Low-Rank Updates of pre-trained Weights for Multi-Task Learning

Alexandre Audibert[†], Massih-Reza Amini[†], Konstantin Usevich[‡], and Marianne Clausel[⋆]

[†]Université Grenoble Alpes, Computer Science Laboratory, Grenoble, France
{firstname.lastname}@univ-grenoble-alpes.fr
[‡]CNRS, Université de Lorraine, CRAN, Nancy, France
konstantin.usevich@univ-lorraine.fr
[⋆]Université de Lorraine, Institut Elie Cartan de Lorraine, Nancy, France
marianne.clausel@univ-lorraine.fr

Abstract

Multi-Task Learning used with pre-trained models has been quite popular in the field of Natural Language Processing in recent years. This framework remains still challenging due to the complexity of the tasks and the challenges associated with fine-tuning large pre-trained models. In this paper, we propose a new approach for Multi-task learning which is based on stacking the weights of Neural Networks as a tensor. We show that low-rank updates in the canonical polyadic tensor decomposition of this tensor of weights lead to a simple, yet efficient algorithm, which without loss of performance allows to reduce considerably the model parameters. We investigate the interactions between tasks inside the model as well as the inclusion of sparsity to find the best tensor rank and to increase the compression rate. Our strategy is consistent with recent efforts that attempt to use constraints to fine-tune some model components. More precisely, we achieve equivalent performance as the state-of-the-art on the General Language Understanding Evaluation benchmark by training only 0.3% of the parameters per task while not modifying the baseline weights.

1 Introduction

Multi-task learning (MTL) aims in exploiting simultaneously similarities and differences between related tasks (Caruana, 1997). Compared to training the models separately, this can lead to enhance learning efficiency and prediction accuracy for the task-specific models.

In addition to certain similarities to transfer learning and data augmentation, MTL has a regularizing effect in practice (Caruana, 1997). MTL also has the advantage of storage efficiency, which is advantageous for devices with less memory. On the other hand, MTL performance may be impacted by task covariance (Wu et al., 2020), various loss functions, and difference between dataset sizes (Pilault et al., 2021). Additionally, there are still some MTL-related limitations, including negative transfer, in which learning two tasks at once lowers the model's performance on both tasks (Crawshaw, 2020; Wu et al., 2020), and catastrophic forgetting in which one some tasks features can be overlooked during the training process (Serra et al., 2018).

In this study, we aim to decrease the amount of language model trainable parameters in the MTL framework. To achieve this, we suggest stacking weight matrices corresponding to several tasks in a 3-way tensor and performing a tensor low-rank update, which is similar to the LoRA technique in the single-task case (Hu et al., 2021). One of the main advantages of the tensor approach is that it allows for splitting the weight updates into shared and task-specific parts. Moreover we extend our approach to the bias term which showed remarkable results in Ben Zaken et al. (2022). We test our method using the General Language Understanding Evaluation (GLUE) Benchmark. Thus we demonstrate that low-rank update for both matrix and bias successfully strikes a balance between preserving positive transfer and minimizing negative transfer by only training 0.3% of the initial parameters per task. We also look into how different model factors affect the way tasks interact with one another.

2 Related Work

Multi-Task Learning for NLP. Training the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) in hard-sharing Multi-Task Learning may be subject to negative transfer (Liu et al., 2019; Glover and Hokamp, 2019). To tackle this problem, some studies propose to use a shared hyper-network or to do conditional learning (Pilault et al., 2021; Mahabadi et al., 2021; He et al., 2022). This method consists in creating a task embedding which is used to build model's layers. Another approach to circumvent the negative transfer problem, consists to use Knowledge Distillation (KD) where single-task

Findings of the Association for Computational Linguistics: ACL 2023, pages 7544–7554
July 9-14, 2023

teachers transfer their knowledge to one multi-task student (Clark et al., 2019; Wei et al., 2021).

