

Scaling Compute Instead of Parameters: Exploring Iterative Transformers for Data-Efficient Language Modeling

Wiktor Kamzela¹ and Ondřej Dušek² and Mateusz Lango^{1,2}

¹Poznan University of Technology, Faculty of Computing and Telecommunications, Poland

²Charles University, Faculty of Mathematics and Physics, Czechia

wiktor.kamzela@student.put.edu.pl, {odusek, lango}@ufal.mff.cuni.cz

Abstract

Neural language model architectures that iterate over their layers have recently shown superior performance on reasoning benchmarks and proved to be a promising method for reducing model size while maintaining performance at scale. However, the effectiveness of such architectures in data-constrained settings remains unexplored. We present a systematic study of layer-sharing transformer architectures trained on the BabyLM corpus (10M tokens), across various iteration strategies and a range of layer and iteration counts. We find that iterative architectures are highly effective under data-constrained setting: a 3-layer iterated model outperforms a standard 24-layer transformer model (ModernBERT-inspired) on SuperGLUE benchmark.

1 Introduction

Over the past several years, the dominant strategy for improving pretrained large language models (LLM) has been to scale model size, following empirical scaling laws (Kaplan et al., 2020) that associate larger architectures with better performance. However, this approach substantially increases memory requirements and, according to compute-optimal scaling laws, also requires substantially more training data as model size grows (Hoffmann et al., 2022). With LLMs already reaching hundreds of billions of parameters, further parameter scaling is becoming increasingly inefficient. Consequently, there is growing interest in alternative architectures that can achieve comparable or superior performance with fewer parameters.

One promising direction are *iterative architectures*, in which the same layer is applied multiple times to hidden representations (Yang et al., 2024; Saunshi et al., 2025; Bae et al., 2025). When unrolled, such architectures are equivalent to much deeper networks whose layers share parameters, while substantially reducing memory requirements.

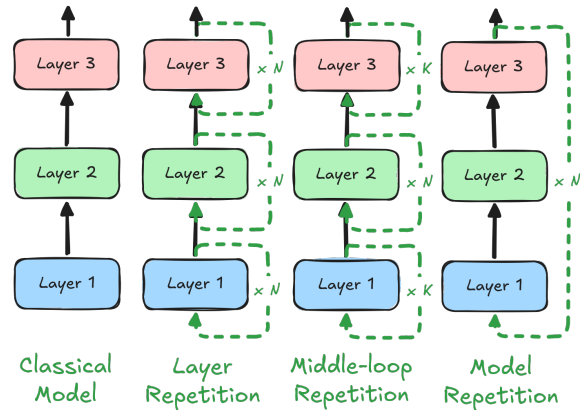


Figure 1: Overview of the classic transformer model and three iteration strategies explored in this paper: layer repetition, middle-loop repetition and model repetition.

Since these techniques are designed to improve performance while reducing the number of trainable parameters, they may be particularly well suited to small-data settings, where a smaller effective capacity can mitigate overfitting, echoing the classical bias-variance trade-off (Hastie et al., 2009).

In this work, we present the first systematic study of iterative, layer-sharing architectures for language modeling in a strictly data-constrained setting, using developmentally plausible corpora (10M tokens BabyLM corpus, Haga et al. 2024). We investigate three families of iterative architectures, explore iteration-specific specialization via LoRA adapters (Hu et al., 2022), as well as investigate further parameter reduction through smaller input representations (reduced vocabulary size and embedding layer factorization). Our experiments span architectures ranging from 1 to 24 transformer layers and from 2 to 16 recursive iterations, resulting in a comprehensive exploration of the trade-offs between model depth, recursion, and parameter efficiency.

The results demonstrate the effectiveness of iterative computation in data-constrained regimes.

For instance, we discover that a model containing only three distinct layers, repeatedly applied through recursion, surpasses the performance of a conventional 24-layer ModernBERT-based architecture (Warner et al., 2025) on Super-GLUE benchmark (Wang et al., 2019) while using the same training data and substantially fewer trainable parameters. This highlights the potential of parameter-sharing architectures for efficient learning from limited data. To boost further research on iterative architectures, we make our code and models publicly available.¹

2 Related Works

Iterative language models In the context of language modeling, iterative computation has previously proven highly effective in models such as ALBERT (Lan et al., 2020), which combined cross-layer parameter sharing (i.e., single-layer repetition throughout the whole model) with embedding factorization to dramatically reduce model size while maintaining competitive performance.

