.1	**17.3**	68.6	69.1	**81.2**	67.1
Lora	90.4	40.7	70.7	75.0	72.7	71.0	75.6	74.1	17.1	60.0	61.4	78.3	63.3
Compacter	89.2	38.5	65.8	73.9	72.6	67.0	72.5	71.1	16.6	59.4	64.1	73.1	61.3
Adapter Drop	90.4	30.2	71.2	76.3	75.4	71.4	77.4	78.3	16.8	66.7	65.4	79.1	64.4
Adapter Fusion	92.0	42.6	71.8	**79.2**	75.1	76.2	79.4	78.5	16.5	66.3	69.8	80.7	66.9
MADX -en	91.5	43.6	69.1	78.9	75.3	73.9	78.8	76.6	16.2	63.8	68.7	81.1	66.0
MADX - hi	91.1	34.6	69.9	76.6	75.6	70.8	76.8	74.2	17.0	63.5	66.7	79.3	64.1
FT	**92.8**	38.7	71.6	77.4	77.8	**75.3**	79.3	**78.7**	17.1	65.6	**70.7**	81.6	**66.7**
MTL	91.0	34.0	69.8	78.3	76.0	74.2	77.9	78.4	16.1	**66.9**	66.7	79.7	65.3

Table 16: Results on IndicNER task with IndicBERT. Metric: F1 Score. Column "Avg.XL" reports average cross-lingual zero-shot performance.

Method	en	as	bn	gu	hi	kn	ml	mr	or	pa	ta	te	Avg.XL
Houlsby	81.7	**44.7**	**52.9**	**45.2**	54.9	46.7	**46.2**	**48.9**	**51.8**	52.4	**44.9**	**60.9**	**50.0**
Bapna	80.9	44.1	51.4	44.0	55.6	46.4	42.5	45.9	49.8	52.4	43.0	60.0	48.6
Houlsby Parallel	82.0	43.9	52.6	44.4	55.2	**47.5**	43.9	46.2	50.7	**53.2**	43.6	60.3	49.2
Bapna Parallel	81.4	44.2	52.0	43.6	55.6	47.2	43.6	45.4	50.8	52.8	43.4	59.7	48.9
Prefix Tuning	81.0	43.0	50.9	43.9	52.7	46.8	43.2	46.5	51.1	50.8	43.5	59.9	48.4
Lora	79.5	41.9	50.6	43.9	52.9	44.3	43.0	44.3	48.8	51.4	43.1	57.8	47.4
Compacter	73.0	40.8	48.5	42.3	50.9	43.9	41.6	45.1	46.8	49.6	42.0	59.2	46.4
Adapter Drop	70.7	38.3	46.8	40.9	50.1	42.3	40.3	43.0	46.3	47.4	38.1	55.2	44.4
Adapter Fusion	81.9	44.4	51.9	43.9	**55.8**	46.0	42.8	45.5	50.1	52.1	43.6	59.5	48.7
MADX-en	81.1	41.4	50.6	43.3	53.8	45.3	42.4	44.8	49.8	52.1	42.5	58.1	47.6
MADX-hi	79.4	41.1	50.2	43.7	54.9	44.9	41.9	44.3	49.0	51.4	41.5	58.6	47.4
FT	**82.1**	44.4	52.8	44.9	54.6	46.9	44.6	46.5	51.3	52.0	43.9	60.3	49.3
MTL	**82.1**	39.8	49.1	42.6	48.9	42.9	42.2	43.6	48.1	47.3	39.7	56.2	45.5

Table 17: Results on IndicQA task with IndicBERT. Metric: F1 score. Column "Avg.XL" reports average cross-lingual zero-shot performance.