Tensor methods. The use of tensor methods mainly focused on applying tensor approximations for compression of pretrained models (compression of fully-connected (Oseledets, 2011) and convolutional networks Lebedev et al. (2014); Kim et al. (2015)). Ren et al. (2022) utilized tensor decomposition for compressing Pre-trained Language Models and presented a formal framework with defined nomenclature to thoroughly explore tensor decomposition approaches to compress Transformer-based language models. In multi-task learning, tensor methods have been used to introduce sharing between weights across different tasks (Romera-Paredes et al., 2013; Wimalawarne et al., 2014; Yang and Hospedales, 2017). Recent work considered splitting in task-agnostic (shared) and task-specific parts. However, the cited works were mostly focused on learning compressed representation or tensor completion, mostly with so-called Tucker tensor decomposition. The weight representation in our work is much simpler and more efficient in terms of parameters: the same frozen weight matrix is shared and task-specific updates use canonical polyadic decomposition (CPD). It is compact and have an additional interpretation with shared and task-specific factors. We can even do it with tucker (see more details on CPD in Appendix A.1.

3 MORRIS: Multi-task learning based on lOw-Rank updates of pRe-trained weIghtS

In the following, we designate vectors, matrices, and tensors, respectively, with bold lowercase letters, bold capital letters, and calligraphy letters.

We assume that there are T tasks and T associated datasets $D_i = \{(x_j^{(i)}, y_j^{(i)}) \mid j \in \{1, \ldots, N_i\}\}$ where N_i is the size of the i^{th} collection. We denote by l_i the loss function, ϕ_i the specific parameters of the i^{th} task, and Θ the shared parameters between tasks. The Multi-Task Learning objective function is:

$$\min_{\Theta, \{\phi_i\}_{i=1}^T} L(\Theta, \{\phi_i\}_{i=1}^T, \{D_i\}_{i=1}^T) = \sum_{i=1}^{T} \sum_{(x_j^{(i)}, y_j^{(i)}) \in D_i} l_i(f(\Theta, \phi_i, x_j^{(i)}), y_j^{(i)}) \quad (1)$$

By adopting a low-rank tensor update for the weights tensor and a low-rank matrix update for the biases, we suggest extending the approaches proposed by Hu et al. (2021) and Ben Zaken et al. (2022) to multi-task learning and updating the weights and biases for several tasks concurrently.

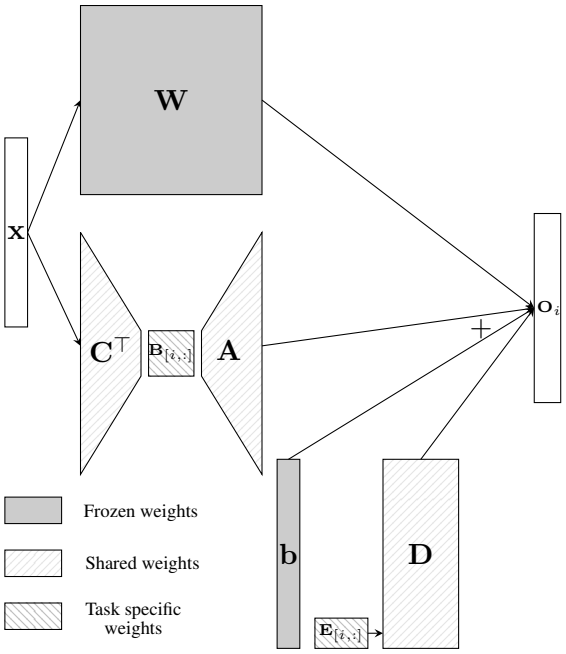

Figure 1: Tensor representation for multi-task learning in `Morris`.

3.1 The proposed framework

Our proposal is to update the output of the transformer layer for the i^{th} task as follows: if the dense layer is the query or value matrix; $\mathbf{O}_i = (\mathbf{W} + \mathbf{Q}_i)\mathbf{X} + \mathbf{b} + \mathbf{b}_i$, otherwise we have $\mathbf{O}_i = \mathbf{W}\mathbf{X} + \mathbf{b} + \mathbf{b}_i$; where $\mathbf{W}$ and $\mathbf{b}$ are frozen weights of BERT, $\mathbf{Q}_i$ and $\mathbf{b}_i$ are the updates and $\mathbf{X}$, $\mathbf{O}_i$ are respectively the input, output of the layer (Figure 1).

