PC-LoRA: Progressive Model Compression with Low Rank Adaptation

Injoon Hwang[1]*, **HaeWon Park**[1]*, **Jooyoung Yang**[1], **SunJae Maeng**[1], **Youngwan Lee**[1 2 3]†
[1]MODULABS, South Korea
[2]Electronics and Telecommunications Research Institute (ETRI), South Korea
[3]Korea Advanced Institute of Science and Technology (KAIST), South Korea

Abstract

This work presents Progressive Compression LoRA (PC-LoRA), a novel extension of Low-Rank Adaptation (LoRA), designed to enable model compression and parameter-efficient fine-tuning. To mitigate the computational costs of large-scale models, PC-LoRA introduces a approach of decaying model weights to zero. This method allows to model compression and efficient fine-tuning by progressively reducing the pre-trained weights during the fine-tuning process until they are completely removed. Through empirical analysis on various models, we demonstrate that PC-LoRA significantly reduces computational costs with minor performance degradation. Compared to full fine-tuning and LoRA fine-tuning, PC-LoRA shows an average performance drop of -3.085%. Despite this, our method substantially compresses models, by 94.1% / 89.1% in parameters and FLOPs for vision models, and achieves a 93.5% parameter and 84.2% Flops reduction for NLP models.

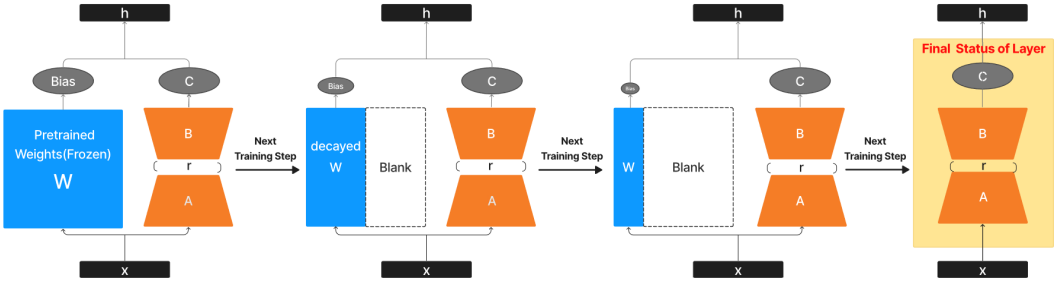

Figure 1: Progressive Compression LoRA (PC-LoRA): The overall diagram of PC-LoRA method. At each training step, the pre-trained weights 'W' gradually decay, eventually disappears and only the Low Rank Adapter corresponding weights 'A', 'B', and 'C' remain. Weights A and B are the low-rank (r) adapters' weights, and C is the low-rank adapter's bias.

1 Introduction

Since the introduction of pre-trained models from the Transformer (Vaswani et al., 2017), they have demonstrated exceptional performance across various Natural Language Processing (NLP) (Brown et al., 2020; Devlin et al., 2019) and Computer Vision (CV) (Dosovitskiy et al., 2021; Carion et al., 2020; Wang et al., 2021) tasks. However, their massive size and computational costs present challenges in deployment and fine-tuning. To overcome these obstacles, parameter-efficient fine-tuning methods such as Prefix Tuning (Li & Liang, 2021), P-Tuning (Liu et al., 2021), Prompt Tuning (Lester et al., 2021), adapters (Houlsby et al., 2019), and Low-Rank Adaptation (LoRA) (Hu et al., 2022) have been developed. LoRA, in particular, introduces trainable low-rank matrices to transformer layers, significantly reducing the number of trainable parameters for fine-tuing. However,

*These authors contributed equally to this work.
†Corresponding author.

since LoRA relies on the pre-trained model as a backbone, it still demands significant memory for inference, similar to the original pre-trained model.

Based upon the foundation of LoRA, we question "*Can we make it representative enough with just LoRA weights at the final phase of finetuning without the pre-trained weights?*" To find the answer to this question, we propose a **Progressive Compression-LoRA (PC-LoRA)** method, which gradually reduces the pre-trained weights (i.e., the base weights) during fine-tuning until they are completely removed as shown in Figure 1. We intentionally decay the original weights of a model, and this process ensures that the low-rank adapter learns the original weights' representations, while simultaneously updating the weights for downstream tasks. By this approach, PC-LoRA achieves a dual purpose of low-rank compression and parameter-efficient fine-tuning.

