BUS (Bootstrapped Model UpScaling): An Efficient and Continuous Method of Large Language Model Growth

Brian Naklycky

University of Texas at Austin

December 3, 2024

Abstract

This paper introduces BUS (Bootstrapped Model UpScaling), a novel method combining the efficiency gains of the normalized GPT (nGPT) (Loshchilov *et al.*, 2024) architecture with a fractional scaling technique called BUS (Bootstrapped model UpScaling). Our research demonstrates that the nGPT architecture provides significant improvements in training efficiency, with 4-20x faster convergence compared to traditional methods, fractional scaling, inspired by and improving upon HyperCloning (Samragh *et al.*, 2024), enables more gradual and flexible model growth, and the introduction of the κ parameter allows for tuning the optimal point at which to scale the model.

We present experimental results on a 64M parameter model, showcasing the potential of this approach for developing LLMs that are not only larger but also more efficient and robust. Our findings indicate that training a model to maximum performance is not a prerequisite for successful scale-up. Instead, we demonstrate an incremental scaling process that maintains model efficacy throughout expansion.

This research opens new avenues for efficient scaling of both model width and depth, potentially revolutionizing the development of larger language models. By enabling more efficient use of computational resources and smoother transitions between model sizes, BUS presents a promising direction for the future of LLM development

1 Introduction

Modern Large Language Models (LLMs) have evolved from the original transformer architecture introduced by (Vaswani *et al.*, 2023). State-of-the-art models now primarily utilize a decoder framework trained on next token prediction. These models typically undergo extensive and very expensive pre-training to ensure better initializations for larger architectures ((Brown *et al.*, 2020), (Grattafiori *et al.*, 2024),(Cottier *et al.*, 2024)).

The prevailing approach suggests training models on approximately 20 tokens for every parameter. Additionally, models are being trained on increasingly longer context lengths, despite the need for higher quality data to adequately fill these extended contexts. Evidence suggests this may be inefficient, as models often perform significantly worse when tested on longer context lengths, indicating inadequate quality of long-context data in training corpora ((An *et al.*, 2024), (Gao *et al.*, 2024), (Kaplan *et al.*, 2020), (Rae *et al.*, 2022), (Hoffmann *et al.*, 2022)).

Research on efficient training has revealed that some hyperparameters, such as context length, which greatly increase training complexity, need not be as large as previously thought. Furthermore, new architectures have been developed to provide faster convergence.

A fundamental issue arises from the fact that Transformer models of various sizes are all being trained on essentially the same problem: accurately predicting the next token in a sequence. This leads to models of different sizes learning low-dimensional projections of the same problem space, as evidenced by their shared spectral decomposition (Wang *et al.*, 2020) and the rapid training speed-up observed in the normalized GPT (nGPT) architecture (Loshchilov *et al.*, 2024).

While larger models consistently demonstrate better performance, likely due to the lottery ticket hypothesis, where they are more likely to contain stronger and more robust subnetworks with better initializations, the current training process remains inefficient. It typically involves a

Standard Transformer	nGPT
$x_a \leftarrow ATTN(LayerNorm(x))$	$x_a \leftarrow Norm(Attn(x))$
$x \leftarrow x + x_a$	$x \leftarrow Norm(x + \alpha_a(x_a - x)$
$x_b \leftarrow MLP(LayerNorm(x))$	$x_b \leftarrow Norm(MLP(x))$
$x_b \leftarrow x + x_b$	$x \leftarrow Norm(x + \alpha_b(x_b - x))$
$Final : x \leftarrow LayerNorm(x_b)$	-

Table I: Transformer vs. Normalized Transformer

lengthy warmup stage, large batch sizes, and extended context lengths, often exceeding the effective context length observed during testing (Frankle and Carbin, 2019).

Recent advancements have focused on modifying model architectures to accelerate learning. The Roformer architecture, for instance, provides a more effective method for inserting positional information into the self-attention process. Similarly, the nGPT architecture has led to significant speedups in training.

Concurrent with our research, methods for scaling smaller models to initialize larger ones have been developed, such as HyperCloning (Samragh *et al.*, 2024). This technique involves doubling the size of all matrices, demonstrating substantial downstream model improvements. However, it lacks the ability to scale continuously. Exisiting literature on the topic start from simple methods ((Du *et al.*, 2024), (Chen *et al.*, 2016), (Yang *et al.*, 2020), (Samragh *et al.*, 2023)).