Weights assumption. The weight updates for T tasks can be stacked in a single $d \times T \times d$ tensor, so each matrix is a slice is $\mathcal{Q}_{[:,i,:]} = \mathbf{Q}_i$ of the tensor. Then our assumption is that $\mathcal{Q}$ has low-rank; $\mathcal{Q} = \sum_{r=1}^{r_w} \mathbf{A}_{[:,r]} \otimes \mathbf{B}_{[:,r]} \otimes \mathbf{C}_{[:,r]}$, where $\mathbf{A}, \mathbf{C} \in \mathbb{R}^{d \times R_w}$, $\mathbf{B} \in \mathbb{R}^{T \times R_w}$ and r_w represents the rank.

Bias assumption. The variation of the original bias for each task can be represented by a matrix $\hat{\mathbf{B}} \in \mathbb{R}^{d \times T}$ where the i^{th} column $\mathbf{b}_i$ represents the bias of the task $i \in \{1, \ldots, T\}$. We assume that the matrix $\hat{\mathbf{B}}$ is a low-rank matrix and can be written by the product of two matrices $\mathbf{D} \in \mathbf{R}^{d \times r_b}$ and

$\mathbf{E} \in \mathbf{R}^{r_b \times T}$ where r_b represented the rank, i.e.,

$$\mathbf{b}_i = \sum_{t=1}^{r_b} \mathbf{D}_{[:,t]} \times \mathbf{E}_{[t,i]} \tag{2}$$

3.2 Motivation

The straightforward MTL extension of LoRA (Hu et al., 2021) combined with BitFit (Ben Zaken et al., 2022) would be to train the same low-rank matrix $\mathbf{Q}_i$ and bias $\mathbf{b}_i$ for all tasks. This approach will be called **LoRA_Bitfit_MTL** in the rest of the paper. However we argue that our approach is more flexible than **LoRA_Bitfit_MTL** because it is a particular case of **MORRIS** where the entries of matrices **B**, **E** are all set to 1. Moreover our approach is quite natural because the concatenation of low rank matrices creates a tensor with a rank at most equal to the sum of the rank of the matrices.

3.3 Interpretation as shared and task-specific weights

The underlying assumptions allow the following interpretations. The slices of the weight tensor with low-rank tensor structure factorize as follows; $\mathbf{Q}_i = \mathbf{A} \times diag(\mathbf{B}_{[i,:]}) \times \mathbf{C}^\top$, where $diag()$ is the diagonal matrix built from a given vector. The matrix **A** and **C** are then shared between task whereas rows of **B** are task specific parameters. Similarly, for biases the matrix **D** is shared between each task and the column of **E** are task-specific (Figure 1).

3.4 Apply L_0 to find the optimal rank

The rank of the tensor must be at most the sum of the ranks of the preceding matrices. Decreasing this rank will reduce the number of model parameters, however there is no straightforward manner to fix this rank, as well as the bias rank which is similarly tough to be defined. Following, Louizos et al. (2018)'s work, we propose to use L_0 regularisation on the rows and columns of respectively **B** and **E** to define the ranks. In this case, the binary mask associated to $\boldsymbol{\alpha}$ called **z** can estimated as:

$$\begin{aligned} u &\sim U(0,1) \\ s_j &= \sigma(log(u) - log(1-u) + \alpha_j) \\ \bar{s}_j &= j \cdot (r-l) + l \\ z_j &= \min(1, \max(0, \bar{s}_j)) \end{aligned} \tag{3}$$

Based on this definition, Equation (1) can then be written in the following form:

$$\min_{\Theta, \{\phi_i\}_{i=1}^T, \alpha} L(\Theta, \{\phi_i\}_{i=1}^T, \{D_i\}_{i=1}^T) = \underset{u \sim U[0,1]}{E(u)} \sum_{i=1}^{T} \sum_{(x_j^{(i)}, y_j^{(i)}) \in D_i} l_i(f(\Theta, \phi_i \circ z, x_j^{(i)}), y_j^{(i)}) + \lambda \sum_{j=0}^{d} \sigma(\alpha_j - log(\frac{-l}{d})) \tag{4}$$

Where l and d are two stretching constants, λ controls the strength of the L_0 regularisation and σ is the the sigmoid function. More details of this regularisation can be found in Louizos et al. (2018); Guo et al. (2021).