PC-LoRA provides several benefits, such as decreased floating-point operations (FLOPs), a more compact model size, and versatility due to its layerwise approach. Our method cuts computational costs significantly while preserving accuracy, achieving up to 94.1%p parameter reduction and 89.1%p Flops decrease for CV models, and up to 93.5%p parameter and 84.2%p Flops reduction for NLP models. Additionally, due to its orthogonality to other model compression strategies, such as quantization (Jacob et al., 2017; Dettmers et al., 2023) and knowledge distillation (Romero et al., 2015; Hinton et al., 2015), PC-LoRA can be combined with various compression methods.

2 Related Works

Low-Rank Adaptation (LoRA) LoRA optimize the fine-tuning by adding a small set of trainable parameters, while the original model parameters remain frozen. This is achieved by constraining the update of a pre-trained weight matrix W_0 to a low-rank structure, represented as $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank r is significantly smaller than the minimum of d and k. During training, W_0 is kept static, and gradient updates are applied only to A and B. The adaptation is applied as $h = W_0x + \Delta Wx = W_0x + BAx$, ensuring that W_0 and $\Delta W = BA$ are multiplied by the same input x, and their output vectors are summed coordinate-wise to yield the modified output h. Our approach adds weight decay scheduling and a bias term to LoRA, decaying parameters for model compression and enhancing weight restoration.

Low-rank factorization Low-rank factorization is a technique to reduce model size by decomposing weight matrices into products of smaller matrices. Singular Value Decomposition (SVD) is typically employed, where a matrix A is split into $U\Sigma V^T$: U and V^T are orthogonal matrices containing left and right singular vectors, respectively, and Σ is a diagonal matrix of singular values. This compression method has evolved to include both fixed and variable rank approaches during training, with the former utilizing SVD or tensor decomposition on pre-trained networks (Lebedev et al., 2015; Sainath et al., 2013) and the latter adapting the rank dynamically (Ioannou et al., 2016; Wen et al., 2017). Despite the sophistication of these methods, the process still revolves around SVD, demanding high computational resources. However, our PC-LoRA approach achieves Low Rank Approximation without needing actual SVD operations.

3 Approach: PC-LoRA

We present Progressive Compression with Low-Rank Adaptation, called PC-LoRA, designed to incrementally compress a model by progressively reducing and eventually eliminating pre-trained weights during fine-tuning. In this method, the original weight, along with the LoRA weight, is used together in the forward pass output, where the original weight gradually decays and eventually disappears, leaving only the LoRA weight as the final result. More detailed information will be provided in the following paragraph. A detailed diagram illustrating our approach and its underpinning principles can be found in Figure 1.

PC-LoRA Layer: Similar to the original LoRA method, our PC-LoRA replaces the model's linear layers with PC-LoRA linear layers while preserving original weights and biases. It introduces a decay factor d, progressively reducing the influence of the original model parameters. LoRA weight C controls the bias, allowing to correct bias dynamically. Initially, A is set with a random Gaussian distribution, B at zero, and C also at zero, so BA starts at zero.

Training Configuration: The model's output is computed by combining the decay factor-scaled original output (D_i) and the LoRA output (L_i). Here, i represents the index of the current LoRA layer in the model. While training, the final output (F_i) is computed as:

$$D_i = d(W_i(x_i)), \tag{1}$$
$$L_i = B_i(A_i(x_i)) + C_i, \tag{2}$$
$$F_i = D_i + L_i. \tag{3}$$

LoRA weights A_i, B_i, and C_i are updated during fine-tuning, while the original weights (W_i) and bias are frozen. The bias term is omitted for simplicity in Equation 1. PC-LoRA method's loss optimization is based on the target (ground-truth), for instance, in a classification task, we utilize classification labels and optimize it using classification loss such as cross-entropy.

