Prune As You Tune: LoRA-Enabled Model Compression

1st Joey Bejjani*, 2nd Amulya Garimella†, 3rd Swati Goel‡, 4th Mohammed Sakib§
John A. Paulson School of Engineering and Applied Sciences
Harvard University
*jbejjani@college.harvard.edu, †agarimella@college.harvard.edu,
‡sgoel@college.harvard.edu, §msakib@college.harvard.edu

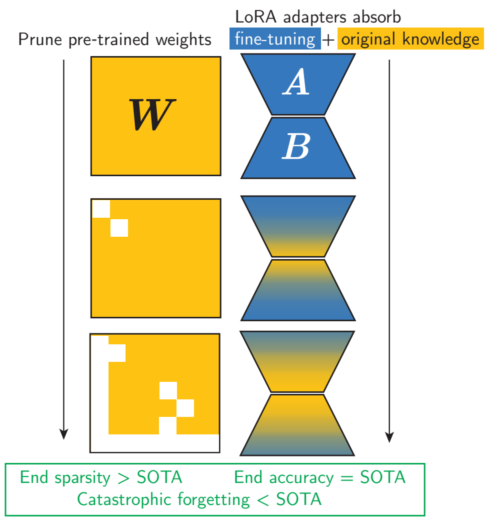

Prune pre-trained weights
LoRA adapters absorb
fine-tuning + original knowledge
W
A
B
End sparsity > SOTA
End accuracy = SOTA
Catastrophic forgetting < SOTA

Fig. 1. Graphical overview of our method. Prune As You Tune interleaves pruning of pre-trained parameters with LoRA fine-tuning updates guided by a custom KD loss function.

Abstract*—Parameter-Efficient Fine-Tuning (PEFT) techniques such as LoRA have gained significant attention in recent years due to their ability to reduce the computational cost of fine-tuning Large Language Models (LLMs). However, post-PEFT, inference remains expensive, as all model parameters are still required for the forward pass, in addition to parameters introduced with LoRA adapters. Moreover, the fine-tuning step typically leads to Catastrophic Forgetting (CF) of the original task. We propose Prune As You Tune (PAYT), a method that interleaves pruning of pre-trained parameters with LoRA updates to enable efficient fine-tuning while gradually compressing the final model. Crucially, PAYT mitigates CF through a custom Knowledge Distillation (KD) loss during fine-tuning. We show that PAYT can achieve 50% sparsity with minimal accuracy degradation and lower perplexity on the original task compared to baselines such as full fine-tuning. Compared to state-of-the-art approaches, PAYT offers superior accuracy, model compression, and retention of pre-training knowledge. Our code is avaialble at https://github.com/jbejjani2022/prune-as-you-tune.

***Index Terms*—Parameter Efficient Fine-Tuning, LLM Pruning, Knowledge Distillation, Catastrophic Forgetting.**

I. Introduction

Large Language Models (LLMs) excel in natural language processing and generation tasks but have substantial memory and compute requirements during training and inference. In particular, standard fine-tuning of pre-trained LLMs requires updating all model parameters, incurring significant costs. Recent work on Parameter Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA), has dramatically reduced the number of trainable parameters during fine-tuning while achieving competitive accuracy [1].

However, post-PEFT, forward passes of the fine-tuned model still require all pre-trained parameters, in addition to parameters introduced by LoRA, resulting in high memory and compute requirements. Model compression techniques aim to solve this problem. For instance, pruning methods remove a subset of network parameters, yielding a sparse model with decreased resource requirements [2] [3].

However, fine-tuning and pruning each pose the risk of Catastrophic Forgetting (CF) of the original task; fine-tuning optimizes for the new task potentially at the expense of pre-training knowledge, while during pruning, knowledge is discarded as parameters are removed. Retention of pre-trained knowledge is crucial for a model to perform well in multi-task learning or transfer learning settings.

Previous work has integrated model compression with PEFT [4] [5] [6] [7]. However, current methods in the literature suffer from significant accuracy degradation or do not address CF.

We present Prune As You Tune (PAYT), which achieves a dual objective of PEFT and model compression while mitigating CF. Figure 1 summarizes our method. PAYT offers the following key advantages:

- **Sparsity.** By spreading out pruning over fine-tuning epochs, PAYT achieves up to 50% unstructured sparsity of the pre-trained backbone with minimal accuracy degradation compared to standard fine-tuning approaches.
- **Retention of original knowledge.** Our use of a decaying pruning schedule and a custom KD loss function reduces CF of the original task.
- **More knowledge in fewer parameters.** PAYT's KD loss *encourages the LoRA adapters to absorb the knowledge of the pre-trained parameters as they are pruned, recovering error while simultaneously learning the fine-tuning task*. Our results indicate that this mechanism enables storing more knowledge in fewer parameters.
- **Parameter-efficiency.** PAYT employs rsLoRA to train only 4.6% of total parameters during fine-tuning.