4 Experiments and Analyse

We shall now present our experimental results.

4.1 Implementation details

We use BERT as the base model in `Morris`, that we implemented using Pytorch[1]. Furthermore, We use a fully connected layer on the [CLS] token as decoder for each task; with the cross entropy loss and the mean squared error for respectively the classification and the regression tasks. The values of the hyperparameters were fixed as the ones in LoRA (Hu et al., 2021). We select a batch size of 32 for all experiments, learning rate in $\{4e^{-4}, 1e^{-4}, 5e^{-5}\}$ for single task approaches, $\{1e^{-3}, 4e^{-4}, 1e^{-4}, 5e^{-5}\}$ for Multi-task approaches and dropout equal to 0.1 with AdamW (Loshchilov and Hutter, 2017) as the optimizer. For single task approaches the rank was set to 8, and, $\{8, 16, 32, 64\}$ for **LoRA_BitFit_MTL**. For `Morris` the rank of the bias was set to 4 in all experiments and the rank of the tensor corresponding to the weights to 64. For the L_0 regularisation, λ was found in the interval $\{1e^{-5}, 5e^{-6}, 2e^{-6}$ which corresponds to a rate of sparsity equals to {60%, 40%, 20%}. In Multi-Task learning one of the major influencing factor is the choice of the data sampling policy (Glover and Hokamp, 2019). We picked the same as Mahabadi et al. (2021) and the same number of training steps equal to 2^{18} since our objective is not to research the impacts of sampling policy. In our experiments, we did a short pre-training of 10000 step with a learning rate equals to $4e^{-4}$, after that all α_j lower than 0.5 were pushed to 0 and the

[1] https://pytorch.org

Model	Total Params	Trained params/ task	QNLI	RTE	QQP	MNLI-m	MNLI-mm	SST-2	MRPC	COLA	STS-B	Avg
Single Task												
BERT[a]	x9	100%	90.5	66.4	71.2	84.6	83.4	**93.5**	88.9	52.1	85.8	79.6
Bitfit[b]	x1.008	**0.09%**	89.7	65.5	67.8	80.8	80.9	92.4	87.4	47.2	**87.6**	77.7
Multi-task												
BERT MTL[c]	**x1**	11.1%	90.5	74.5	70.4	83.5	83.1	93.1	88.0	48.5	80.6	79.1
BERT MTL[d]	**x1**	11.1%	89.3	**76.6**	70.8	84.0	83.4	93.4	86.7	51.2	83.6	79.9
PALs[d]	x1.13	12.5%	90.0	76.0	**71.5**	84.3	83.5	92.6	88.7	51.2	85.8	80.4
CA-MTL[d]	x1.12	5.6%	90.5	76.4	69.2	**85.9**	85.8	93.2	88.6	**53.1**	85.3	**80.9**
Our approach												
Morris†	x1.024	0.27%	91.1	73.7	70.6	83.8	83.5	92.8	**90.2**	52.0	85.8	80.4
Morris L0†	x1.014	0.16%	**91.6**	73.7	70.5	84.1	83.1	92.1	89.6	49.9	86.3	80.1

Table 1: Experimental results on the test set. The best result on each column is in bold face. Results of [a] are from (Devlin et al., 2018); [b] are from (Fu et al., 2022); [c] are from (Glover and Hokamp, 2019); [d] are from (Pilault et al., 2021); † are results obtained with the best checkpoint in our settings.

others were set to pushed to 1. The pseudo-code of our approach is provided in the Appendix A.6).

4.2 Metrics and Baselines

We considered the General Language Understanding Evaluation (GLUE) benchmark in our experiments (benchmark details are given in Appendix A.4). As metrics, we considered standard measures that are Matthews Correlation for COLA and Spearman Correlation for STS-B, F1 score for MRPC/QQP, as well as accuracy. As baselines, we compared `Morris` to our implementations of the following approaches: **LoRA** (Hu et al., 2021), **Lora_BitFit** which combines **LoRA** and **BitFit** in (Hu et al., 2021; Ben Zaken et al., 2022), as well as **LoRA_BitFit_MTL** presented above. All experiments are done on 3 seeds and the results are the average value of the performances. For this part, we chose to not use the test online but we split each dev set into dev/test set.

