DyLoRA: Parameter Efficient Tuning of Pre-trained Models using Dynamic Search-Free Low Rank Adaptation

Mojtaba Valipour
David R. Cheriton School of Computer Science
University of Waterloo
mojtaba.valipour@uwaterloo.ca

Mehdi Rezagholizadeh
Huawei Noah's Ark Lab
mehdi.rezagholizadeh@huawei.com

Ivan Kobyzev
Huawei Noah's Ark Lab
ivan.kobyzev@huawei.com

Ali Ghodsi
Department of Statistics and Actuarial Science
University of Waterloo
ali.ghodsi@uwaterloo.ca

Abstract

With the ever-growing size of pre-trained models (PMs), fine-tuning has become more expensive and resource-hungry. As a remedy, low-rank adapters (LoRA) keep the main pre-trained weights of the model frozen and just introduce some learnable truncated SVD modules (so-called LoRA blocks) to the model. While LoRA blocks are parameter efficient, they suffer from two major problems: first, the size of these blocks is fixed and cannot be modified after training (for example, if we need to change the rank of LoRA blocks, then we need to train them from scratch); second, optimizing their rank requires an exhaustive search. In this work, we introduce a dynamic low-rank adaptation (DyLoRA) solution to address these two problems together. Our DyLoRA method trains LoRA blocks for a range of ranks instead of a single rank by sorting out the representation learned at different ranks during training. We evaluate our solution on different tasks in the GLUE benchmark using the RoBERTa model. Our results show that we can train DyLoRA at least $7x$ faster than LoRA without compromising performance significantly. Moreover, our models can perform consistently well on a much larger range of ranks compared to LoRA[1]

1 Introduction

Pre-training/fine-tuning has become a popular paradigm for solving many tasks in natural language processing (NLP) [2]–[4] and Computer Vision [5]–[10]. Pre-trained models (PMs) such as pre-trained language models (PLMs) [2], [4], and pre-trained visual-language models [11]–[14] have advanced a lot in recent years. With the ever growing size of these pre-trained models, fine-tuning on downstream tasks become more expensive. There are two category of solutions in this regard: first, model compression [15], [16]; second, parameter efficient tuning (PET) [17]–[19].

There are many different model compression techniques for PLMs such as matrix factorization [20]–[22], pruning [23], quantization [24], [25], and knowledge distillation [15], [26]–[31]. There are also different types of PET techniques in the literature such as low-rank adapters [18], [32]–[34], and prompt-based techniques [35].

[1]The code is available at https://github.com/huawei-noah/KD-NLP.

Efficient Natural Language and Speech Processing (NeurIPS 2022).

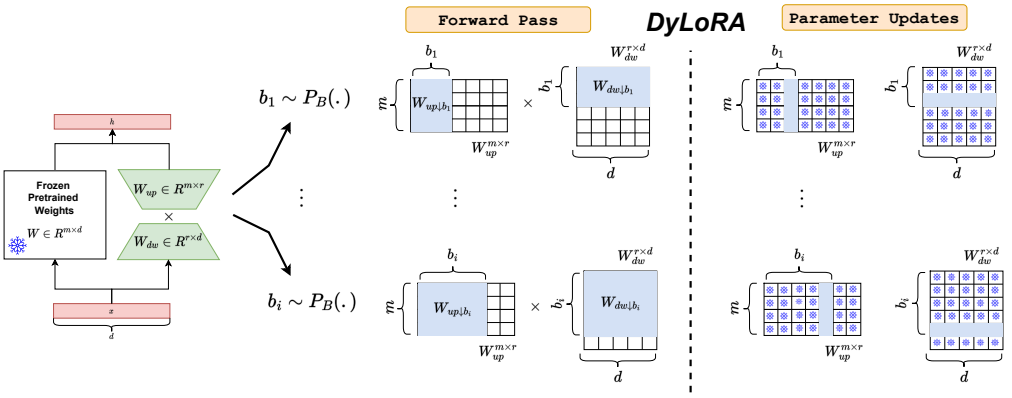

Figure 1: DyLoRA: The overall diagram of our proposed method. During training, in each iteration, we sample from a pre-defined random distribution which will help us to truncate the up-projection and down-projection matrices in the LoRA [1] objective.

Although model compression solutions are well-established in recent years in the literature, using these techniques usually leads to degrading the model performance more drastically compared to PET techniques; or in some cases such as knowledge distillation, it requires having a fine-tuned larger teacher model. Among all PET techniques, low-rank adapters have received a lot of attention because in contrast to prompt-tuning techniques, low-rank adapters do not add to the sequence length. Even though there are several well-known low-rank adaptation techniques in the literature such as [33], compacter [18], LORA [34]; they all suffer from two major common problems: first, it is not clear how to select the size of their rank (while their performance is very sensitive based on this rank selection); second, their training is static meaning that if a low-rank model is trained based on a particular rank size, it will not work well in other rank values (i.e. for any other rank value we need to train a separate model).