Method	en	as	bn	gu	hi	kn	ml	mr	or	pa	ta	te
Houlsby	83.9	60.7	67.4	**71.0**	**69.7**	67.8	**67.4**	**67.2**	57.7	66.7	68.3	70.7
Pfeiffer	83.1	59.2	66.8	69.3	67.6	66.2	66.0	66.1	56.9	65.9	66.7	69.3
Houlsby Parallel	83.6	**60.8**	67.4	69.9	68.3	67.6	67.2	**67.2**	57.8	66.6	68.3	**70.9**
Pfeiffer Parallel	83.4	58.8	66.7	69.5	67.8	66.8	66.8	66.1	57.8	65.8	67.7	69.8
Prefix Tuning	83.7	59.9	66.8	70.8	68.8	67.8	66.5	66.9	**58.2**	**67.7**	68.2	70.7
Lora	81.8	58.2	65.6	67.1	65.9	65.6	64.3	64.5	55.7	63.9	64.1	68.4
Compacter	75.5	57.3	64.8	67.1	66.1	63.2	63.4	63.5	54.4	65.4	66.0	68.3
Adapter Drop	77.0	55.7	65.4	67.7	66.6	64.3	64.7	65.1	54.9	66.4	65.3	68.8
Adapter Fusion	83.2	59.5	66.9	69.1	67.3	66.6	65.9	65.8	56.9	66.1	66.1	68.7
MADX - en	82.7	60.1	66.5	69.7	68.0	67.0	65.8	66.4	56.8	66.6	67.4	70.1
MADX - hi	82.0	58.3	66.2	69.0	67.7	66.1	65.4	65.5	57.2	66.3	66.7	69.2
FT	**84.5**	60.0	**67.6**	70.8	68.6	**68.0**	**67.4**	**67.2**	58.0	67.4	**68.7**	70.8

Table 18: This table compares the performance of various adapters and FT with results averaged across all tasks.

Method	FLOP	% ↑ FLOS	Epoch	% ↑ Epoch
Houlsby	1.8E+17	249.8	17	240
Bapna	1.5E+17	185.2	14	180
Houlsby Parallel	1.1E+17	105.3	10	100
Bapna Parallel	8.6E+16	62.9	8	60
Prefix Tuning	1.5E+17	190.9	13	160
Lora	1.7E+17	223.2	16	220
Compacter	2.4E+17	363.9	23	360
Adapter Drop	1.2E+17	124.3	11	120
FT	5.3E+16	0.0	5	0
Avg Adapter	1.5E+17	188.2	14	180

Table 19: This table report the total computation cost on Sentiment task for FT and various adapters using IndicBERT. Here % ↑ FLOS refers to the percent increase of FLOs relative to FLOs of FT, similarly % ↑ Epoch reports percent increase of epoch relative to epoch of FT

Method	FLOP	% ↑ FLOS	Epoch	% ↑ Epoch
Houlsby	4.0E+17	208.5	15	200
Bapna	4.5E+17	246.5	17	240
Houlsby Parallel	4.0E+17	208.5	15	200
Bapna Parallel	3.9E+17	205.4	15	200
Prefix Tuning	4.4E+17	237.2	15	200
Lora	3.9E+17	203.1	15	200
Compacter	2.9E+17	121.7	11	120
Adapter Drop	4.2E+17	225.6	16	220
FT	1.3E+17	0.0	5	0
Average Adapter	4.0E+17	207.1	14.88	197.5

Table 20: This table report the total computation cost on XNLI task for FT and various adapters using IndicBERT.. here % ↑ FLOS refers to the percent increase of FLOs relative to FLOs of FT, similarly % ↑ Epoch reports percent increase of epoch relative to epoch of FT

Method	FLOP	% ↑ FLOS	Epoch	% ↑ Epoch
Houlsby	3.8E+17	376.6	28	366.7
Bapna	3.3E+17	321.0	25	316.7
Houlsby Parallel	3.0E+17	274.2	22	266.7
Bapna Parallel	2.3E+17	185.7	17	183.3
Prefix Tuning	2.2E+17	179.4	15	150.0
Lora	2.1E+17	168.0	16	166.7
Compacter	5.9E+17	650.9	45	650.0
Adapter Drop	1.9E+17	136.4	14	133.3
FT	7.9E+16	0.0	6	0.0
Average Adapter	3.1E+17	286.5	22.75	279.2

Table 21: This table report the total computation cost on COPA task for FT and various adapters using IndicBERT.. here % ↑ FLOS refers to the percent increase of FLOs relative to FLOs of FT, similarly % ↑ Epoch reports percent increase of epoch relative to epoch of FT