II. Approach

Our method is comprised of the following techniques:

rsLoRA Fine-Tuning

We fine-tune with Rank-Stabilized LoRA (rsLoRA), which uses an appropriate scaling factor to improve learning compared to standard LoRA [8]. We use rank $r = 32$, scaling factor $\alpha_{\text{lora}} = 32$, and dropout $= 0.1$ for regularization. In initial experiments, we found that $r = 32$ yielded best results compared to $r = 8$ and $r = 256$, providing a balance between learning capacity and parameter-efficiency.

We inject adapters into the query, key, value, and output layers of all self-attention modules and the intermediate and output layers of feed-forward modules, following standard practice [1].

Magnitude-Based Pruning

Our approach globally prunes paramters with the lowest L1-norm. When followed by fine-tuning, such magnitude-based pruning methods have been shown to achieve high levels of sparsity without sacrificing accuracy [2] [3] [9]. Moreover, efficient systolic array implementations, such as systolic array packing [10], can take advantage of unstructured sparsity to save inference time.

Parameters eligible for pruning under PAYT are all frozen, pre-trained parameters, excluding embedding layers to preserve accuracy. Note that rsLoRA adapters are not pruned during PAYT to allow for adequate learning of the new task and distillation of the original task.

Pruning Scheduling

PAYT prunes the model before the start of each fine-tuning epoch by $s =$ an absolute percentage determined by a pruning schedule. We investigate (1) a linear schedule with constant s (2) a staggered linear schedule that prunes every other epoch (3) a decaying schedule performing Automated Gradual Pruning (AGP) [3] and (4) a delayed schedule performing AGP after the first 20% of epochs. We test PAYT over 5 and 10 fine-tuning epochs.

Knowledge Distillation Loss

We adopt a KD setup in which we apply PAYT to a 'student' model and keep a frozen copy of the original model as the teacher. We use the following loss function [4]:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{true}} + (1 - \alpha) \cdot \mathcal{L}_{\text{KD}} \tag{1}$$

where $\mathcal{L}_{\text{true}}$ is the cross-entropy loss of the student model on the fine-tuning task, $\mathcal{L}_{\text{KD}}$ is a KD loss term, and α is a hyperparameter controlling the relative weighting of the KD term. $\mathcal{L}_{\text{KD}}$ is computed as the Kullback-Leibler divergence of the softmax of the student and teacher logits y_{student} and y_{teacher} for a particular fine-tuning sample, divided by a temperature hyperparameter T:

$$\mathcal{L}_{\text{KD}} = T^2 \cdot \text{KL}(\log \text{sm}\left(\frac{y_{\text{student}}}{T}\right) \parallel \log \text{sm}\left(\frac{y_{\text{teacher}}}{T}\right)) \tag{2}$$

Data-Mixing

We attempted to mitigate CF without the inference overhead of KD loss by randomly shuffling samples from the pre-training dataset, labeled by the original model, into the fine-tuning train set and leaving the eval dataset unmixed. We augmented our dataset by 5% and 10% with samples from WikiText-2.

Evaluation Metrics

- **Accuracy on Fine-Tuning Task.** We report accuracy on the fine-tuning test set to evaluate how well our method learns the new task.
- **Final Model Sparsity.** We report the final sparsity of the model to assess the model compression objective.
- **Perplexity on Original Task.** We compute the model's perplexity (PPL) on the pre-training task to assess CF. An increase in PPL compared to the PPL of the original model indicates forgetting of the pre-training task.

III. Implementation

We fine-tune BERT [11] on the IMDb [12] and Stanford Sentiment Treebank (SST) [13] datasets for sentiment classification. We use the Hugging Face API to fine-tune with rsLoRA and the PyTorch prune package for progressive L1 Unstructured pruning. In our experiments, we use a batch size of 64 and fp16 mixed precision for training efficiency.

As BERT is pre-trained for masked language modeling, we follow the literature in calculating a pseudo-log-likelihood to measure perplexity by masking tokens one by one [14]. Due to resource constraints, we evaluate pseudo-perplexity on the WikiText-2 dataset as a proxy for BERT's prohibitively large pre-training set [15], following the approach in [16].

We evaluate PAYT against relevant baselines: full fine-tuning, rsLoRA fine-tuning, prune-then-fine-tune, and PAYT without KD loss. We also test different pruning schedules and determine optimal α and T with a hyperparameter search (Figure 2).

