StopAndGrow: Accelerating Large Language Model Pre-training by Growing Intermediate Checkpoints

Vibha Masti [*1] **Xinyue Liu** [*1] **Ananya Jha** [2 3] **Emma Strubell** [1 3]

Abstract

As the demand for larger and more capable language models grows, the computational cost of pre-training becomes prohibitively expensive. Efficient pre-training techniques are crucial for making LLM training more accessible, particularly for researchers with limited computational resources. We introduce StopAndGrow, a method for efficient pre-training where a model is partially trained at a smaller size (e.g., Pythia-70M) before being grown in a function-preserving manner to a compatible larger size (e.g., Pythia-410M) to continue training. Our approach reduces total GPU hours while maintaining the same training token exposure. We compare full model fine-tuning, LoRA, and ReLoRA approaches after model growth across different growth timing strategies, finding that parameter-efficient methods, particularly ReLoRA, offer the best performance-efficiency trade-off. To facilitate reproducibility, we release our implementation as open-source code with accompanying Jupyter notebooks.

1. Introduction

Pre-training large Transformer-based (Vaswani et al., 2017) models requires substantial computational resources, often involving thousands of GPU-hours and enormous amounts of data. The recent trend toward developing and releasing model families of various sizes (Meta, 2024; Team et al., 2023; Biderman et al., 2023) has highlighted the correlation between parameter count and model capability, with larger models consistently demonstrating superior performance (Brown et al., 2020; Touvron et al., 2023; Dosovitskiy et al., 2020; Meta, 2024). However, the standard practice of training each model variant from scratch with randomly initialized parameters (Radford et al., 2019; Devlin et al., 2019) introduces significant inefficiencies in the development pipeline, resulting in excessive energy consumption, carbon emissions, and financial costs (Strubell et al., 2019).

The machine learning community has investigated various approaches to improve model efficiency. Post-training optimization techniques such as quantization (Liu et al., 2023; Hubara et al., 2016), pruning (Frankle and Carbin, 2018; Frankle et al., 2020), and knowledge distillation (Hinton et al., 2015; Guo et al., 2025) have successfully reduced memory requirements and computational costs at inference time. Complementary to these approaches, model expansion techniques (Wang et al., 2023; Chen et al., 2021; 2015; Samragh et al., 2024; Shen et al., 2022) aim to transform smaller pre-trained models into larger ones using function-preserving projections. However, existing expansion methods typically require approximately twice the training tokens compared to training from scratch (Samragh et al., 2024) and have primarily been demonstrated using high-end computational resources that remain inaccessible to many researchers. The efficiency of these approaches in computationally constrained environments has received limited attention.

To address these challenges, we introduce StopAndGrow (Figure 1), a technique that leverages partially trained smaller models as starting checkpoints for larger models through function-preserving transformations. Using modest compute (8 A6000 GPUs, 48-hour budget), our experiments with models under 1B parameters demonstrate significant efficiency gains while preserving model quality. We systematically compare full fine-tuning, LoRA (Hu et al., 2021), and ReLoRA (Lialin et al., 2023) after growth, finding that ReLoRA consistently delivers the best performance-efficiency trade-off. These results establish a foundation for more accessible LLM pre-training in resource-constrained environments.

[*]Equal contribution [1]Language Technologies Institute, Carnegie Mellon University [2]University of Washington [3]Allen Institute for AI. Correspondence to: Vibha Masti <vmasti@alumni.cmu.edu>, Xinyue Liu <xinyuel4@andrew.cmu.edu>.

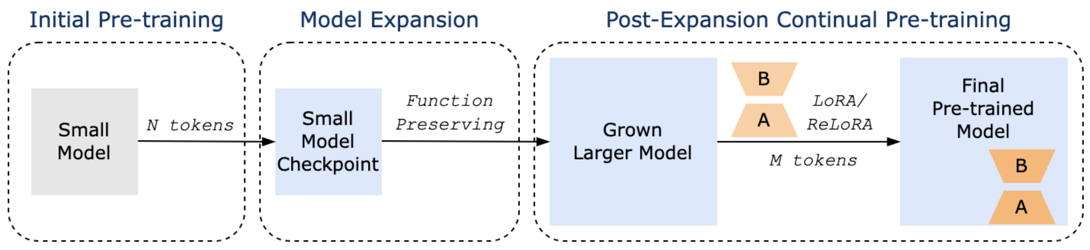

Figure 1. Overview of StopAndGrow pipeline. The process involves (left) pre-training a small model on N tokens, (middle) applying function-preserving expansion to increase model size, and (right) continuing pre-training on M additional tokens using either full model fine-tuning or parameter-efficient methods like LoRA/ReLoRA. This approach reduces computational cost while maintaining the same total training tokens ($N + M$).