Method	FLOP	% ↑ FLOS	Epoch	% ↑ Epoch
Houlsby	1.5E+17	88.5	22	83.3
Bapna	1.1E+17	43.6	17	41.7
Houlsby Parallel	7.4E+16	-5.8	11	-8.3
Bapna Parallel	1.2E+17	52.6	18	50.0
Prefix Tuning	1.5E+17	96.2	21	75.0
Lora	1.5E+17	93.6	23	91.7
Compacter	9.8E+16	25.9	15	25.0
Adapter Drop	4.6E+16	-40.6	7	-41.7
FT	7.8E+16	0.0	12	0.0
Total Adapter	1.1E+17	44.2	16.75	39.6

Table 22: This table report the total computation cost on Paraphrase task for FT and various adapters using IndicBERT. here % ↑ FLOS refers to the percent increase of FLOs relative to FLOs of FT, similarly % ↑ Epoch reports percent increase of epoch relative to epoch of FT

Method	FLOP	% ↑ FLOS	Epoch	% ↑ Epoch
Houlsby	4.2E+15	19.8	14	16.7
Bapna	6.2E+15	77.7	21	75.0
Houlsby Parallel	6.6E+15	88.0	22	83.3
Bapna Parallel	4.4E+15	26.9	15	25.0
Prefix Tuning	6.2E+15	77.4	19	58.3
Lora	8.5E+15	143.6	29	141.7
Compacter	1.2E+16	252.4	42	250.0
Adapter Drop	4.2E+15	19.8	14	16.7
FT	3.5E+15	0.0	12	0.0
Average Adapter	6.6E+15	88.2	22	83.3

Table 23: This table report the total computation cost on NER task for FT and various adapters using IndicBERT. here % ↑ FLOS refers to the percent increase of FLOs relative to FLOs of FT, similarly % ↑ Epoch reports percent increase of epoch relative to epoch of FT

Method	FLOP	% ↑ FLOS	Epoch	% ↑ Epoch
Houlsby	7.3E+17	599.0	41	583.3
Bapna	5.8E+17	456.7	33	450.0
Houlsby Parallel	3.9E+17	275.0	22	266.7
Bapna Parallel	5.1E+17	389.4	29	383.3
Prefix Tuning	3.1E+17	198.1	16	166.7
Lora	5.2E+17	402.9	30	400.0
Compacter	8.7E+17	735.6	50	733.3
Adapter Drop	1.1E+17	1.9	6	0.0
FT	1.0E+17	-	6	-
Avg Adapter	5.0E+17	382.3	28.4	372.9

Table 24: This table report the total computation cost on QA task for FT and various adapters using IndicBERT. here % ↑ FLOS refers to the percent increase of FLOs relative to FLOs of FT, similarly % ↑ Epoch reports percent increase of epoch relative to epoch of FT

Method	XLMR-Base	XLMR-Large
Houlsby	484.3	200.5
Bapna	547.1	139.9
Houlsby Parallel	197.0	143.3
Bapna Parallel	409.1	168.6
Prefixtuning	256.5	287.3
Lora	734.7	270.0
compacter	805.3	490.1
Adapter drop	345.1	214.1

Table 25: This table reports the **percentage** increase in total FLOs with respect to FT for both XLMR-Base and XLMR-Large model

	XLMR-Base				XLMR-Large			
Method	WikiANN	XNLI	XQuAD	Total	WikiANN	XNLI	XQuAD	Total
Houlsby	506.4	483.1	484.0	484.3	316.4	75.6	299.6	200.5
Bapna	439.4	464.6	582.0	547.1	172.3	73.7	194.3	139.9
Houlsby Parallel	270.3	483.1	86.0	197.0	103.8	126.8	160.0	143.3
Pfeiffer Parallel	643.4	409.2	401.0	409.1	71.1	167.5	175.9	168.6
Prefixtuning	157.4	237.7	267.0	256.5	200.6	226.3	344.9	287.3
Lora	474.3	404.0	869.0	734.7	332.1	57.9	446.9	270.0
Compacter	905.8	818.2	797.0	805.3	528.9	336.8	618.4	490.1
Adapter drop	281.9	409.2	323.0	345.1	617.0	153.1	240.0	214.1