IV. Empirical Results

rsLoRA Fine-Tuning

rsLoRA fine-tuning helped mitigate forgetting as compared to full fine-tuning, with minimal accuracy degradation (Tables I, II). rsLoRA saw only marginal increase in perplexity as compared to the base model (with no fine-tuning or pruning), which achieved perplexity of 5.805 on the WikiText-2 dataset.

Magnitude-Based Pruning

Simple pruning generally degraded accuracy and increased perplexity and loss. For example, a pruned model at 50% sparsity with no fine-tuning yielded 68.535 perplexity. PAYT mitigated forgetting while learning the new task: our PAYT models at 50% sparsity achieved a perplexity of 21.524 on the same dataset (Table I).

Even after fine-tuning, simply pruned models showed an accuracy degradation of up to 0.02 (Table II). PAYT was able to recover and even **reverse accuracy degradation** in some

0% sparsity

Method	Eval Acc	Eval Loss	PPL
Full fine-tuning (FFT)	0.941	0.393	30.382
rsLoRA	0.939	0.280	7.906

Prune 50% then fine-tune

Method	Eval Acc	Eval Loss	PPL
FFT	0.933	0.396	83.981
rsLoRA	0.930	0.303	58.478

PAYT, rsLoRA, 50% sparsity

Method	Eval Acc	Eval Loss	PPL
Linear Schedule	0.931	0.231	38.987
Linear Schedule, KD	0.932	0.287	47.642
Delayed AGP (DAGP)	0.936	0.255	23.885
DAGP, Data-Mixing	0.933	0.254	24.626
DAGP, KD	0.936	0.221	21.524

TABLE I

Task: Binary sentiment classification on IMDb. Comparison of pure fine-tuning methods, typical prune-then-fine-tune methods, and PAYT. PPL is computed on WikiText-2 after 10 epochs of fine-tuning. For KD, we used $\alpha = 0.4$ and $T = 4$, optimal parameters determined by a grid search.

cases, improving accuracy by 0.006 points over the second best model in Table II.

Pruning Scheduling

Of the schedules tested, PAYT with 20% delayed AGP had the highest accuracy and lowest CF (Figure 2) (Table I). We **improve accuracy, perplexity, and evaluation loss over both prune-then-fine-tune methods, and achieve accuracy only marginally lower than a model with no pruning** (0.003 difference from standard rsLoRA).

We hypothesize that this schedule recovers accuracy and mitigates forgetting by allowing gradual distillation of the original model into low-rank adapters without shocking the network. Indeed, Figure 2 illustrates accuracy recovery after the first few pruning steps under the AGP regime.

Knowledge Distillation Loss

KD loss decreased perplexity and loss without degrading accuracy (Table I), indicating that the teacher's predictions for the new task give the student knowledge about the original task [17].

Data-Mixing

Data-mixing decreased CF only marginally compared to KD, likely because assigning hard labels to samples did not expose the pruned model to 'dark knowledge' in the teacher model [18].

V. Analysis of Related Works

Previous works integrate model compression with PEFT but degrade accuracy, achieve low sparsity, or do not mitigate forgetting of the original task.

Method	Eval Acc	Eval Loss	PPL
FFT BERT	0.532	1.476	17.08
rsLoRA BERT	0.530	1.078	7.823
Prune 50% then FFT	0.533	1.160	176.529
Prune 50% then rsLoRA	0.514	1.100	40.258
PAYT - 50% sparsity	0.539	0.579	32.521

TABLE II

Task: Fine-grained (5 class) sentiment classification on SST. Experiment design same as Table I.

PC-LoRA decays all pre-trained weights to zero during fine-tuning, leaving only the LoRA adapters. PC-LoRA leads to 3.6% accuracy degradation on the new task and likely induces CF of the original task [4]. KD-LoRA incurs high costs by fully fine-tuning a teacher model and then distilling to a smaller student model, and does not measure CF [5]. Pruning-Aware Tuning is resource-efficient, but only generates 25% sparsity during fine-tuning [6]. LoRAPrune generates 50% sparsity, but does not leverage techniques for reducing CF, such as KD and data-mixing [7].

PAYT improves on these methods by achieving sparsity as high as 50% while retaining accuracy and mitigating CF.

VI. Conclusion & Future Work

Our method integrates Parameter-Efficient Fine-Tuning and progressive pruning, leveraging a custom KD loss to encourage the adapted model to absorb knowledge of the original task while learning the new task. The result is a more compact, resource-efficient model that performs comparably to fully fine-tuned counterparts on new tasks and preserves pre-trained knowledge for broader multi-task applications.

We seek to build upon the following research directions in future work:

- **Structured pruning.** We implemented L2 Structured pruning and saw a decrease in accuracy. In future work, we hope to recover error induced by structured sparsity and compare efficiency gains across pruning methods.
- **LoRA initialization.** Following [16], we implemented a LoRA initialization optimized for KD, and hope to integrate with PAYT to continue improving the best-case accuracy/perplexity trade-off.