After progressive compression has been completed through training, the model relies only on LoRA output (L_i), aiming to replace the original model weights with the LoRA weights after training:

$$F_i = L_i \tag{4}$$

We also investigated the feasibility of training solely with Equation 4, without our weight decay scheduling method; we found that learning was completely unattainable.

Weight Decay Scheduling: The process follows three phases. In the **Warm-Up (0%-5% Iterations)** phase, $d(n) = 1$ which allows the model to fully utilize the original outputs and adjust to new data without initiating weight decay. During the **Decay (5%-85% Iterations)** phase, $d(n)$ transitions from 1 to 0, gradually diminishing the influence of the original weights and shifting to LoRA weights. Finally, in the **Cool-Down (≥85% Iterations)** phase, $d(n) = 0$ where the focus shifts to fine-tuning with LoRA weights exclusively, leading to a compact model that preserves performance.

The equation and graph of decay factor d is as follows:

$$d(n) = \begin{cases} 1 & \text{for } n \leq a \\ -\frac{n}{b-a} + \frac{a+b}{b-a} & \text{for } a < n < b \\ 0 & \text{for } n \geq b \end{cases} \tag{5}$$

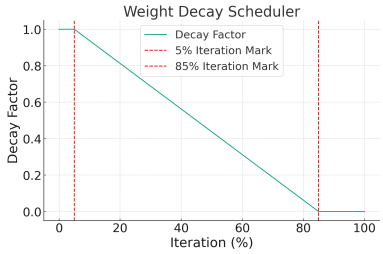

Figure 2: Weight Decay Scheduler

n represents the current step, $a = \lfloor 0.05N \rfloor$ and $b = \lfloor 0.85N \rfloor$ indicate decay start and end steps, respectively, for total iterations N.

To summarize, during training, our method utilizes both the decayed original output and the LoRA output as per Equation 3 ; however, the final outcome is composed exclusively of the LoRA output, following the form of Equation 4. Consequently, our approach involves fine-tuning through updates to the weights corresponding to the LoRA output, while simultaneously achieving model compression.

4 Experiments

4.1 Implementation Details

In our study, we assessed model performances in two benchmark tasks: text classification using the IMDb (Maas et al., 2011) dataset and image classification with the CIFAR-10 (Krizhevsky, 2009) dataset. Training details are provided in Appendix A and B. We conducted comparisons of PC-LoRA against two baseline methods. The first baseline method, `Full Fine-Tuning (Full-FT)`, involves updating all the parameters of a pre-trained model. The second baseline method, `LoRA Fine-Tuning (LoRA-FT)` integrates LoRA into the pre-trained model. LoRA was applied

CIFAR10					IMDb				
Model	Setting	Acc(%)	GFlops	Param(M)	Model	Setting	Acc(%)	GFlops	Param(M)
ViT-base/DINO	Full-FT	97.97	16.87	85.81	BERT	Full-FT	94.00	48.37	109.48
	LoRA-FT	97.86	16.87	85.81		LoRA-FT	94.01	48.37	109.48
	PC-LoRA	93.28	1.06	5.59 (94.1%↓)		PC-LoRA	90.83	7.599	29.34 (93.5%↓)
ViT-base/Clip	Full-FT	98.07	16.87	85.81	XLNET	Full-FT	94.00	60.45	117.31
	LoRA-FT	98.14	16.87	85.81		LoRA-FT	95.21	60.45	117.31
	PC-LoRA	93.90	1.07	5.59 (94.1%↓)		PC-LoRA	93.53	32.97	63.12 (58.3%↓)
R50-ViT-base	Full-FT	98.72	20.94	97.90	ELECTRA	Full-FT	95.33	48.37	109.48
	LoRA-FT	98.60	20.94	97.90		LoRA-FT	94.63	48.37	109.48
	PC-LoRA	97.11	5.14	17.68 (94.1%↓)		PC-LoRA	91.12	7.599	29.34 (93.5%↓)
ViT-base/MAE	Full-FT	96.20	16.87	85.81	ROBERTA	Full-FT	94.67	48.37	124.65
	LoRA-FT	96.12	16.87	85.81		LoRA-FT	94.40	48.37	124.65
	PC-LoRA	93.90	1.06	5.59 (94.1%↓)		PC-LoRA	90.62	7.599	44.50 (93.6%↓)