More recently, the concept of iterative computation has been revisited to improve LLM reasoning capabilities. Geiping et al. (2025) proposed recurrent-depth architecture to scale test-time computation and perform reasoning in implicit latent space. Saunshi et al. (2025) further explored latent thoughts by analyzing transformer models that iterate over layers multiple times, showing their higher efficiency on reasoning benchmarks. These architectures can be extended by adding learnable adapters, individual for every iteration, that make the repeated layer act differently in every iteration while maintaining low parameter count (Bae et al., 2025). Many other different iteration schemas have been proposed, such as dynamic selection of the number of iterations basing on token difficulty (Chen et al., 2025) or iterating in the space of word embeddings (Zeng et al., 2026).

The effort to apply iterative language models at large scale showed that such models achieve competitive performance with several times larger state-of-the-art language models (Zhu et al., 2026).

While these works demonstrate the promise of iterative architectures at large scale or for reasoning-heavy tasks, their effectiveness in data-constrained regimes remains unexplored.

Building models on developmentally plausible corpora Several editions of the BabyLM work-

shop (Charpentier et al., 2025; Hu et al., 2024; Warstadt et al., 2023) provided significant insight into techniques effective for data-constrained language modeling on developmentally plausible corpora.

While many different training and data techniques were explored, for instance curriculum learning (Roque and Velasco, 2025), specialized training objectives (Kamzela et al., 2025) or data creation (Ulm et al., 2025), works exploring new architectures are less common: Charpentier and Samuel (2023) explored adding layer weighting to the LTG-BERT architecture (Samuel et al., 2023). Other works explored different variants of recurrent neural networks (Haller et al., 2024), state space models (Lee et al., 2025), replacing attention with mLSTM token mixer (Haller et al., 2025) or training only LoRA adapters with specifically initialized layers (Kumar et al., 2025). An iterative architecture, to the best of our knowledge, was not explored.

3 Iterative Language Models

The classical transformer architecture (Vaswani et al., 2017) used for language modeling is built out of several layers $f(x)$ consisting of self-attention and fully-connected sublayers. The entire transformer network consisting of L such layers can be denoted as:

$$y(x) = (f_L \odot \cdots \odot f_2 \odot f_1)(x) \triangleq \bigodot_{i=1}^L (f_i)(x)$$

where $\odot$ is the compound function operator.

3.1 Three Iterative Architectures

Iterative transformer architectures aim to obtain higher performance by allowing the model to compute longer, without increasing the number of layers or the number of learnable parameters. This is achieved by iterating over the same layers multiple times and applying them to process the intermediate hidden representations. Such models are more parameter-efficient by reusing a layer’s parameters multiple times, training them to perform operations generalizable across iterations, thus effectively achieving higher model depth without actually increasing the number of layers. In this work we explore three main iteration strategies visualized in Fig. 1 and described below.

¹https://github.com/Wector1/BabyLM_iterative

Layer Repetition is an iteration strategy which applies the same layer consecutively to process the hidden representation and only then returning the result to the next layer. This can be alternatively seen as a model with several first layers, followed by several second layers, etc., as shown in Fig. 2 on the left.

Formally, the Later Repetition architecture with L layers and N iterations is defined as:

$$y(x) = \bigodot_{i=1}^L \left(\bigodot_{j=1}^N f_i \right) (x)$$

Model Repetition iterates over the entire, L -layered model instead of iterating over layers:

$$y(x) = \bigodot_{j=1}^N \left(\bigodot_{i=1}^L f_i \right) (x)$$

This can be thought of as a deeper network with $N \cdot L$ layers, constructed by stacking N models on top of each other.

Middle-loop Repetition is inspired by mechanistic interpretability studies (Wendler et al., 2024; Lad et al., 2025), suggesting a simplified conceptual view that LLM inference consists of three coarse phases: encoding the input, reasoning and synthesizing (including knowledge retrieval), and forming the final answer by choosing appropriate tokens. The reasoning phase is considered to be the most complex one, spread across most layers of LLMs (Lad et al., 2025).

Therefore, we propose a 3-layer architecture, reflecting three general operations of the entire processing. While the first and last layer are repeated K times, the middle layer is applied a higher number of M times ($M > K$):

$$y(x) = \left(\bigodot_{j=1}^K f_3 \right) \odot \left(\bigodot_{j=1}^M f_2 \right) \odot \left(\bigodot_{j=1}^K f_1 \right) (x)$$

We assume $N = 2K + M$ for general statements about iterative models.

3.2 Relaxed Iterative Model

Iterative architectures act as deeper $N \cdot L$ layer models while having significantly less parameters, which is achieved by applying the same layer several times. This effectively means, that such models are less expressive than standard $N \cdot L$ layer models because they apply exactly the same operation without any variability.