Acknowledgments and Contribution Statement

JB, AG, SG, MS thank Professor H.T. Kung, Andrew Sabot, Henry Bae, Taj Gulati, and Pavan Pandurangi for their advice and support in producing this work.

Modules contributed by each author below. All collaborated on the codebase and report.

- JB: Core experiments, rsLoRA configuration, progressive pruning, perplexity evaluation, hyperparameter search
- AG: Pruning scheduling, data-mixing, testing infrastructure, hyperparameter search
- SG: KD loss, LoRA configuration, hyperparameter search
- MS: Perplexity evaluation, KD loss, testing

References

[1] 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.

[2] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks.

[3] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression.

[4] Injoon Hwang, Haewon Park, Youngwan Lee, Jooyoung Yang, and SunJae Maeng. PC-LoRA: Low-rank adaptation for progressive model compression with knowledge distillation.

[5] Rambod Azimi, Rishav Rishav, Marek Teichmann, and Samira Ebrahimi Kahou. KD-LoRA: A hybrid approach to efficient fine-tuning with LoRA and knowledge distillation.

[6] Yijiang Liu, Huanrui Yang, Youxin Chen, Rongyu Zhang, Miao Wang, Yuan Du, and Li Du. PAT: Pruning-aware tuning for large language models.

[7] Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. LoRAPrune: Structured pruning meets low-rank parameter-efficient fine-tuning.

[8] Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with LoRA.

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks.

[10] H.T. Kung, Bradley McDaniel, and Sai Qian Zhang. Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding.

[12] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics.

[13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 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. Association for Computational Linguistics.

[14] Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712.

[15] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. _eprint: 1609.07843.

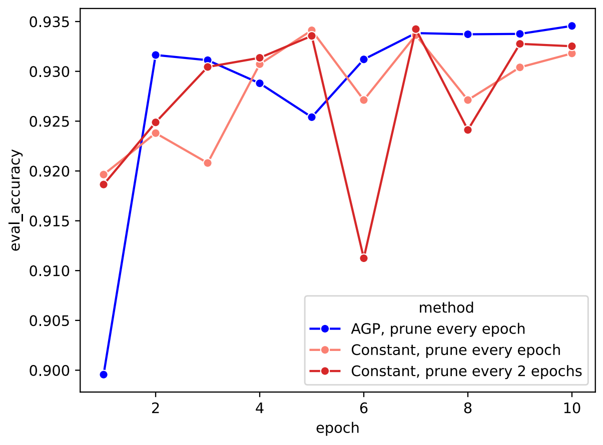
[16] Muhammad Fawi. CURLoRA: Stable LLM continual fine-tuning and catastrophic forgetting mitigation.

[17] Zhizhong Li and Derek Hoiem. Learning without forgetting.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network.

Appendices

VII. Appendix 1



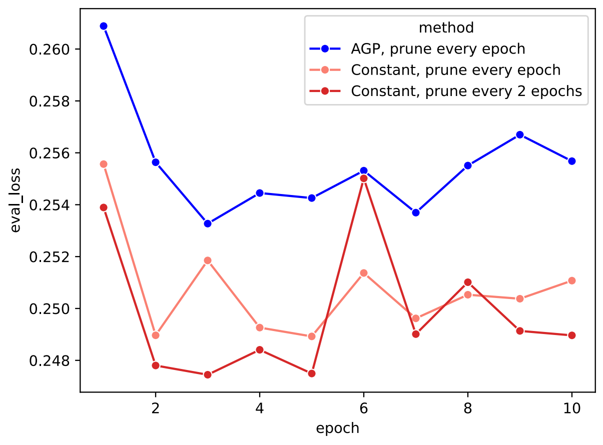


Fig. 2. Evaluation accuracy (top) and loss (bottom) over 10 fine-tuning epochs for Automated Gradual Pruning (with no delay) vs. Linear schedules. Task: Binary sentiment classification on IMDb. Note: 'Constant' refers to the constant sparsity increases implemented by the Linear schedule.

Temp	Alpha	Eval Acc	Eval Loss	PPL
2	0.8	0.932	0.282	29.3
2	0.5	0.93	0.248	27.39
2	0.2	0.5	0.168	22.79
4	0.8	0.929	0.283	28.5
4	0.5	0.929	0.248	25.11
4	0.2	0.5	0.17	21.56

TABLE III

KD loss hyperparameter search, with IMDb, linear pruning schedule, 50% sparsity, 5 epochs. We decided that $T = 4$, $\alpha = 0.5$ gave the best accuracy/PPL balance. After conducting additional experiments, we decreased to $\alpha = 0.4$, seeing further reductions in PPL without accuracy degradation.