In this paper, we propose a dynamic low-rank adapter (DyLoRA) to alleviate these two problems. Without losing generality, we focus on LoRA[1] and train LoRA blocks for a range of ranks instead of a single rank by sorting out the representation learned at different ranks during training.

Our model is able to outperform LoRA in much wider range of ranks without adding to the training time. Moreover, our technique does not need any extra training for searching across ranks.

The following is a summary of our contributions:

- Dynamic LoRA: On top of LoRA, we developed a new algorithm (DyLoRA) that makes it dynamic at inference time without incurring extra cost.
- Search-free LoRA: We demonstrate that by making a negligible compromise in performance, it is possible to avoid the costly search process of choosing the optimal rank for LoRA.

2 Related Work

LORA [34] is a low-rank up-projection/down-projection transformation without any non-linearity that is applied in parallel to key and value attention matrices. The main benefit of LORA is that the adapter module after training can be integrated into the original weight matrices of the model, which in turn can lead to a very efficient inference time. All the available low-rank adapters [18], [33], [36] in the literature, however, suffer from two major issues: first, finding the best rank size requires a heavy exhaustive training and search; second, the tuned adapter module works well only with a particular rank size.

While there has been some efforts in the literature towards dynamic networks such as DynaBERT [37] and GradMax [38], to the best of our knowledge, this problem for factorized networks and low-rank adapters is still open. DRONE [16] propose a technique for data-aware low-rank model compression

however their approach is not search-free and also it is not dynamic. DynaBERT introduces a two-stage method to train width and depth-wise dynamic networks. However, DynaBERT requires a fine-tuned teacher model on the task to train its dynamic sub-networks which is a major limitation dealing with large PLMs. GradMax is a technique which gradually adds to the neurons of a network without touching the already trained neurons. But it is not clear how GradMax can be deployed for reducing the rank-search problem in low-rank adapters. [23] propose a structured pruning technique which is called factorized low-rank pruning (FLOP). FLOP decomposes weight matrices of a network into sum of rank-1 components which are regularized during training to gain sparsity. It is worth mentioning that FLOP aims at compressing the main model and even if it can be used for finding a good rank in lower-rank representation of full weight matrices, the final low-rank model will not be dynamic (i.e. it is trained well only for one rank and not a range of ranks, same as LoRA.).

3 Our Method: DyLoRA

In this section, we introduce our solution to get dynamic low-rank adapters that can be trained and deployed well on a range of ranks instead of a single particular rank (with a fixed training budget). This flexibility can free us from searching for the best ranks by training the model multiple times.

Without loss of generality, we explain our solution on top of LoRA. In each LoRA module, we have an up-projection ($W_{up} \in \mathbb{R}^{m \times r}$) and a down-projection matrix ($W_{dw} \in \mathbb{R}^{r \times d}$). Let's assume that we would like to train the LoRA module to operate in the range of $r \in \text{Range}[r_{min}, r_{max}]$ where r_{min} and r_{max} can be treated as new hyper-parameters. To make the LoRA module work in a range of ranks instead of a single rank, we need to ensure that increasing or decreasing the rank will not significantly hamper the model's performance. One way to implement such behavior would be by sorting the information content of different ranks in the training process of LoRA modules. In this regard, at each training step, we sample $b \sim p_B(.), b \in \{r_{min}, r_{min}+1, ..., r_{max}\}$ form a pre-defined categorical distribution (which has a support in $\text{Range}[r_{min}, r_{max}]$) and truncate W_{dw} and W_{up} matrices accordingly.

$$\begin{aligned} W_{dw\downarrow b} &= W_{dw}[1:b,:] \\ W_{up\downarrow b} &= W_{up}[:,1:b] \end{aligned} \tag{1}$$

$W_{dw\downarrow b}$ and $W_{up\downarrow b}$ are b-truncated versions of W_{dw} and W_{up} respectively (see Fig. 1 for the visualization). Moreover, let's define W^b_{dw} as the b^{th} row of W_{dw}; W^b_{up} corresponds to the b^{th} column of W_{up}.

$$\begin{aligned} W^b_{dw} &= W_{dw}[b,:] \\ W^b_{up} &= W_{up}[:,b] \end{aligned} \tag{2}$$