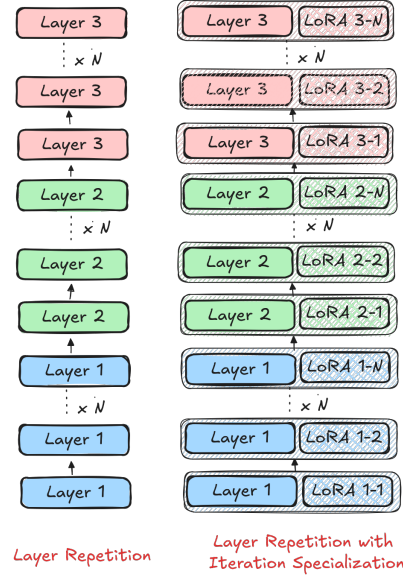


Figure 2: Unrolled visualization of standard Layer Repetition iteration strategy and its variant with iteration specialization.

If we slightly relax the assumption that the number of model parameters does increase at all with the number of iterations chosen, we can enable the model to perform different operations in different iterations, potentially bringing their expressivity closer to deeper classical models while still requiring only a fraction of their parameters. The idea is that while the same layer is applied multiple times to process the hidden representations, it is applied in every iteration with a different LoRA adapter (Hu et al., 2022) that alters the computed operation. In this way, the model has different $N \cdot L$ layers while still having the number of parameters significantly reduced in comparison to classical $N \cdot L$ model. Such architecture is visualized in Fig. 2 on the right and defined below:

$$y(x) = \bigodot_{i=1}^L \left(\bigodot_{j=1}^N f_i \odot LoRA_{ij} \right) (x)$$

3.3 Embedding Compression

As the parameter count of the smallest iterative models we test is largely influenced by the significant size of the embedding matrix, we explore two additional techniques that reduce the number of trainable parameters: embedding factorization and custom tokenizers.

Embedding factorization was adapted from earlier works on ALBERT (Lan et al., 2020), that

proposed the decomposition of the embedding matrix into a shared linear layer and an embedding matrix with reduced dimensionality. Each token has an embedding with a smaller dimensionality d , which is then scaled up to the size of model’s hidden representation h by a linear layer. Since the original embedding matrix has $|V| \times h$ parameters and the new setup has $h \times d + |V| \times d$ parameters, as long as $d \ll h$, a reduction of parameter count is achieved.

Custom tokenizers Another possibility to reduce the size of embedding matrix is to limit the size of vocabulary $|V|$ the tokenizer operates on. To this end, we used the WordPiece algorithm, originally developed for BERT (Devlin et al., 2019), to construct tokenizers on training corpora that have a smaller vocabulary set V .

4 Experiments

4.1 Experimental Setup

Model architecture Our models are based on ModernBERT (Warner et al., 2025) implementation from the HuggingFace library (Wolf et al., 2020). Our baseline model has 149M parameters, closely following the architecture of original ModernBERT-small with default parameters, except for the number of hidden layers set to 24, the context size of 256 and sliding attention disabled. We tested all three iteration strategies as well additional techniques mentioned in Sec. 3.

Iteration strategies We evaluated models with the number of layers $L = \{1, 3, 6, 12, 24\}$ and iterations $N = \{1, 2, 4, 8, 16\}$, except for the largest models, where computational constraints prevented evaluating $N = 16$ for $L = 12$ and $N = \{16, 8\}$ for $L = 24$. The Middle-loop Repetition variant is by definition $L = 3$ and was tested for $K = \{1, 3\}$ and values of M that will result in architectures having $M + 2K$ unrolled layers in the set of $\{6, 12, 24, 48\}$.

Relaxed iterative models were trained with LoRA adapters of dimensionality 64 and $\alpha = 128$. The models are trained with the adapters from the very first epoch and the training was performed without further changes.

Embedding Compression Embedding factorization was applied with an embedding matrix with a dimensionality of 128. The embeddings were scaled up to the model’s 768 dimensionality by an additional linear layer.

As a custom tokenizer, we trained a WordPiece model on the BabyLM 10M-word corpus using the HuggingFace tokenizers library. Prior to vocabulary construction, text was normalized via NFD decomposition, lowercasing, and accent stripping, then pre-tokenized by splitting on whitespace and punctuation. The tokenizer was trained with a vocabulary size of 24,000 and a minimum subword frequency of 2. Note that the original ModernBERT tokenizer has a 50,368 vocabulary size.

Training details Models were trained with AdamW optimizer with 64 batch size on masking modeling objective with masking probability of $p = 0.4$, following the initial setup of Edman and Fraser (2025). Learning rate followed the cosine schedule with 500 warmup steps and a learning rate of 0.0003. A small weight decay of 0.01 was applied. The models were trained for 10 epochs with the best checkpoint selected basing on the best perplexity value on validation set.