2. Related Work

Function-preserving model growth techniques enable knowledge transfer from smaller to larger models while maintaining identical network behavior. Net2Net (Chen et al., 2015) pioneered this approach by introducing network widening and deepening operations that preserve functional equivalence during expansion. Building on this foundation, bert2BERT (Chen et al., 2022) adapted these techniques to BERT (Kenton and Toutanova, 2019) models, introducing advanced knowledge initialization (AKI) that leverages adjacent layer weights to initialize grown models by combining upper layer rows with sampled lower layer rows.

Recent advances have introduced more sophisticated expansion methods. Mango (Pan et al., 2024a) establishes linear correlations between pre-trained and target model weights. LEMON (Wang et al., 2023) develops vector and matrix-level lossless expansion operators. Apollo (Pan et al., 2024b) prepares information about future higher layers during lower layer training. LiGO (Wang et al., 2022) constructs grown matrices by interleaving rows and columns with their linear combinations. Shen et al. (2022) transfer both weights and optimizer states to maintain training dynamics during expansion. HyperCloning (Samragh et al., 2024) applies function-preserving expansion to fully pre-trained models before continuing training on the entire dataset.

While prior research has established various model expansion techniques, most studies focus primarily on the technical mechanisms of growth rather than computational efficiency. Our work shifts this focus by investigating the parameter and cost efficiency of using grown models as starting checkpoints. We quantitatively measure GPU hour reductions when growing from smaller checkpoints compared to training from scratch, while maintaining constant total token exposure. Our analysis of growth timing relative to total training provides insights into optimal expansion points for maximizing efficiency.

Our approach most closely relates to Du et al., who investigated optimal pre-training strategies using growth operators at different training stages under fixed computational budgets. While they focused on full model fine-tuning after expansion, we explore combining model growth with parameter-efficient fine-tuning methods. Our key contribution is extending their framework to evaluate whether LoRA (Hu et al., 2021) and ReLoRA (Lialin et al., 2023) can maintain model quality while further reducing computational requirements—a particularly valuable investigation for resource-constrained research environments.

3. StopAndGrow

We introduce StopAndGrow, a method for efficient language model pre-training that follows a three-phase approach. First, we partially pre-train a smaller model with parameters θ_{small} on N tokens. Next, we apply a function-preserving transformation (Figure 2) to expand the model to a larger size with parameters θ_{grown_small}. Finally, we continue pre-training the expanded model on M additional tokens, either by updating all parameters or using parameter-efficient fine-tuning methods.

The function-preserving property is central to our approach, ensuring that the expanded model initially produces outputs identical to the smaller model for any input: $\forall x, f(x; \theta_{small}) = g(x; \theta_{grown\_small})$, where f and g represent the smaller and grown models respectively. This property provides an optimal initialization point for continued training, eliminating the performance drop typically associated with random initialization of larger models.

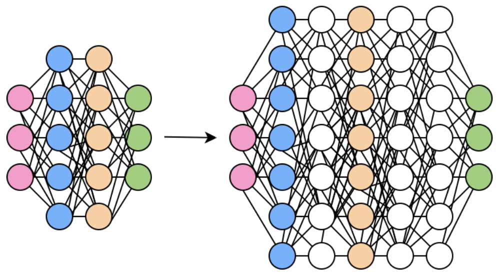

Figure 2. Function-preserving model growth encompasses both width expansion (increasing hidden dimensions) and depth expansion (increasing number of layers), similar to approaches in Shen et al. (2022); Du et al..