Our research builds upon these findings, introducing a smoother scaling technique that allows models to maintain learning efficiency while taking advantage of increased capacity to learn more complex language patterns. This approach aims to optimize the trade-off between knowledge transfer from smaller models and the benefits of larger architectures, potentially revolutionizing the development of more efficient and capable LLMs.

2 Methodology

Our approach combines three key elements: the nGPT architecture, a modified HyperCloning technique, and our novel scaling method called BUS (Bootstrapped model UpScaling). The main idea behind this scaling procedure is to optimize the training process by focusing on the most efficient learning phase, typically the first 20% of training tokens that account for approximately 90% of the total loss improvements.

2.1 nGPT Architecture

The nGPT architecture normalizes the weight matrices of the transformer, replacing existing normalization layers with standard vector normalization across model dimensions. This approach supports representational learning on the hypersphere and improves implicit gradient descent as a meta-optimization (Loshchilov *et al.*, 2024).

The key benefits of the nGPT architecture include significant efficiency gains, up to 4-20x faster convergence to model capacity, and the ability to directly train without warm up or weight decay. The details of the modified self-attention mechanisn in nGPT compared to the standard transformer is shown in Table I.

While this modification adds some computational overhead, the overall wall-time learning rate is faster. The authors suggest that this overhead can be mitigated with the creation of fused kernels for these operations.

2.2 HyperCloning

HyperCloning is an efficient method for model growth that expands the model's width. It provides better initialization for larger models, resulting in improved downstream performance and reduced risk of training failures due to poor initialization, loss divergence, or suboptimal hyperparameter tuning (Samragh *et al.*, 2024) (Grattafiori *et al.*, 2024).

$$W_L = \begin{bmatrix} W_S/2 & W_S/2 \\ W_S/2 & W_S/2 \end{bmatrix} \tag{1}$$

This approach preserves model performance during scaling, as the output logits remain consistent between models. HyperCloning has demonstrated a 2-4x speedup in training and improved final model performance compared to standard random weight initializations.

2.3 BUS (Bootstrapped Model UpScaling)

Unlike the original HyperCloning method, which doubles the model size, our BUS approach allows for incremental scaling. This enables a more gradual increase in model size, potentially leading to better resource utilization and smoother performance improvements. This approach introduces a new hyperparameter, κ, which determines the optimal point for scaling based on the model's predicted performance. The scaling target is calculated as follows:

$$t = tp * \kappa + m \tag{2}$$

Where t = Scaling Target loss, tp = total loss performance gains, κ = Scaling ratio, m = predicted final model loss

This formula allows for a more flexible scaling approach, optimizing the trade-off between extracting maximum performance from smaller models and leveraging the benefits of larger architectures, and can help balance between the amount of knowledge to upscale.

The upscaling process itself can be viewed as a fractional version of HyperScaling. Where the existing matrix is scaled according to what was defined in HyperScaling but then truncated to fit the new dimensions.

$$W_S \in \mathbb{R}^{d_{old} \times d_{old}}, W_L \in \mathbb{R}^{d_{new} \times d_{new}} \tag{3}$$

$$W_L = \frac{d_{old}}{d_{new}} \begin{bmatrix} W_S & W_{S[:,:d_{new}-d_{old}]} \\ W_{S[:d_{new}-d_{old},:]} & W_{S[:d_{new}-d_{old},:d_{new}-d_{old}]} \end{bmatrix} \tag{4}$$

Where d_{old} is the dimensions of the old model and d_{new} is the new dimensions. You can see that if we set $d_{new} = 2d_{old}$ we get exactly HyperScaling. But, instead of doubling the size of the matrix, if we start with a smaller one we can follow the same procedure as HyperScaling but simply cut the matrix to the desired dimensions.

With this rough cutting comes a loss of immediate performance that is only present in the full HyperScaling case which preserves logit outputs. We argue that there is still benefit from doing this kind of partial scaling. The reasoning why is that a completely random scaling yields effectively too much noise for the model to learn through. While a partial HyperScaling does preserve some logit outputs, allowing it to train not fully from scratch and potentially helping the model break out of local minima.

2.4 Implementation Details

Due to computational limitations, we used a small model with 64M parameters and a 30k token vocabulary using the BERT tokenizer (Devlin *et al.*, 2019). The architecture features a Decoder model design, Rotary Positional Encoding (Su *et al.*, 2023), and the nGPT framework modifications (Loshchilov *et al.*, 2024).