Table 1: Comparison of PC-LoRA with Full-FT, and LoRA-FT methods applied on various pre-trained models with CIFAR10 and IMDb Benchmarks. GFlops and Params are measured during inference. The values in parentheses indicate the reduction in parameters of the model with LoRA-FT applied, compared to when PC-LoRA is applied, both excluding the embedding layers.

across all linear layers except for the embedding layers, with a set rank of 32. For PC-LoRA method, we adopted the same settings with LoRA-FT baseline. As previously described in our method, training was conducted solely on the layers that were adapted to low rank.

4.2 Main Results

Table 1 presents a comparative analysis of the performance of various vision and language models on the CIFAR10 and IMDbbbbenchmarks, utilizing Full-FT, LoRA-FT, and the proposed PC-LoRA method. The performance of Full-FT and LoRA-FT is similar. When compared to the LoRA-FT method, which uses the same number of parameters for training, the PC-LoRA method shows an average performance degradation of about -3.085%p. Despite such performance degradation, the final outcome of PC-LoRA is a compressed model, resulting in a reduction of 94.1%p excluding the embedding layers, and an average 89.1%p decrease in total GFlops in vision models. For NLP models, also excluding the embedding layer, the reductions are about 93.5%p, and 84.2%p in total GFlops. Due to the unique structure of the XLNet model, where our method was not applied to the attention layers, the reduction in model parameters is 58.3%p, and GFlops decrease by 45.4%p. As the results indicate, the PC-LoRA method demonstrates a favorable trade-off on both CIFAR10 and IMDb benchmarks, by significantly reducing the GFlops and model parameters while only modestly compromising accuracy.

Additionally, as mentioned in the **Approach: PC-LoRA** section, an ablation study was conducted to assess the performance of training solely with the weights of the LoRA without weight decay scheduling. This study revealed that, in the absence of weight decay scheduling method, the models were unable to learn, resulting in negligible accuracy of 50% for IMDb (*which has two classes*), and 10% for CIFAR10 (*which has ten classes*). This underscores the critical role of weight decay scheduling in facilitating the learning process for our method.

5 Conclusion

PC-LoRA introduces a novel approach for simultaneous model compression and parameter-efficient fine-tuning. By deliberately decreasing and then recovering model weights with low-rank adaptations, PC-LoRA facilitates fine-tuning for new tasks while compressing the model. Our method effectively reduces computational costs without greatly affecting performance.

Future works We should focus on searching for a more **(1) optimized weight decay scheduler** that enhances compression performance. Additionally, we plan to incorporate a **(2) knowledge distillation(KD) loss term** (Hinton et al., 2015) into current loss function as a regularization term. Considering the pre-trained model's output as a teacher target, we can compute the KD loss term between this target and the LoRA's output, treating the latter as a student. Furthermore, we will compare our approach with other model compression approaches such as pruning (Liu et al., 2019b) and low-rank factorization (Hsu et al., 2022).

6 ACKNOWLEDGEMENT

This research was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS2022-00187238, Development of Large Korean Language Model Technology for Efficient Pre-training) and Brian Impact Foundation, a non-profit organization dedicated to the advancement of science and technology for all.

REFERENCES

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2005.14165. 1

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pp. 213–229, 2020. URL https://arxiv.org/abs/2005.12872. 1

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. URL https://arxiv.org/abs/2104.14294. 7

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020. URL https://arxiv.org/abs/2003.10555. 7

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314. 2

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4171–4186, 2019. URL https://arxiv.org/abs/1810.04805. 1, 7

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.11929. 1, 7

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. URL https://arxiv.org/abs/2111.06377. 7

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531. 2, 4

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, Jun 2019. URL http://arxiv.org/abs/1902.00751. 1

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization, 2022. URL https://arxiv.org/abs/2207.00112. 4

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9. 1

Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training cnns with low-rank filters for efficient image classification. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.06744. 2

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference, 2017. 2

Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. URL https://api.semanticscholar.org/CorpusID:18268744. 3