We systematically evaluate StopAndGrow across various configurations by adjusting the token allocation between initial pre-training (N) and continued training (M) while keeping their sum constant ($N + M$). This allows us to identify optimal growth timing and quantify efficiency gains compared to training the larger model from scratch with parameters θ_{large}. We also compare the effectiveness of different parameter-efficient fine-tuning methods applied after model growth to determine which approach offers the best performance-efficiency trade-off.

4. Results

For our experiments, we evaluated models with parameters under 1 billion using a cluster of 8 A6000 GPUs with a 48-hour training time limit. We additionally validated our findings by reproducing key experiments on a single A100 GPU on Google Colab [1]. We selected the Pythia family of models and the Pile (Gao et al., 2020) pre-training corpus for two primary advantages: the Pythia family includes several models in the $\leq$ 1B parameter range and provides the exact pre-training data ordering used in the original Pile training. While the entire dataset contains 300B tokens, our computational constraints limited us to using only the last n tokens from the dataset, ranging from 1B to 4B in steps of 1B tokens, while maintaining the original token order. We conducted experiments to partially pre-train a small model (Pythia-70M) with $(100 - x)\%$ tokens, grow the model using the $grow()$ operator, and then continue training the larger model (Pythia-410M) with $x\%$ tokens. Due to computational constraints, we evaluated configurations with $x\%$ corresponding to 1 billion, 2 billion, and 4 billion tokens.

To evaluate performance, we measured perplexity (Jelinek et al., 1977) on LAMBADA (Paperno et al., 2016) as pre-processed by OpenAI (EleutherAI, 2024) and the Paloma (Magnusson et al., 2023) benchmarks. To quantify computational requirements, we calculated total pre-training FLOPs as shown in Equation 1, where $FLOPs$ represents the floating point operations, T is the number of tokens processed, and L is the sequence length of each chunk. Our calculations include both forward and backward pass FLOPs, with the approximation that backward pass FLOPs typically require 2× the computation of forward pass FLOPs (DeepSpeed, 2025). The performance results are presented in Table 1 and the computational savings in Table 2. We also compare the compute costs and perplexity scores between full model pre-training, LoRA, and ReLoRA in Figure 3.

$$\text{Compute cost} = \frac{FLOPs \times T}{L} \tag{1}$$

5. Future Work

We introduced StopAndGrow, an efficient pre-training approach that grows smaller models to larger sizes using function-preserving transformations and parameter-efficient fine-tuning. Our experiments with limited compute (8 A6000 GPUs, 48-hour training) on models under 1B parameters show promising efficiency gains. Future research should extend this work to larger model scales and earlier growth checkpoints to better understand the efficiency-performance trade-offs across different computational regimes.

[1] https://github.com/EfficientLLMs/lazy-pretrain/blob/main/Efficient_Pre_training.ipynb

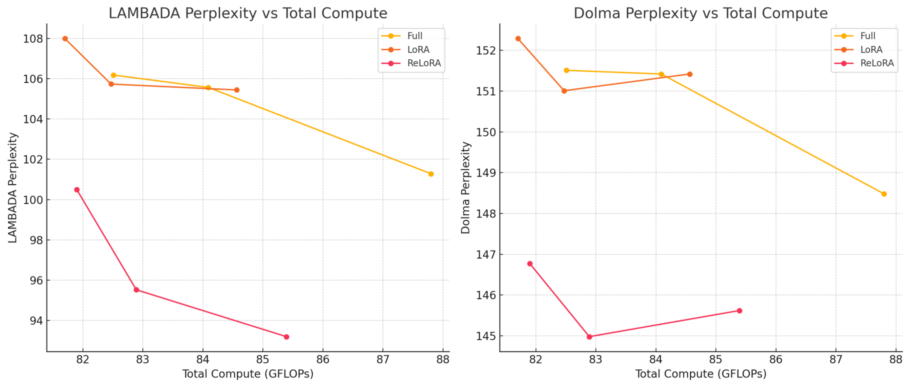

Figure 3. Perplexity on the LAMBADA (left) and Dolma (right) datasets as a function of total compute (FLOPs). ReLoRA consistently outperforms both Full pretraining and LoRA in terms of perplexity while requiring similar or less total compute.