Evaluation Our evaluation protocol follows the one provided by the BabyLM Challenge organizers (Charpentier et al., 2025). More concretely, we evaluated on two benchmarks:

- SuperGLUE, which measures language understanding capabilities on tasks like BoolQ, MNLI, MultiRC, RTE, WSC, MRPC, and QQP (Wang et al., 2019);
- BLiMP (the Benchmark of Linguistic Minimal Pairs), consisting of 67 sub-datasets of minimal sentence pairs probing syntax, morphology, and semantics (Warstadt et al., 2020).

Additionally, masked language modeling perplexity was also measured.

For each SuperGLUE task, pretrained models were fine-tuned with default parameters provided by BabyLM organizers, without any further hyperparameter tuning.

5 Results

5.1 Layer Repetition

The results of the Layer Repetition strategy across 5 layer sizes and 5 iteration numbers are shown in Tab. 1. For the baseline model ($N = 1$), the results on both benchmarks grow as expected with the increasing number of hidden layers. Iterative models, while having the same number of parameters, achieve superior results in comparison to the

L (Layers)	N (Iter.)	Params	S. GLUE	BLiMP	Perp.
1	1	44.3M	62.96	61.69	17.69
	2	44.3M	64.66	63.47	14.69
	4	44.3M	63.91	64.07	10.81
	8	44.3M	65.14	66.58	9.91
	16	44.3M	64.13	57.90	28.56
3	1	54.4M	63.59	66.74	10.27
	2	54.4M	65.14	66.60	9.63
	4	54.4M	65.75	68.08	9.46
	8	54.4M	62.45	69.40	9.54
	16	54.4M	64.92	68.06	8.95
6	1	69.4M	65.73	68.12	9.42
	2	69.4M	65.00	68.56	8.79
	4	69.4M	65.66	69.54	8.58
	8	69.4M	66.75	69.57	8.59
	16	69.4M	65.07	70.04	9.23
12	1	99.5M	65.54	70.36	8.42
	2	99.5M	65.55	71.33	8.04
	4	99.5M	65.61	70.15	9.01
	8	99.5M	66.34	71.46	8.77
24	1	159.7M	66.04	71.84	8.34
	2	159.7M	65.72	72.25	8.37
	4	159.7M	67.54	71.50	8.53

Table 1: The results of Layer Repetition strategy across different numbers of layers L (Layers) and the number of iterations N (Iter.). Note that the results for $N = 1$ are standard transformer models. The number of model parameters (Params) and results of Super GLUE (S. GLUE), BLiMP and Perplexity (Perp.) also reported.

baseline and their performance seem to generally increase with increasing the number of iterations.

Impressively, the iterative model with a single layer and $N = 8$ iterations achieves a SuperGLUE performance comparable to a baseline model with 12 trainable layers. This shows the effectiveness of iterative models in modeling complex functions with a limited number of layers, and the potential of scaling the computation time while keeping the parameter count fixed.

The best iterative model in our comparison is the iterative model with $L = 24$ layers and $N = 4$ iterations. The best baseline has also $L = 24$ layers and achieves 1.5 p.p. worse result on SuperGLUE and a slightly better result on BLiMP (0.34 p.p., within standard deviation) than the iterative counterpart.

5.2 Iterative Schema Comparison

Model Repetition is compared to Layer Repetition in Tab. 2. It appears to work better than Layer repetition overall; however, the differences between the architectures are typically within ± 1 p.p. for both benchmarks. Interestingly, even Model repetition with $N = 8$ and $L = 3$ layers

L (Layers)	N (Iter.)	Rep. schema	S. GLUE	BLiMP
1	8	Model rep.	65.08	66.54
		Layer rep.	65.14	66.58
3	16	Model rep.	64.16	58.36
		Layer rep.	64.13	57.90
3	8	Model rep.	66.43	69.40
		Layer rep.	62.45	69.40
3	16	Model rep.	63.97	59.30
		Layer rep.	64.92	68.06
6	2	Model rep.	66.07	68.72
		Layer rep.	65.00	68.56
	4	Model rep.	66.83	70.22
		Layer rep.	65.66	69.54
6	8	Model rep.	66.81	69.82
		Layer rep.	66.75	69.57
12	2	Model rep.	66.14	70.47
		Layer rep.	65.55	71.33
	4	Model rep.	67.10	71.05
		Layer rep.	65.61	70.15
24	1	Baseline	66.04	71.84
	2	Model rep.	66.74	72.00
		Layer rep.	65.72	72.25

Table 2: Comparison of Layer and Model Repetition schema on models with different number of layers L (Layers) and number of iterations N (Iter.).

only achieves better SuperGLUE than the biggest baseline model with $L = 24$. BLiMP seems to be a more challenging benchmark for iterative models, and the improvements over the biggest baseline are obtained only by iterating a model of a similar size.