We also compared `Morris` to single task models: **BERT** (Devlin et al., 2018), **Bitfit** (Ben Zaken et al., 2022), as well as, **Multi-task models** which are two extensions of **BERT** to this case (Glover and Hokamp, 2019), PALs (Stickland and Murray, 2019) and CA-MTL (Pilault et al., 2021). This comparison is done on the online test set, the best model on three seeds was kept for the comparison.

4.3 Results

We first begin to compare our approach with **LoRA** and its extensions. Performances are shown in Table 2. As a result, the average performance appears to be improved by a factor of 0.5 when the **LoRA** and **Bitfit** techniques are combined. Moreover, this approach seems to be efficient in a Multi-task Learning setting, as **LoRA_BitFit_MTL** increases the general performance by 0.6. Finally, `Morris` outperforms all our baseline. In addition, the use of the L_0 regularisation enables the model to decrease the number of parameters by a factor of 0.6 with no loss of performance. Given that all approaches employ training parameters of the same order of magnitude, comparisons in this situation are straightforward.

Model	Total Params	Trained params/ task	Avg
Single Task			
LoRA	x1.024	0.27%	80.7
Lora_BitFit	x1.03	0.34%	81.2
Multi Task			
LoRA_BitFit_MTL	x1.022	0.246%	81.8
Morris	x1.024	0.27%	82.2
Morris L0	**x1.014**	**0.16%**	**82.4**

Table 2: Experimental results on our build test set. The best result on each column is in bold face. The results are obtained by compute the mean of three Seeds

In a more general case, when `Morris`is compared to the other methods in Table 1, this is not the case. We first notice that `Morris` performs better than all single task approaches. Furthermore, our approach trains the model with less parameters than the other approaches by a higher factor in the multi-target case. Only the **CA-MTL** (Pilault et al., 2021) seems to be competitive with `Morris`. We justify this by pointing out that, in contrast to our sampling approach, the sampling strategy used in (Pilault et al., 2021) is highly extensive. In the general case, our approach is equivalent or better than the most of the baselines in term of average

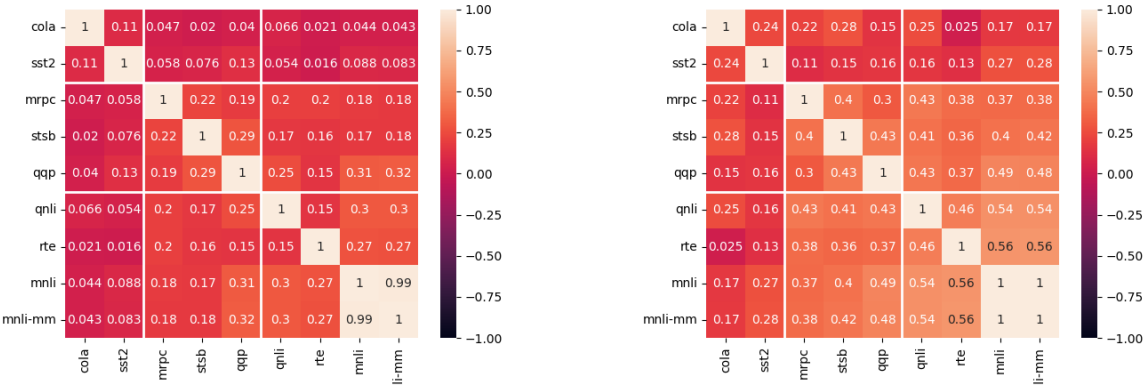

Figure 2: S^{weight} for w_q with L_0 regularisation without and with L_0 regularisation

performance or the number of parameters used in training.