Table 26: This table reports the **percentage** increase in computational cost with respect to FT for XLM-R model for task NER, XNLI and QA. "Total" reports the percentage increase of total FLOs for the method relative to total FT FLOs

EN	XLMR-Base				XLMR-Large			
Method	NER	XNLI	QA	Average	NER	XNLI	QA	Average
Houlsby	81.0	82.7	**84.1**	82.6	**83.5**	85.9	88.2	85.9
Bapna	79.9	81.3	83.2	81.5	83.1	86.4	87.4	85.7
Houlsby parallel	80.5	83.5	83.7	82.6	83.0	87.9	88.0	86.3
Bapna parallel	80.8	80.9	82.8	81.5	82.5	88.0	87.7	86.1
Prefixtuning	79.0	79.5	81.7	80.1	83.2	**88.2**	**88.3**	**86.5**
Lora	78.6	79.7	81.7	80.0	81.7	85.6	86.9	84.7
compacter	72.3	76.4	76.6	75.1	76.1	85.6	85.0	82.2
Adapter drop	81.1	80.3	82.7	81.4	82.6	88.0	88.0	86.2
FT	**82.3**	**83.1**	83.3	**82.9**	82.8	87.3	88.0	86.0

Table 27: Overall performance on **English** for XLMR-B and XLMR-L model

XL	XLMR-Base				XLMR-Large			
Method	**NER**	**XNLI**	**QA**	**Average**	**NER**	**XNLI**	**QA**	**Average**
Houlsby	61.0	72.6	**71.5**	**68.4**	64.6	76.2	**78.6**	73.1
Bapna	58.3	71.3	69.9	66.5	64.3	76.7	78.0	73.0
Houlsby Parallel	59.2	72.8	70.1	67.4	**65.3**	78.7	77.8	73.9
Bapna Parallel	57.1	70.3	69.7	65.7	63.1	**78.8**	77.6	73.2
Prefixtuning	58.5	69.9	67.7	65.4	64.7	78.7	77.6	73.7
Lora	58.6	70.5	68.4	65.8	62.3	76.9	77.1	72.1
Compacter	55.1	66.8	64.1	62.0	58.5	76.4	75.3	70.1
Adapter drop	60.5	70.2	71.3	67.3	64.6	**78.8**	78.5	**74.0**
FT	**61.7**	**73.7**	70.8	**68.7**	63.9	77.0	78.0	73.0

Table 28: Overall **cross-lingual** performance for XLMR-B and XLMR-L model

Method	en	ar	bg	de	el	es	fr	hi	ru	sw	th	tr	ur	vi	zh	Avg.XL
houlsby	83.5	45.6	81.6	79.3	79.9	76.2	78.7	71.1	68.0	68.2	0.6	82.0	69.1	77.5	26.2	64.6
Bapna	83.1	41.0	81.4	78.1	77.2	77.0	78.7	73.2	71.5	68.3	2.0	79.3	69.1	76.8	26.2	64.3
houlsby parallel	83.0	46.4	83.1	79.2	79.2	76.1	79.0	70.0	71.4	70.4	0.6	82.0	75.6	76.6	24.7	65.3
Bapna parallel	82.5	48.3	79.3	77.9	77.9	72.7	78.3	66.5	71.5	68.8	1.4	80.0	63.3	75.0	22.2	63.1
prefixtuning	83.2	48.5	79.2	77.8	79.1	76.8	79.8	73.5	69.1	66.5	4.3	79.9	71.1	75.6	24.6	64.7
lora	81.7	46.0	80.0	77.9	76.9	68.7	77.8	66.9	67.9	66.6	2.5	76.4	65.7	76.8	21.5	62.3
compacter	76.1	38.8	75.5	75.5	74.8	73.8	74.8	62.7	58.7	60.2	1.2	75.9	65.5	69.1	12.2	58.5
Adapter drop	82.6	48.0	82.0	78.5	78.7	75.0	79.8	68.0	69.4	68.1	1.0	80.0	75.0	76.4	24.0	64.6
FT	82.8	49.3	81.6	79.1	76.6	77.7	81.1	70.6	70.9	66.9	0.4	78.3	60.7	77.7	23.1	63.9