Extensive use of the GPT-Neo github library was used (Black *et al.*, 2021) and nanoGPT (Karpathy, 2023) to help generate the code for the model. Use of FlashAttention was used as well to speed up training (Dao, 2023).

For our experiments, we utilized the OpenWebText dataset (Gokaslan and Cohen, 2019). This dataset, which is a recreation of the WebText corpus used to train GPT-2, provides a diverse range of web content and is widely used in language model training. The OpenWebText dataset was chosen for its broad coverage of topics and its similarity to the data used in training larger language models, making it suitable for our scaling experiments.

Additionally, hyperparameters were chosen according to (Izsak *et al.*, 2021), who showed that you can save on compute and get comparable performance with BERT sized models at a fraction of the compute used by using smaller context lengths and higher learning rates. The work of (Gao *et al.*, 2024) Who also showed that training on a shorter context earlier in training can save on compute before scaling context length up later in training. Recommendations from [(Kaplan *et al.*, 2020), (Hoffmann *et al.*, 2022)] were also used to ensure sample efficiency by encouraging use of larger models and larger batch sizes.

We selected the Natural Language Inference (NLI) task for benchmarking, analyzing performance on both the SNLI(Glockner *et al.*, 2018) and MNLI (Williams *et al.*, 2018)datasets. This task was chosen due to its relative simplicity and the fact that models of this parameter scale typically struggle with more complex benchmarks such as Hellaswag (Zellers *et al.*, 2019), OpenBookQA (Mihaylov *et al.*, 2018), ARC (Clark *et al.*, 2018), and Winograd (Sakaguchi *et al.*, 2019).

	SNLI	MNLI[1]
Standard	40.4	###
BUS	**41.6**	38.6
HyperScaling	40.9	###

Table II: Model Accuracy on NLI task

1

3 Results

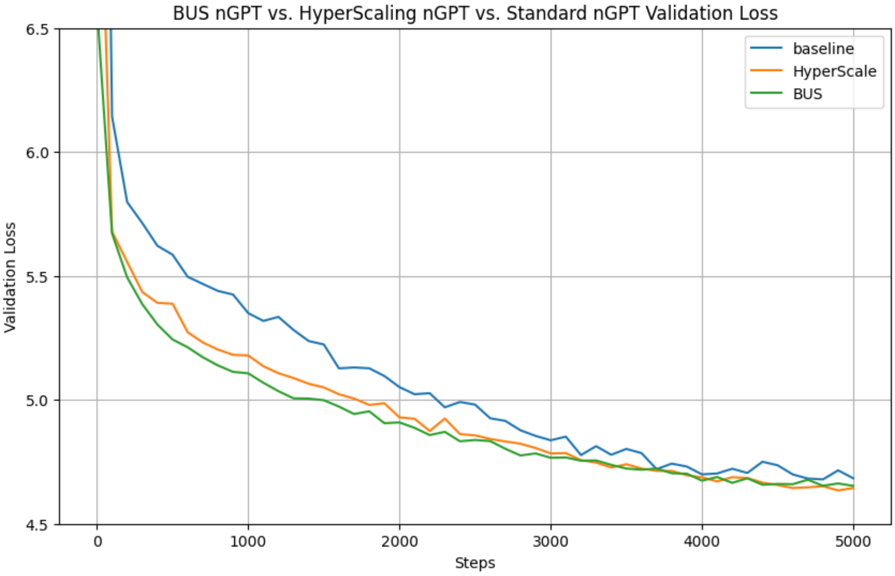

Figure 1: Validation loss across the baseline, BUS, and HyperScale models

We conducted a comparative analysis of three training methods to evaluate their effectiveness. A baseline 64M parameter nGPT model wihtout scaling, trained until loss plateaued. A HyperScaling model following (Samragh *et al.*, 2024). And the BUS model, trained with two partial transfer steps of the same total training steps as in the HyperScaling model. Due to compute restraints, only the LM head was trained during fine-tuning for all models.

The training performance of the BUS model demonstrated a slight advantage over HyperScaling. Both meethods enabled the models to reach their capacity, with BUS achieving this milestone marginally faster. Notably, both BUS and HyperScaling models outperformed the standard model in terms of convergence speed and final performance, as illustrated in Figure 1

In terms of downstream task performance, the BUS framework exhibited superior performance compared to both HyperScaling and the baseline model. Table II summarizes these results, showing that BUS outperformed HyperScaling by 0.7 points and the base model by 1.2 points on our benchmark tasks.