Table 1. Perplexity on LAMBADA and Dolma datasets across different model growth checkpoints, adaptation methods, and training budgets. Lower is better (↓).

Base Checkpoint	**Target Modules**	**# Tokens (B)**	**LAMBADA PPL (↓)**	**Dolma PPL (↓)**
Baseline Pythia-70m	–	–	130.96	162.57
	Full Pretraining	1	106.18	151.51
step143000	LoRA	1	107.99	152.29
	ReLoRA	1	100.49	146.77
	Full Pretraining	2	105.56	151.42
step142000	LoRA	2	105.73	151.01
	ReLoRA	2	95.53	144.98
	Full Pretraining	4	101.28	148.48
step141000	LoRA	4	105.44	151.42
	ReLoRA	4	93.20	145.62

Table 2. Total compute cost (in GFLOPs) and savings across different pretraining strategies. Compute savings are relative to training the Pythia-410M model from scratch.

Checkpoint	**Method**	**Pretrain Comp. (GFLOPs)**	**Pretrain Tokens (B)**	**Continue Comp. (GFLOPs)**	**Continue Tokens (B)**	**Total Compute (↓) (GFLOPs)**	**Compute Saved (↑) (%)**
Pythia-410M	–	1086.66	300	0.00	0	636.71	–
Pythia-70M	–	137.19	300	0.00	0	80.38	87.38%
	Full Model	137.19	300	1086.66	1	82.51	87.04%
Step 143k	ReLoRA	137.19	300	777.31	1	81.90	87.14%
	LoRA	137.19	300	671.52	1	81.70	87.17%
	Full Model	137.19	298	1086.66	2	84.09	86.79%
Step 142k	ReLoRA	137.19	298	777.31	2	82.89	86.98%
	LoRA	137.19	298	671.52	2	82.47	87.05%
	Full Model	137.19	296	1086.66	4	87.80	86.21%
Step 141k	ReLoRA	137.19	296	777.31	4	85.39	86.59%
	LoRA	137.19	296	671.52	4	84.56	86.72%

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Meta. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2024. 04.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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*, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

Bin Liu, Fengfu Li, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yite Wang, Jiahao Su, Hanlin Lu, Cong Xie, Tianyi Liu, Jianbo Yuan, Haibin Lin, Ruoyu Sun, and Hongxia Yang. Lemon: Lossless model expansion. In *The Twelfth International Conference on Learning Representations*, 2023.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. bert2bert: Towards reusable pretrained language models. *arXiv preprint arXiv:2110.07143*, 2021.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Mohammad Samragh, Iman Mirzadeh, Keivan Alizadeh Vahid, Fartash Faghri, Minsik Cho, Moin Nabi, Devang Naik, and Mehrdad Farajtabar. Scaling smart: Accelerating large language model pre-training with small model initialization. *arXiv preprint arXiv:2409.12903*, 2024.

Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. Staged training for transformer language models. In *International Conference on Machine Learning*, pages 19893–19908. PMLR, 2022.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2023.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. bert2bert: Towards reusable pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2134–2148, 2022.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.

Yu Pan, Ye Yuan, Yichun Yin, Zenglin Xu, Lifeng Shang, Xin Jiang, and Qun Liu. Reusing pretrained models by multi-linear operators for efficient training. *Advances in Neural Information Processing Systems*, 36, 2024a.

Yu Pan, Ye Yuan, Yichun Yin, Jiaxin Shi, Zenglin Xu, Ming Zhang, Lifeng Shang, Xin Jiang, and Qun Liu. Preparing lessons for progressive training on language models. *arXiv preprint arXiv:2401.09192*, 2024b.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. In *The Eleventh International Conference on Learning Representations*, 2022.

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient llm pre-training. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 1977.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, 2016.

EleutherAI. Lambada openai dataset. https://huggingface.co/datasets/EleutherAI/lambada_openai, 2024. 07.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Pete Walsh, Yanai Elazar, Kyle Lo, et al. Paloma: A benchmark for evaluating language model fit. *arXiv preprint arXiv:2312.10523*, 2023.

DeepSpeed. Flops profiler, 2025. URL https://www.deepspeed.ai/tutorials/flops-profiler/#flops-measurement.