Table 29: Results on WikiANN task with **XLM-R Large** model, metric: F1 score

Method	en	ar	bg	de	el	es	fr	hi	ru	sw	th	tr	ur	vi	zh	Avg.XL
houlsby	81.0	44.4	76.0	73.6	74.3	71.6	76.1	70.1	61.7	69.3	1.5	75.8	65.5	68.9	25.2	61.0
Bapna	79.9	41.7	73.2	72.3	73.0	73.7	74.9	62.9	59.2	67.6	2.0	72.6	56.2	62.7	23.6	58.3
houlsby parallel	80.5	44.7	75.8	73.3	73.8	67.7	74.7	66.4	62.1	66.7	1.8	74.0	57.5	64.8	26.0	59.2
Bapna parallel	80.8	42.1	74.3	72.4	70.6	70.3	74.3	62.0	61.1	61.7	1.0	71.3	51.3	62.6	24.4	57.1
prefixtuning	79.0	46.5	75.9	70.0	70.6	72.8	74.8	62.4	59.7	62.3	1.1	70.8	64.2	67.4	20.2	58.5
lora	78.6	42.8	74.6	71.2	70.7	71.7	74.1	62.4	57.8	67.0	2.9	70.6	63.0	68.0	23.7	58.6
compacter	72.3	42.0	72.9	70.2	68.3	61.6	67.6	59.9	54.4	62.8	0.7	69.0	56.8	63.5	21.6	55.1
Adapter drop	81.1	45.7	76.8	73.8	74.5	69.2	74.7	66.1	62.4	66.0	1.9	74.8	65.7	67.4	27.1	60.5
FT	82.3	48.5	77.0	73.3	74.7	75.3	75.7	67.7	63.0	69.2	3.8	76.6	64.7	69.8	24.1	61.7

Table 30: Results on WikiANN task with **XLM-R Base** model, metric: F1 score

Method	en	ar	bg	de	el	es	fr	hi	ru	sw	th	tr	ur	vi	zh	Avg.XL
houlsby	85.9	75.0	80.0	80.4	78.8	81.2	80.0	73.4	76.8	69.7	74.1	76.2	68.3	76.8	75.4	76.2
Bapna	86.4	75.5	80.3	81.0	79.2	81.2	80.6	73.8	77.8	69.7	74.9	76.3	70.4	77.1	76.3	76.7
houlsby parallel	87.9	77.7	82.3	82.6	80.9	83.7	82.3	77.1	79.6	71.2	77.0	78.3	72.1	78.5	78.8	78.7
Bapna parallel	88.0	78.4	82.9	82.7	81.2	84.0	82.7	76.1	79.5	71.3	76.4	78.4	72.2	78.7	78.1	78.8
prefixtuning	88.2	78.4	82.3	81.6	81.6	83.3	82.7	76.0	79.6	71.1	77.3	78.5	72.6	78.9	78.3	78.7
lora	85.6	75.3	80.8	80.5	79.7	81.7	80.9	74.5	78.5	70.0	75.1	76.9	70.1	77.0	76.3	76.9
compacter	85.6	74.2	80.2	80.5	78.9	80.7	80.5	74.7	77.3	69.7	74.9	75.9	69.7	76.5	76.4	76.4
Adapter drop	88.0	77.7	82.8	82.5	82.1	84.0	82.6	76.1	80.1	72.2	76.7	78.8	71.3	79.0	77.6	78.8
FT	87.3	76.1	81.9	80.5	79.5	82.3	81.7	73.9	79.5	65.5	75.7	76.0	68.7	78.4	78.4	77.0