4.4 Interaction between tasks

We assume that tasks are similar if their weight variations are similar. As a measure, we compare biases and slices using the cosine similarity:

$$S^{bias}_{[i,j]} = \frac{\langle \hat{B}_{[i,:]} \hat{B}_{[j,:]} \rangle}{\|\hat{B}_{[i,:]}\| \|\hat{B}_{[j,:]}\|}, \quad S^{weight}_{[i,j]} = \frac{\langle \mathcal{Q}_{[:,i,:]} \mathcal{Q}_{[:,j,:]} \rangle}{\|\mathcal{Q}_{[:,i,:]}\| \|\mathcal{Q}_{[:,j,:]}\|} \tag{5}$$

We will explore the weights (wq) and ($bm2$) due to their significant variance in light of prior works (Ben Zaken et al., 2022; Hu et al., 2021). For this, we examine the relationship between `Morris` and `Morris` with the L_0 regularization. In order to analyze more broad interactions, we often create NB similarity matrices by using Equation (5). We then decide to average these NB similarity matrices. These findings are presented in Figure 2 where it is clear that the diagonal block reflecting the kind of task has the greatest similarity score. Additionally, the coefficients are not close to 1, indicating that task-specific weights enable successful task differentiation. The bias similarity seeks to distinguish the **CoLA** and **SST-2** tasks from the other tasks, which are known to be uncorrelated one from another and regularization reduces task similarity.

5 Conclusion

In this paper we presented, a novel method for multi-task learning that relies on stacking the neural network weights into a tensor. We demonstrated that low-rank updates in the conventional polyadic tensor decomposition of this tensor of weights result in an efficient technique that allows for a significant reduction in model parameters without sacrificing performance. On the GLUE Benchmark, we showed that our proposed approach successfully achieves a compromise between maintaining positive transfer and reducing negative transfer by only using 0.3% of the initial model's parameters.

6 Limitations

The drawbacks of our method are the same as those of **LoRA**: it is tricky to batch inputs to many tasks with varying **A** and **B** in a single forward pass, and the rank may be greater for tasks that are more challenging. Moreover, we believe that weights obtained during a single task may be used for a better initialisation. Finally, the use of a different sampling policy on a different dataset may also be appropriate, however this choice is not obvious.

7 Acknowledgement

This work was supported by the ANR (Agence Nationale de Recherche) grants Lawbot (ANR-20-CE38-0013) and LeaFleT (ANR-19-CE23-0021).

References

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

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

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5931–5937, Florence, Italy. Association for Computational Linguistics.

Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *In North American Association for Com- putational Linguistics (NAACL)*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Chin-Lun Fu, Zih-Ching Chen, Yun-Ru Lee, and Hung-yi Lee. 2022. AdapterBias: Parameter-efficient token-dependent representation shift for adapters in NLP tasks. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2608–2621, Seattle, United States. Association for Computational Linguistics.

John Glover and Chris Hokamp. 2019. Task selection policies for multitask learning.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. 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)*, pages 4884–4896, Online. Association for Computational Linguistics.

Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. 2022. Hyperprompt: Prompt-based task-conditioning of transformers.

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.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. First quora dataset release: Question pairs.

Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. 2015. Compression of deep convolutional neural networks for fast and low power mobile applications. *In International Conference on Learning Representations*.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. 2014. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *Computer Science*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l_0 regularization.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *ACL*.

I. V. Oseledets. 2011. Tensor-train decomposition. volume 33, pages 2295–2317.

Jonathan Pilault, Amine Elhattami, and Christopher Joseph Pal. 2021. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. *International Conference on Representation Learning - ICLR*, abs/2009.09139.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Yuxin Ren, Benyou Wang, Lifeng Shang, Xin Jiang, and Qun Liu. 2022. Exploring extreme parameter compression for pre-trained language models.

Bernardino Romera-Paredes, Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil. 2013. Multilinear multitask learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1444–1452, Atlanta, Georgia, USA. PMLR.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4548–4557. PMLR.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Asa Cooolicyer Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671.

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

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Tianwen Wei, Jianwei Qi, and Shenghuan He. 2021. A flexible multi-task model for BERT serving.

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

Kishan Wimalawarne, Masashi Sugiyama, and Ryota Tomioka. 2014. Multitask learning meets tensor factorization: task imputation via convex optimization. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Sen Wu, Hongyang R Zhang, and Christopher Ré. 2020. Understanding and improving information transfer in multi-task learning. In *International Conference on Representation Learning - ICLR*.