Then, the forward pass of this truncated LoRA module during training will be calculated as following:

$$h = W_0 x + \frac{\alpha}{b} W_{up\downarrow b} W_{dw\downarrow b} x \tag{3}$$

For the sake of simplicity, let's assume that we have only one LoRA module in the network (the one which is described in Eq. 3). Let's first consider the regular static loss function ($\mathcal{L}^{\mathcal{S}}$) of the network $f(x; W_{dw}, W_{up})$ with W_{dw} and W_{up} tunable parameters for N given input-output pairs $(\mathbf{x}, \mathbf{y}) = (x_i, y_i)_{i=1}^N$:

$$\begin{aligned} &\min_{W_{dw}, W_{up}} \mathcal{L}^{\mathcal{S}}(\mathbf{x}, \mathbf{y}; W_{dw}, W_{up}) \triangleq \\ &\sum_{i=1}^{N} l(f(x_i; W_{dw}, W_{up}), y_i). \end{aligned} \tag{4}$$

where $l(f, \mathbf{y})$ is a loss function that measures the divergence of network predictions compared with the target labels. Then, let's extend the training loss to make the network dynamic considering the b-truncation process. We can define our dynamic loss function $\mathcal{L}^{\mathcal{DY}}$ as follows.

$$\mathcal{L}^{\mathcal{DY}}_{\downarrow b} = \sum_{i=1}^{N} l(f(x_i; W_{dw\downarrow b}, W_{up\downarrow b}), y_i). \tag{5}$$

In the parameter update phase, it is also possible to update only the b^{th} corresponding row and column sampled in the truncation phase in order to avoid affecting the lower rank parameters.

$$
\begin{aligned}
W_{dw}^{b} &\leftarrow W_{dw}^{b} - \eta \nabla_{W_{dw}^{b}} \mathcal{L}_{\downarrow b}^{\mathcal{DY}} \\
W_{up}^{b} &\leftarrow W_{up}^{b} - \eta \nabla_{W_{up}^{b}} \mathcal{L}_{\downarrow b}^{\mathcal{DY}}
\end{aligned}
\tag{6}
$$

4 Experiments

4.1 LoRA rank selection problem

There is no clear guidance on how to determine the rank for LoRA algorithm. As can be seen from several experiments in LoRA [1] paper, performance of different ranks does not indicate any clear trend. We also observe the same problem in the GLUE benchmark. We may argue that theoretically, the rank with the best performance is always the highest. High ranks, however, introduce additional parameters into the adaptive process and this might be undesirable. In practice, the most effective rank differs depending on the task.

4.2 Dynamic low rank adaptation

Model	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg
The average performance over all ranks (1 to 8)									
LoRA	56.63	89.15	78.83	32.07	73.57	79.61	64.35	59.64	66.73
DyLoRA	**86.21**	**94.12**	**88.94**	**59.12**	**91.55**	**88.07**	**81.27**	**90.90**	**85.01**
Performance over Rank = 8 only									
LoRA (8)	**87.09**	**94.84**	87.25	60.82	**92.59**	**90.69**	**84.84**	**91.47**	**86.20**
DyLoRA (8)	86.51	94.04	**89.46**	**61.12**	91.84	88.73	83.75	91.43	85.86

Table 1: In this table, the task is to find a low-rank adaptation matrix that works with different ranks at inference time given a fixed budget (training time). In all experiments, the pre-training model is RoBERTa base [3]. The accuracy (matched and mismatched) for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, as well as accuracy for other tasks are reported.

For example, suppose we have a neural network that we wish to deploy on various devices with different configurations. The use of higher ranks may pose a problem for very sensitive devices as they have a greater number of parameters. Therefore, we must either train several models with different configurations, or find the most optimal rank. The cost associated with this is significant, as even in the setting of LoRA, we are required to find the best rank for each task and each device. Using DyLoRA, however, one needs to train one model per task and, as our method is adaptive at inference time, we are able to deploy it according to our needs. In Table 1, we demonstrate the dynamic properties of DyLoRA. In LoRA, we lose performance when performing inferences for the lower ranks. This occurs because the model has been trained only for rank 8 during training. In DyLoRA, we preserve a high level of performance for lower ranks while competing well with LoRA on rank 8.

5 Conclusion

In this paper, we presented our solution DyLoRA, to address two problems in low-rank adapters regarding rank selection and making them dynamic. We showed that DyLoRA can select the rank without requiring multiple re-training and can make LoRA dynamic at inference time. As a result, we are able to avoid the process of searching for the most optimal ranks for many real-life scenarios. It has been demonstrated that DyLoRA performance is comparable with LoRA, yet we can support a wider range of ranks without adding any additional time.

References

[1] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[4] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] A. Howard, M. Sandler, G. Chu, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[9] M. Chen, A. Radford, R. Child, *et al.*, "Generative pretraining from pixels," in *International conference on machine learning*, PMLR, 2020, pp. 1691–1703.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[11] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in neural information processing systems*, vol. 32, 2019.