Middle-loop Repetition A more detailed evaluation on $L = 3$ layered architectures in Table 3 reveals that the differences between the best and the worse iteration schema increase with the increase of the number of iterations. Most importantly, it seems that Layer and Model repetition do not work well in high iterations regimes (48 unfolded layers) and increase of iteration count does not bring further benefits. In contrast, the interpretability-inspired Middle-loop Repetition schema is more stable in high iteration regimes, bringing consistent improvements on SuperGLUE. These models end up having better SuperGLUE than the baseline with $L = 24$ and better BLiMP than the baseline $L = 6$, while reducing the number of trainable layers by 8 and 2 times, respectively.

5.3 Impact of Iteration Specialization

Adding iteration specialization seems to only have a moderate positive impact on iterative models, as

Layers	Un-rolled	Iteration schema	Super GLUE	BLiMP
3	6	Layer Rep.	65.14	66.60
		Model Rep.	64.72	67.95
		Mid.-loop Rep. ($K = 1$)	64.72	67.97
3	12	Layer Rep.	65.75	68.08
		Model Rep.	66.45	69.32
		Mid.-loop Rep. ($K = 1$)	65.60	68.75
		Mid.-loop Rep. ($K = 3$)	65.81	68.60
3	24	Layer Rep.	62.45	69.40
		Model Rep.	66.43	69.40
		Mid.-loop Rep. ($K = 1$)	66.14	69.30
		Mid.-loop Rep. ($K = 3$)	66.41	69.42
3	48	Layer Rep.	64.92	68.06
		Model Rep.	63.97	59.30
		Mid.-loop Rep. ($K = 1$)	66.68	68.41
		Mid.-loop Rep. ($K = 3$)	67.95	68.78

Table 3: Comparison of iteration schemas for models with $L = 3$ hidden layers and a different number of iterations N . For easier comparisons, we report the number of layers in unrolled architectures (Unrolled). For Layer and Model Repetition schemas, the number of iterations N can be computed by dividing this number by $L = 3$. For Middle-loop Repetition, the value of $2K$ needs to be subtracted from it to get M .

shown in the results presented in Fig. 3 (see App.A for raw results). Adding adapters to the baseline architectures results in an inconsistent and small effect on BLiMP but – quite surprisingly – a positive effect on the SuperGLUE benchmark (except for the largest $L = 24$ model). Note that the latter benchmark requires model finetuning, which may indicate better suitability for finetuning. However, the general lack of consistency on SuperGLUE results suggests that the hyperparameters on the standard BabyLM pipeline used for evaluation may be suboptimal for iterative architectures.

Iteration specialization improve results on BLiMP for almost all the values of N and smaller numbers of layers $L \leq 6$, with larger increases for higher values of N . For a larger model ($L = 12$), iteration specialization provides better results only for the highest $N = 4$ tested. For the largest $L = 24$ models, we do not observe any improvement. However, we only tested $N = \{1, 2\}$ in this case due to computational limitations.

In contrast, the results on SuperGLUE are typically worse for models with specialized iterations with the number of iterations N growing. Since this benchmark uses fine-tuning to assess model capabilities, it is possible that the default finetuning parameters are suboptimal for this kind of models.