Yongxin Yang and Timothy M. Hospedales. 2017. Deep multi-task representation learning: A tensor factorisation approach. In *International Conference on Learning Representations*.

A Appendix

A.1 CP decomposition

Tensor compression can be carried out using the potent technique of canonical polyadic (CP) decomposition.

The aim here is to approximate an N-way tensor where $N \geqslant 3$ by low-rank tensor, that can be written as a sum of rank-one tensors. For the sake of presentation and without loss of generality, let $N = 3$ and $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ be a 3-way tensor. This tensor has rank one if and only if there exists three vectors $\mathbf{a} \in \mathbb{R}^{I_1}$, $\mathbf{b} \in \mathbb{R}^{I_2}$ and $\mathbf{c} \in \mathbb{R}^{I_3}$ such that:

$$\mathcal{X}_{ones} = \mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c} \tag{6}$$

where $\otimes$ is the tensor (outer) product operation. A general CP representation of a 3-way tensor $\mathcal{X}$ is of the form:

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}^{(r)} \otimes \mathbf{b}^{(r)} \otimes \mathbf{c}^{(r)}, \tag{7}$$

where R is an integer. The smallest R such that Equation: 7 is verified, is called the rank of the tensor $\mathcal{X}$. For convenience, the vectors in the previous expression (7) can be stacked into the factor matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times R}$, $\mathbf{B} \in \mathbb{R}^{I_2 \times R}$ and $\mathbf{C} \in \mathbb{R}^{I_3 \times R}$ such as the i^{th} columns of $\mathbf{A}$ is the vector $\mathbf{a}^{(i)}$ (same reasoning for $\mathbf{B}$ and $\mathbf{C}$).

In the sequel, the goal is to approximate neural network tensors by low-rank tensors of the form.

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{A}_{[:,r]} \otimes \mathbf{B}_{[:,r]} \otimes \mathbf{C}_{[:,r]} \tag{8}$$

A.2 Training and initialisation

A crucial aspect of deep learning is initialization. Inspired by general approach, our adding part has to be equal to zero at the beginning of our training. The natural choice is to initialize the specific task matrix of each layer $\mathbf{B}$ and $\mathbf{E}$ at zero. The rest of our adding parameters are initialized randomly.

A.3 Parameter Efficiency

In this section, we investigate the number of training parameters. We note $\theta^\star$ the parameters of the frozen BERT, and we note the set of our adding matrix index by the number of blocks NB, and by the number of bias per block nb_{bias}: $\{\mathbf{A}^j, \mathbf{B}^j, \mathbf{C}^j\}_{j=1}^{2\times NB}, \{\mathbf{D}^j, \mathbf{E}^j\}_{j=1}^{nb_{bias}\times NB}$. In this case the model parameters are: $\Theta = (\theta^\star, \{\mathbf{A}^j\}_{j=1}^{2\times NB}, \{\mathbf{C}^j\}_{j=1}^{2\times NB}, \{\mathbf{D}^j\}_{j=1}^{nb_{bias}\times NB})$ and $\{\phi_i\}_{i=1}^{T} = (\{\mathbf{B}^j_{[i,:]}\}_{j=1}^{2\times NB}, \{\mathbf{E}^j_{[:,i]}\}_{j=1}^{NB})$.

The number of trainable shared parameters is equal to $|\Theta \backslash \theta^\star| = NB \times d \times (4r_w + r_b \times nb_{bias})$ and the number of specific task parameters is equal to $|\{\phi_i\}_{i=1}^{T}| = NB \times T(2r_w + r_b \times nb_{bias})$. In the case where the regularisation L_0 is applied, we considered that these parameters are negligible. The number of added parameters depends linearly on the number of tasks but does not depend on the dimension of the hidden space d which makes our approach efficient in terms of parameters.

A.4 Dataset

We considered the General Language Understanding Evaluation (GLUE) benchmark in our experiments. This benchmark is composed of a large variety of task like *Single-Sentence Classification*: CoLA (Warstadt et al., 2018), SST-2(Socher et al., 2013), *Similarity and Paraphrase tasks*: MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017), QPP (Iyer et al., 2017) and *Inference Tasks*: MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005) and WNLI(Levesque et al., 2012).