3.1 Discussion

The BUS framework has demonstrated promising results, outperforming both standard and HyperScaled models on downstream tasks while exhibiting a faster learning rate during pre-training. This suggests that the BUS expansion algorithm offers a slight advantage over HyperScaling, or at the very least, performs comparably. The superior performance of BUS over HyperScaling implies that these results may extend to larger models as well. However, further research across different model scales is necessary to confirm this hypothesis. These findings indicate that the BUS method presents a promising approach for efficient and effective model scaling in language model development.

[1] *Results are still being run*

Further investigation is needed to fully understand the impact of different κ values on final model performance. Additionally, experimenting with various selection methods for rows and columns to be placed in the expanded part of W_L from W_S could potentially yield improved performance with BUS. While we suspect this approach may lead to better results, more data and research are required to substantiate this claim. These areas of exploration could provide valuable insights into optimizing the BUS method and further enhancing its effectiveness in language model scaling.

We speculate that the improved performance of the BUS architecture stems from each upscaling step allowing the model to learn parameter weights that apply more generally to the problem of next token prediction. This contrasts with standard transformers or single upscaling steps like in HyperScaling. Evidence from the Linformer and nGPT papers supports this hypothesis, showing that models of different sizes share eigenvalues (Wang *et al.*, 2020)and that transformers are representational learners (Loshchilov *et al.*, 2024) . However, more investigation is needed to confirm this reasoning.

4 Conclusion

Our research introduces a novel approach to scaling language models, combining the efficiency of nGPT architecture with a fractional HyperCloning technique we call BUS (Bootstrapped model UpScaling).This Method Addresses several key challenges in the development of larger, more capable LLMs.

By leveraging the nGPT architecture, we achieve significant improvements in training efficiency, with 4-20x faster convergence compared to traditional methods. This allows us to capitalize on the rapid inital learning phase of model training, where approximately 15% of training tokens account of about 85% of total loss improvements.

Our fractional scaling approach, inspired by but improving upon HyperCloning, enables more gradual and flexible model growth. This strategy allows for, optimal utilization of compute resources, smoother transitions between model sizes, and better preservation of learned representations during scaling. The introduction of the κ parameter provides a tunable mechanism for determining the ideal point at which to scale the model, balancing the trade-off between extracting maximum performance from smaller models and leveraging the benefits of larger architectures. While our experiments were limited to a 64M parameter model due to computational constraints, the principles demonstrated here have potential implications for scaling much larger models. The combination of nGPT's efficiency gains and our fractional scaling technique opens new possibilities for developing LLMs that are not only larger but also more efficient and robust.

Future work should focus on, applying this technique to larger model scales, investigating the optimal κ values for different model sizes and tasks, exploring the potential for scaling both model width and depth using this approach, and analyzing the long-term effects of this scaling method on model performance and generalization

In conclusion, our BUS method, built upon the foundations of nGPT and HyperCloning, presents a promising direction for the future of LLM development. By enabling more efficient and flexible scaling, we pave the way for the creation of more powerful and resource-efficient language models.

References

An, Chenxin *et al.*, (2024). *Why Does the Effective Context Length of LLMs Fall Short?*, arXiv: 2410.18745 [cs.CL]. https://arxiv.org/abs/2410.18745.

Black, Sid *et al.*, (2021). *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*, version 1.0. If you use this software, please cite it using these metadata. DOI: 10.5281/zenodo.5297715. https://doi.org/10.5281/zenodo.5297715.

Brown, Tom B. *et al.*, (2020). *Language Models are Few-Shot Learners*, arXiv: 2005.14165 [cs.CL]. https://arxiv.org/abs/2005.14165.

Chen, Tianqi, Goodfellow, Ian, and Shlens, Jonathon (2016). *Net2Net: Accelerating Learning via Knowledge Transfer*, arXiv: 1511.05641 [cs.LG]. https://arxiv.org/abs/1511.05641.

Clark, Peter *et al.*, (2018). *Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge*, arXiv: 1803.05457 [cs.AI]. https://arxiv.org/abs/1803.05457.

Cottier, Ben *et al.*, (2024). *The rising costs of training frontier AI models*, arXiv: 2405.21015 [cs.CY]. https://arxiv.org/abs/2405.21015.

Dao, Tri (2023). *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning*, arXiv: 2307.08691 [cs.LG]. https://arxiv.org/abs/2307.08691.

Devlin, Jacob *et al.*, (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv: 1810.04805 [cs.CL]. https://arxiv.org/abs/1810.04805.