[12] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," *arXiv preprint arXiv:1908.03557*, 2019.

[13] W. Su, X. Zhu, Y. Cao, *et al.*, "Vl-bert: Pre-training of generic visual-linguistic representations," *arXiv preprint arXiv:1908.08530*, 2019.

[14] Q. Xia, H. Huang, N. Duan, *et al.*, "Xgpt: Cross-modal generative pre-training for image captioning," in *CCF International Conference on Natural Language Processing and Chinese Computing*, Springer, 2021, pp. 786–797.

[15] A. Jafari, M. Rezagholizadeh, P. Sharma, and A. Ghodsi, "Annealing knowledge distillation," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Online: Association for Computational Linguistics, Apr. 2021, pp. 2493–2504. [Online]. Available: `https://aclanthology.org/2021.eacl-main.212`.

[16] P. Chen, H.-F. Yu, I. Dhillon, and C.-J. Hsieh, "Drone: Data-aware low-rank compression for large nlp models," *Advances in neural information processing systems*, vol. 34, pp. 29 321–29 334, 2021.

[17] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2790–2799.

[18] R. Karimi Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1022–1035, 2021.

[19] Y. Mao, L. Mathias, R. Hou, *et al.*, "Unipelt: A unified framework for parameter-efficient language model tuning," *arXiv preprint arXiv:2110.07577*, 2021.

[20] M. B. Noach and Y. Goldberg, "Compressing pre-trained language models by matrix decomposition," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 2020, pp. 884–889.

[21] M. S. Tahaei, E. Charlaix, V. P. Nia, A. Ghodsi, and M. Rezagholizadeh, "Kroneckerbert: Learning kronecker decomposition for pre-trained language models via knowledge distillation," *arXiv preprint arXiv:2109.06243*, 2021.

[22] K. Kumar, P. Passban, M. Rezagholizadeh, Y. Lau, and Q. Liu, "From fully trained to fully random embeddings: Improving neural machine translation with compact word embedding tables," 2022.

[23] Z. Wang, J. Wohlwend, and T. Lei, "Structured pruning of large language models," *arXiv preprint arXiv:1910.04732*, 2019.

[24] C. Tao, L. Hou, W. Zhang, *et al.*, "Compression of generative pre-trained language models via quantization," *arXiv preprint arXiv:2203.10705*, 2022.

[25] G. Prato, E. Charlaix, and M. Rezagholizadeh, "Fully quantized transformer for machine translation," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1–14.

[26] L. Li, Y. Lin, S. Ren, P. Li, J. Zhou, and X. Sun, "Dynamic knowledge distillation for pre-trained language models," *arXiv preprint arXiv:2109.11295*, 2021.

[27] P. Passban, Y. Wu, M. Rezagholizadeh, and Q. Liu, "ALP-KD: attention-based layer projection for knowledge distillation," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, AAAI Press, 2021, pp. 13 657–13 665. [Online]. Available: `https://ojs.aaai.org/index.php/AAAI/article/view/17610`.

[28] E. Kamalloo, M. Rezagholizadeh, P. Passban, and A. Ghodsi, "Not far away, not so close: Sample efficient nearest neighbour data augmentation via minimax," *arXiv preprint arXiv:2105.13608*, 2021.

[29] M. Rezagholizadeh, A. Jafari, P. Salad, P. Sharma, A. S. Pasand, and A. Ghodsi, "Pro-kd: Progressive distillation by following the footsteps of the teacher," *arXiv preprint arXiv:2110.08532*, 2021.

[30] A. Rashid, V. Lioutas, and M. Rezagholizadeh, "Mate-kd: Masked adversarial text, a companion to knowledge distillation," *arXiv preprint arXiv:2105.05912*, 2021.

[31] Y. Wu, P. Passban, M. Rezagholizadeh, and Q. Liu, "Why skip if you can combine: A simple knowledge distillation technique for intermediate layers," *arXiv preprint arXiv:2010.03034*, 2020.

[32] R. Wang, D. Tang, N. Duan, *et al.*, "K-adapter: Infusing knowledge into pre-trained models with adapters," *arXiv preprint arXiv:2002.01808*, 2020.

[33] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2790–2799.

[34] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[35] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[36] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," *arXiv preprint arXiv:2110.04366*, 2021.

[37] L. Hou, Z. Huang, L. Shang, X. Jiang, X. Chen, and Q. Liu, "Dynabert: Dynamic bert with adaptive width and depth," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9782–9793, 2020.

[38] U. Evci, M. Vladymyrov, T. Unterthiner, B. van Merriënboer, and F. Pedregosa, "Gradmax: Growing neural networks using gradient information," *arXiv preprint arXiv:2201.05125*, 2022.