Following other studies, we did not take into account WNLI (Levesque et al., 2012) and we considered MNLI to be composed of two different tasks: MNLI-m (matched) and Mnli-mm (mismatched) more details can be found in Table 3.

A.5 Interaction between task

When it comes to mode-2 unfolding, the cosine similarity between slices is comparable. With our tensor representation, the following may be efficiently computed:

$$S^{weight} = normalize(\mathbf{Q}_{(2)}\mathbf{Q}_{(2)}^T) \tag{9}$$

$$S^{weight} = normalize(\mathbf{B}(\mathbf{C} \odot \mathbf{A})^T(\mathbf{C} \odot \mathbf{A})\mathbf{B}^T) \tag{10}$$

$$S^{weight} = normalize(\mathbf{B}(\mathbf{C}^T\mathbf{C} * \mathbf{A}^T\mathbf{A})\mathbf{B}^T) \tag{11}$$

Tasks	Corpus	|Train|	|Test|
CoLA (Warstadt et al., 2018)	Corpus of Linguistic Acceptability	8.5K	1K
SST-2 (Socher et al., 2013)	Stanford Sentiment Treebank	67K	1.8K
MRPC (Dolan and Brockett, 2005)	Microsoft Research Paraphrase Corpus	3.7K	1.7K
STS-B (Cer et al., 2017)	Semantic Textual Similarity Benchmark	7K	1.4K
QQP (Iyer et al., 2017)	Quora Question Pairs	364K	391K
MNLI (Williams et al., 2018)	Multi-Genre NLI	393K	20K
QNLI (Rajpurkar et al., 2016)	Question NLI	105K	5.4K
RTE (Dagan et al., 2005)	Recognition Textual Entailment	2.5K	3K
WNLI (Levesque et al., 2012)	Winograd NLI	634	146

Table 3: Presentation of tasks in GLUE (Wang et al., 2018) with their corresponding training and test sets sizes.

Algorithm 1: Training of `Morris`

Input: Dataset $\{D_i\}_{i=1}^{T}$;
Loss function per task $\{l_i\}_{i=1}^{T}$
Task sampling policy: P;

if *Apply Regularisation L0* **then**
  for $step \leftarrow 1$ **to** 10^4 **do**
    Select one task 't' according to P;
    Select one batch $b_t = (X_t, Y_t) \in D_t$;
    Compute the loss $l_t(f(\Theta, \phi_t \circ z, X_t), Y_T)$;
    Update Θ, ϕ_t and α;
  end
  Threshold on α:;
  $\alpha[\alpha < 0.5] = -10$;
  $\alpha[\alpha > 0.5] = 10$
end
for $step \leftarrow 1$ **to** 2^{18} **do**
  Select one task 't' according to P;
  Select one batch $b_t = (X_t, Y_t) \in D_t$;
  Compute the loss $l_t(f(\Theta, \phi_t, X_t), Y_T)$;
  Update Θ and ϕ_t;
end

Output: Trained Parameters Θ and $\{\phi_i\}_{i=1}^{T}$

A.6 Implementation details

We utilized the pre-trained BERT base uncased offered by Hugging Face[2]. For the optimizer AdamW Loshchilov and Hutter (2017), we use a linear decay with a warmup of 0.06 and gradient clipping for all experiments. Our model is evaluated each 2000 steps for a total of 2^{18} training steps. Only the best checkpoint on average is kept. For LoRA approach we choose a rank equal to eight which seems to be very efficient, for number of epochs we also followed the instruction in LoRA (Hu et al., 2021) used for the Roberta model. Our model is training like in (Liu et al., 2019) .

[2]https://huggingface.co/bert-base-uncased

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
6

☒ A2. Did you discuss any potential risks of your work?
Our work does not involve any risk.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

☒

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

4 and A (appendix)

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We report the type of GPU used in appendix the rest of the computation budget is academic.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4 and A (appendix)

☒ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
The main results are obtained online (https://gluebenchmark.com) with the test set which has a limited number of submission per day.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.