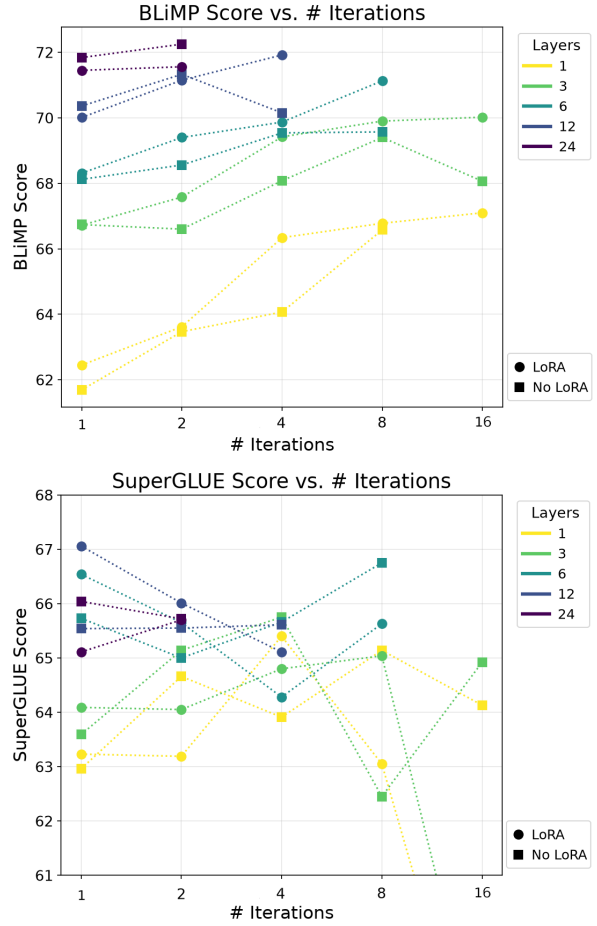


Figure 3: Visualization of models' results with iteration specialization via LoRA compared with standard Layer Repetition iterative models across different numbers of hidden layers L (Layers) and number of iterations N (# Iterations). See the raw numbers in Tab. 5.

Different LoRA parameterizations In Appendix A, we present results with two additional ways of adding LoRA parametrization for consecutive iterations. As LoRA adapters are typically used for fine-tuning the models, we trained the standard iterative model for first few epochs, adding LoRA only in the 8th or 9th epoch. This was accompanied with an increase of learning rate to 0.0003 to allow faster training of newly added parameters.² Nevertheless, the standard technique of having iteration specialization from the beginning of training seems to yield the best results.

5.4 Embedding Compression

Results for models constructed with different input representation techniques are shown in Tab. 4; all are reported for the Layer Repetition schema.

²In our preliminary experiments, it offered better results than keeping the original learning rate.

Lay.	Iter.	Tok.	Fac.	Param	S.GLUE	BLiMP
1	8	M.BERT	Yes	11.7M	64.55	66.49
		M.BERT	No	44.3M	65.14	66.58
		Custom	Yes	8.3M	65.21	66.51
		Custom	No	24.1M	64.81	65.82
	16	M.BERT	Yes	11.7M	34.96	49.85
		M.BERT	No	44.3M	64.13	57.90
		Custom	Yes	8.3M	63.93	51.88
		Custom	No	24.1M	65.23	59.58
3	8	M.BERT	Yes	21.7M	66.43	69.44
		M.BERT	No	54.4M	62.45	69.40
		Custom	Yes	18.3M	68.98	69.14
		Custom	No	34.1M	64.97	67.47
	16	M.BERT	Yes	21.7M	67.72	69.19
		M.BERT	No	54.4M	64.92	68.06
		Custom	Yes	18.3M	67.17	68.22
		Custom	No	34.1M	60.76	67.47
6	8	M.BERT	Yes	36.8M	66.76	69.95
		M.BERT	No	69.4M	66.75	69.57
		Custom	Yes	33.4M	68.73	70.00
		Custom	No	49.1M	67.87	68.68
12	1	M.BERT	Yes	66.9M	68.61	69.87
		M.BERT	No	99.5M	65.54	70.36
		Custom	Yes	63.5M	68.98	68.68
		Custom	No	79.2M	67.34	69.20
	2	M.BERT	Yes	66.9M	67.47	70.56
		M.BERT	No	99.5M	65.55	71.33
		Custom	Yes	63.5M	70.12	69.42
		Custom	No	79.2M	67.39	70.67
	4	M.BERT	Yes	66.9M	67.76	70.76
		M.BERT	No	99.5M	65.61	70.15
		Custom	Yes	63.5M	69.58	69.86
		Custom	No	79.2M	67.32	69.78
24	1	M.BERT	Yes	127.1M	68.13	70.73
		M.BERT	No	159.7M	66.04	71.84
		Custom	Yes	123.7M	69.26	70.60
		Custom	No	139.4M	68.08	69.93
	2	M.BERT	Yes	127.1M	68.99	70.72
		M.BERT	No	159.7M	65.72	72.25
		Custom	Yes	123.7M	69.23	69.87
		Custom	No	139.4M	67.36	71.57

Table 4: Comparison of results obtained with ModernBERT and Custom Wordpiece tokenizer, with or without embedding factorization (Fac.) under Layer Repetition schema.

Combining a custom tokenizer with smaller vocabulary size and embedding matrix factorization, results in approx. 36M decrease in parameter count. For the smallest of our models with $L = 1$, this is equal to more than 5 fold compression of the model size. Interestingly, the impact of this reduction in parameter size often results in an improved predictive performance.

Embedding matrix factorization applied across both tokenizers results in an improved SuperGLUE score for almost all layer and iteration numbers.

It also improved results on BLiMP for a smaller architecture with 3 and 6 layers, but caused a drop – typically up to 2 p.p. – on other architecture sizes.