Table 31: Results on XNLI task with **XLM-R Large** model, metric: Accuracy

Method	en	ar	bg	de	el	es	fr	hi	ru	sw	th	tr	ur	vi	zh	Avg.XL
houlsby	82.7	70.9	77.1	76.3	74.7	77.9	77.9	68.6	74.3	64.5	71.1	71.8	65.5	73.4	72.5	72.6
Bapna	81.3	68.8	75.3	74.2	73.4	76.7	75.8	67.6	73.5	64.0	69.7	71.2	64.0	72.8	71.2	71.3
houlsby parallel	83.5	70.3	77.0	75.9	74.8	78.0	77.7	69.6	74.7	65.1	71.3	71.7	65.4	74.8	73.1	72.8
Bapna parallel	80.9	68.0	74.7	73.2	72.1	76.2	75.0	67.2	72.4	63.2	68.1	70.1	62.3	71.5	69.8	70.3
prefixtuning	79.5	68.7	73.7	72.6	71.7	74.1	74.2	66.3	71.3	62.9	69.5	69.1	62.9	72.0	70.1	69.9
lora	79.7	68.4	75.0	73.6	72.2	75.3	74.5	66.6	72.2	63.6	68.1	70.9	63.6	71.4	71.3	70.5
compacter	76.4	64.1	70.7	70.6	69.4	72.8	71.8	61.7	70.0	60.4	63.4	67.4	59.0	68.1	66.5	66.8
Adapter drop	80.3	68.1	74.4	73.6	71.1	75.9	75.3	67.2	72.1	63.3	68.9	69.9	62.1	71.7	70.0	70.2
FT	83.1	71.3	78.0	76.6	75.3	78.6	76.9	71.3	75.4	64.0	73.0	73.0	67.5	75.6	74.7	73.7

Table 32: Results on XNLI task with **XLM-R Base** model, metric: Accuracy

Method	en	ar	de	el	es	hi	ro	ru	th	tr	vi	zh	Avg.XL
houlsby	88.2	77.3	81.2	80.3	83.3	77.0	85.0	80.8	74.3	75.2	80.0	70.1	78.6
Bapna	87.4	75.7	79.9	80.5	82.6	75.6	84.1	80.7	75.6	73.9	79.8	69.7	78.0
houlsby parallel	88.0	75.3	81.4	80.4	81.9	76.2	84.2	79.9	74.2	74.2	79.2	68.7	77.8
Bapna parallel	87.7	75.2	80.4	80.4	82.0	75.6	84.1	79.9	73.7	73.7	79.4	69.4	77.6
prefixtuning	88.3	75.4	81.5	80.5	82.3	75.6	83.0	79.3	74.5	73.9	78.6	68.8	77.6
lora	86.9	75.8	80.6	78.5	81.2	75.0	82.5	79.1	75.2	72.7	77.9	69.4	77.1
compacter	85.0	73.7	77.7	77.6	79.5	74.7	80.9	78.3	70.8	70.5	76.9	68.2	75.3
Adapter drop	88.0	76.1	81.3	81.1	83.2	76.7	85.1	80.7	74.3	74.6	80.3	69.6	78.5
FT	88.0	76.3	80.7	80.3	81.8	76.2	84.2	79.6	75.0	74.4	79.8	69.7	78.0

Table 33: Results on Squad, XQAUD task with **XLM-R Large** model, metric: F1 score

Method	en	ar	de	el	es	hi	ro	ru	th	tr	vi	zh	Avg.XL
houlsby	84.1	67.0	75.0	73.3	76.8	69.8	79.0	72.6	68.3	66.6	73.8	64.6	71.5
Bapna	83.2	64.5	73.6	71.0	75.1	66.1	77.5	72.6	65.6	66.3	72.9	63.4	69.9
houlsby parallel	83.7	65.9	74.6	72.0	75.5	66.5	77.9	72.8	64.4	66.5	71.9	62.9	70.1
Bapna parallel	82.8	64.4	73.2	72.6	74.0	65.4	77.4	73.0	65.0	65.4	72.0	63.9	69.7
prefixtuning	81.7	63.1	71.3	70.0	72.1	64.4	75.3	70.2	62.6	63.5	69.9	62.1	67.7
lora	81.7	61.4	71.8	71.2	72.9	65.4	76.7	71.9	63.1	65.4	71.3	61.0	68.4
compacter	76.6	60.7	67.0	64.9	68.8	62.4	70.3	66.8	58.6	59.5	68.9	57.3	64.1
Adapter drop	82.7	66.6	74.3	73.6	75.3	70.2	77.0	74.6	68.2	66.8	74.5	62.8	71.3
FT	83.3	66.5	74.6	72.2	75.1	66.8	77.5	73.4	66.8	67.5	73.2	65.4	70.8

Table 34: Results on SQUAD, XQUAD task with **XLM-R Base** model, metric: F1 score