Du, Wenyu *et al.*, (2024). *Stacking Your Transformers: A Closer Look at Model Growth for Efficient LLM Pre-Training*, arXiv: 2405.15319 [cs.CL]. https://arxiv.org/abs/2405.15319.

Frankle, Jonathan and Carbin, Michael (2019). *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*, arXiv: 1803.03635 [cs.LG]. https://arxiv.org/abs/1803.03635.

Gao, Tianyu *et al.*, (2024). *How to Train Long-Context Language Models (Effectively)*, arXiv: 2410.02660 [cs.CL]. https://arxiv.org/abs/2410.02660.

Glockner, Max, Shwartz, Vered, and Goldberg, Yoav (2018). *Breaking NLI Systems with Sentences that Require Simple Lexical Inferences*, arXiv: 1805.02266 [cs.CL]. https://arxiv.org/abs/1805.02266.

Gokaslan, Aaron and Cohen, Vanya (2019). *OpenWebText Corpus*, http://Skylion007.github.io/OpenWebTextCorpus.

Grattafiori, Aaron *et al.*, (2024). *The Llama 3 Herd of Models*, arXiv: 2407.21783 [cs.AI]. https://arxiv.org/abs/2407.21783.

Hoffmann, Jordan *et al.*, (2022). *Training Compute-Optimal Large Language Models*, arXiv: 2203.15556 [cs.CL]. https://arxiv.org/abs/2203.15556.

Izsak, Peter, Berchansky, Moshe, and Levy, Omer (2021). *How to Train BERT with an Academic Budget*, arXiv: 2104.07705 [cs.CL]. https://arxiv.org/abs/2104.07705.

Kaplan, Jared *et al.*, (2020). *Scaling Laws for Neural Language Models*, arXiv: 2001.08361 [cs.LG]. https://arxiv.org/abs/2001.08361.

Karpathy, Andrej (2023). *nanoGPT*, https://github.com/karpathy/nanoGPT.

Loshchilov, Ilya *et al.*, (2024). *nGPT: Normalized Transformer with Representation Learning on the Hypersphere*, arXiv: 2410.01131 [cs.LG]. https://arxiv.org/abs/2410.01131.

Mihaylov, Todor *et al.*, (2018). *Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering*, arXiv: 1809.02789 [cs.CL]. https://arxiv.org/abs/1809.02789.

Rae, Jack W. *et al.*, (2022). *Scaling Language Models: Methods, Analysis Insights from Training Gopher*, arXiv: 2112.11446 [cs.CL]. https://arxiv.org/abs/2112.11446.

Sakaguchi, Keisuke *et al.*, (2019). *WinoGrande: An Adversarial Winograd Schema Challenge at Scale*, arXiv: 1907.10641 [cs.CL]. https://arxiv.org/abs/1907.10641.

Samragh, Mohammad *et al.*, (2023). *Weight subcloning: direct initialization of transformers using larger pretrained ones*, arXiv: 2312.09299 [cs.LG]. https://arxiv.org/abs/2312.09299.

Samragh, Mohammad *et al.*, (2024). *Scaling Smart: Accelerating Large Language Model Pre-training with Small Model Initialization*, arXiv: 2409.12903 [cs.CL]. https://arxiv.org/abs/2409.12903.

Su, Jianlin *et al.*, (2023). *RoFormer: Enhanced Transformer with Rotary Position Embedding*, arXiv: 2104.09864 [cs.CL]. https://arxiv.org/abs/2104.09864.

Vaswani, Ashish *et al.*, (2023). *Attention Is All You Need*, arXiv: 1706.03762 [cs.CL]. https://arxiv.org/abs/1706.03762.

Wang, Sinong *et al.*, (2020). *Linformer: Self-Attention with Linear Complexity*, arXiv: 2006.04768 [cs.LG]. https://arxiv.org/abs/2006.04768.

Williams, Adina, Nangia, Nikita, and Bowman, Samuel R. (2018). *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference*, arXiv: 1704.05426 [cs.CL]. https://arxiv.org/abs/1704.05426.

Yang, Cheng *et al.*, (2020). *Progressively Stacking 2.0: A Multi-stage Layerwise Training Method for BERT Training Speedup*, arXiv: 2011.13635 [cs.CL]. https://arxiv.org/abs/2011.13635.

Zellers, Rowan *et al.*, (2019). *HellaSwag: Can a Machine Really Finish Your Sentence?*, arXiv: 1905.07830 [cs.CL]. https://arxiv.org/abs/1905.07830.