The Original ModernBERT tokenizer proved to enable better scores on the zero-shot BLiMP benchmark with only two exceptions: $L = 1, N = 16$ and $L = 6, N = 8$ configurations. On the other hand, our custom tokenizer enabled higher scores on SuperGLUE with only one exception ($L = 3, N = 16$).

Overall, for the majority of configurations, switching from original ModernBERT tokenizer to our custom tokenizer with embedding factorization leads to an improved averaged score of SuperGLUE and BLiMP benchmarks while reducing the number of parameters.

6 Conclusions

In this work, we present the first systematic study of iterative, layer-sharing architectures for language modeling under a data-constrained setting, using the 10M-token BabyLM corpus. We tested three iteration strategies: Layer Repetition, Model Repetition, and an interpretability-inspired Middle-loop Repetition; and showed that reusing a small number of trained layers through repeated computation is an effective alternative to scaling parameter count. Notably, a 3-layer iterated model was able to outperform a conventional 24-layer ModernBERT-style baseline on SuperGLUE and a 6-layer baseline on BLiMP, despite using a fraction of the trainable parameters and identical training data.

Finally, we also explored introducing per-iteration LoRA specialization and reducing parameter count through embedding factorization and a custom, smaller-vocabulary tokenizer. We showed that these techniques shrank the model parameter counts while frequently improving rather than harming the overall performance.

Acknowledgments

Co-funded by the European Union (ERC, NG-NLG, 101039303). This research was conducted using the computational infrastructure of the Poznan Supercomputing and Networking Center (PCSS) and used resources of the LINDAT/CLARIAH-CZ Research Infrastructure (Czech Ministry of Education, Youth, and Sports project No. LM2018101).

Limitations

Due to constraints in computation time, the results presented in this work were not averaged over multiple runs with different random seeds.

Ethical Considerations

We acknowledge the use of GitHub Copilot in drafting parts of our code. Some portions of the manuscript were revised using an AI-assisted grammar checker.

References

- Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. 2025. [Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA](#). In *International Conference on Learning Representations*.
- Lucas Charpentier, Leshem Choshen, Ryan Cotterell, Mustafa Omer Gul, Michael Y. Hu, Jing Liu, Jaap Jumelet, Tal Linzen, Aaron Mueller, Candance Ross, Raj Sanjay Shah, Alex Warstadt, Ethan Gotlieb Wilcox, and Adina Williams. 2025. [Findings of the third BabyLM challenge: Accelerating language modeling research with cognitively plausible data](#). In *Proceedings of the First BabyLM Workshop*, pages 399–420, Suzhou, China. Association for Computational Linguistics.
- Lucas Georges Gabriel Charpentier and David Samuel. 2023. [Not all layers are equally as important: Every layer counts BERT](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 238–252, Singapore. Association for Computational Linguistics.
- Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. 2025. [Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28241–28259, Vienna, Austria. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [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, Minneapolis, Minnesota. Association for Computational Linguistics.
- Lukas Edman and Alexander Fraser. 2025. [Mask and you shall receive: Optimizing masked language modeling for pretraining BabyLMs](#). In *Proceedings of the First BabyLM Workshop*, pages 445–453, Suzhou, China. Association for Computational Linguistics.
- Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. 2025. [Scaling up test-time compute with latent reasoning: A recurrent depth approach](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Akari Haga, Akiyo Fukatsu, Miyu Oba, Arianna Bisazza, and Yohei Oseki. 2024. [BabyLM challenge: Exploring the effect of variation sets on language model training efficiency](#). In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 252–261, Miami, FL, USA. Association for Computational Linguistics.
- Patrick Haller, Jonas Golde, and Alan Akbik. 2024. [BabyHGRN: Exploring RNNs for sample-efficient language modeling](#). In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 82–94, Miami, FL, USA. Association for Computational Linguistics.
- Patrick Haller, Jonas Golde, and Alan Akbik. 2025. [Sample-efficient language modeling with linear attention and lightweight enhancements](#). In *Proceedings of the First BabyLM Workshop*, pages 175–191, Suzhou, China. Association for Computational Linguistics.
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The elements of statistical learning*. Springer.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. [Training compute-optimal large language models](#). In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, New Orleans, LA, USA.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Michael Y. Hu, Aaron Mueller, Candace Ross, Adina Williams, Tal Linzen, Chengxu Zhuang, Ryan Cotterell, Leshem Choshen, Alex Warstadt, and Ethan Gotlieb Wilcox. 2024. [Findings of the second BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora](#). In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 1–21, Miami, FL, USA. Association for Computational Linguistics.