Vadim Lebedev, Yaroslav Ganin, Max Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6553. 2

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*, 2021. URL https://arxiv.org/abs/2104.08691. 1

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *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)*, abs/2101.00190, 2021. URL https://arxiv.org/abs/2101.00190. 1

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *ArXiv*, abs/2103.10385, 2021. URL https://arxiv.org/abs/2103.10385. 1

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019a. URL https://arxiv.org/abs/1907.11692. 7

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning, 2019b. URL https://arxiv.org/abs/1810.05270. 4

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983. 7

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101. 7

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015. 3

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020. 7

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. URL https://arxiv.org/abs/1412.6550. 2

Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6655–6659, 2013. URL https://ieeexplore.ieee.org/document/6638949. 2

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762. 1

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *International Conference on Computer Vision (ICCV)*, 2021. URL https://arxiv.org/abs/2102.12122. 1

Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating filters for faster deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 658–666, 2017. URL https://arxiv.org/abs/1703.09746. 2

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020. URL https://arxiv.org/abs/1906.08237. 7

A Training Details

A.1 Datasets

The IMDb dataset includes 50,000 movie reviews, evenly divided between training and testing sets. The CIFAR-10 dataset contains 60,000 color images in 10 classes, split into 50,000 for training and 10,000 for testing.

A.2 Settings for Training

For the optimization of the PC-LoRA method, we used AdamW (Loshchilov & Hutter, 2019) optimizer, exploring learning rates within 1e-2, 1e-5. In terms of learning rate scheduling, we adopted CosineAnnealingLR (Loshchilov & Hutter, 2017), which is set with a minimum value of 0. We used the batch size to be 64 for CIFAR-10 image classification task and 20 for IMDB text classification task. The image classification was trained for 62,500 iterations, and the text classification task, 100,000 iterations. Additionally, all experiments were conducted on RTX 5000 GPUs using PyTorch version 2.1, Python 3.10, and CUDA 11.8.0.

B Models Information

We employed base versions of various pre-trained models. We used Bert-base (Devlin et al., 2019), XLNet-base (Yang et al., 2020), RoBERTa-base (Liu et al., 2019a), and ELECTRA-base (Clark et al., 2020) for text classification task, which were sourced from the HuggingFace Transformers library. For image classification task , we used ViT-base (Dosovitskiy et al., 2021), ViT-base/DINO (Caron et al., 2021), Clip-ViT-base (Radford et al., 2021), ViT-base-MAE (He et al., 2021), and R50-ViT hybrid (Dosovitskiy et al., 2021), which incorporates the ResNet-50 architecture, sourced from the 'timm' library.

Download Links for pre-trained Models

1. ViT-base/DINO : https://huggingface.co/timm/vit_base_patch16_224.dino
2. ViT-base/Clip : https://huggingface.co/timm/vit_base_patch16_clip_224.openai
3. R50-ViT-base : https://huggingface.co/timm/vit_base_r50_s16_224.orig_in21k
4. ViT-base/MAE : https://huggingface.co/timm/vit_base_patch16_224.mae
5. Bert-base : https://huggingface.co/bert-base-uncased

6. XLNet-base : https://huggingface.co/xlnet/xlnet-base-cased
7. RoBERTa-base IMDB : https://huggingface.co/FacebookAI/roberta-base
8. ELECTRA-base: https://huggingface.co/google/electra-base-discriminator

DOWNLOAD LINKS FOR FINE-TUNED MODELS

1. Bert-base IMDB : https://huggingface.co/nikitakapitan/bert-base-uncased-finetuned-imdb
2. XLNet-base IMDB : https://huggingface.co/pig4431/IMDB_XLNET_5E
3. RoBERTa-base IMDB : https://huggingface.co/aychang/roberta-base-imdb
4. ELECTRA-base: https://huggingface.co/pig4431/IMDB_ELECTRA_5E
5. R50-ViT-base Cifar10 : https://console.cloud.google.com/storage/browser/vit_models/imagenet21k

For these three models, ViT-base/DINO, ViT-base/CLIP, and ViT-base/MAE, we have conducted fine-tuning on CIFAR-10 by searching for the optimal hyperparameters.