- Wiktor Kamzela, Mateusz Lango, and Ondrej Dusek. 2025. [You are an LLM teaching a smaller model everything you know: Multi-task pretraining of language models with LLM-designed study plans](#). In *Proceedings of the First BabyLM Workshop*, pages 469–487, Suzhou, China. Association for Computational Linguistics.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#). *arXiv preprint arXiv:2001.08361*.
- Nalin Kumar, Mateusz Lango, and Ondrej Dusek. 2025. [Pretraining language models with LoRA and artificial languages](#). In *Proceedings of the First BabyLM Workshop*, pages 525–530, Suzhou, China. Association for Computational Linguistics.
- Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. 2025. [Remarkable robustness of LLMs: Stages of inference?](#) In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [Albert: A lite BERT for self-supervised learning of language representations](#). In *International Conference on Learning Representations*.
- Hyunji Lee, Wenhao Yu, Hongming Zhang, Kaixin Ma, Jiyeon Kim, Dong Yu, and Minjoon Seo. 2025. [Understanding and enhancing mamba-transformer hybrids for memory recall and language modeling](#). In *Proceedings of the First BabyLM Workshop*, pages 380–398, Suzhou, China. Association for Computational Linguistics.
- Matthew Theodore Roque and Dan John Velasco. 2025. [Beyond repetition: Text simplification and curriculum learning for data-constrained pretraining](#). In *Proceedings of the First BabyLM Workshop*, pages 246–255, Suzhou, China. Association for Computational Linguistics.
- David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023. [Trained on 100 million words and still in shape: BERT meets British National Corpus](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1954–1974, Dubrovnik, Croatia. Association for Computational Linguistics.
- Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. 2025. [Reasoning with latent thoughts: On the power of looped transformers](#). In *International Conference on Learning Representations*, volume 2025, pages 14855–14881.
- Jannek Ulm, Kevin Du, and Vésteinn Snæbjarnarson. 2025. [Contrastive decoding for synthetic data generation in low-resource language modeling](#). In *Proceedings of the First BabyLM Workshop*, pages 29–41, Suzhou, China. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, Long Beach, California, USA.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. [Superglue: A stickier benchmark for general-purpose language understanding systems](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. [Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. [Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora](#). In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. [BLiMP: The benchmark of linguistic minimal pairs for English](#). *Transactions of the Association for Computational Linguistics*, 8:377–392.
- Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. [Do llamas work in English? on the latent language of multilingual transformers](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15366–15394, Bangkok, Thailand. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Liu Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. 2024. [Looped transformers are better at learning learning algorithms](#). In *The Twelfth International Conference on Learning Representations*.

Boyi Zeng, Shixiang Song, Siyuan Huang, Yixuan Wang, He Li, Ziwei He, Xinbing Wang, Zhouhan Lin, and 1 others. 2026. [PonderLM: Pretraining language models to ponder in continuous space](#). In *The Fourteenth International Conference on Learning Representations*.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, and 14 others. 2026. [Scaling latent reasoning via looped language models](#). *Preprint*, arXiv:2510.25741.

A Additional results of relaxed iterative methods

The results of iterative methods with iteration specialization (via LoRA) are presented in Tab. 5.

Additional results of relaxed iterative methods with different LoRA training schemas are presented in Tab. 6.

Layers	Iter.	LoRA	Super GLUE	BLiMP
1	1	✓ –	63.23 62.96	62.45 61.69
	2	✓ –	63.19 64.66	63.62 63.47
	4	✓ –	65.40 63.91	66.34 64.07
	8	✓ –	63.05 65.14	66.78 66.58
	16	✓ –	56.66 64.13	67.10 57.90
3	1	✓ –	64.09 63.59	66.72 66.74
	2	✓ –	64.05 65.14	67.58 66.60
	4	✓ –	64.80 65.75	69.42 68.08
	8	✓ –	65.04 62.45	69.90 69.40
	16	✓ –	57.58 64.92	70.02 68.06
6	1	✓ –	66.54 65.73	68.31 68.12
	2	✓ –	65.65 65.00	69.40 68.56
	4	✓ –	64.28 65.66	69.87 69.54
	8	✓ –	65.63 66.75	71.13 69.57
12	1	✓ –	67.06 65.54	70.01 70.36
	2	✓ –	66.01 65.55	71.15 71.33
	4	✓ –	65.11 65.61	71.92 70.15
24	1	✓ –	65.11 66.04	71.45 71.84
	2	✓ –	65.70 65.72	71.56 72.25

Table 5: Results of models with iteration specialization via LoRA compared with standard iterative models across different numbers of hidden layers L (Layers) and number of iterations N (Iter.).

Layers	Iter.	LoRA 0th	LoRA 8th	LoRA 9th	Decayed LR	S. GLUE	BLiMP
1	8	✓	✓	✓	0.00030	64.84	65.91
					0.00030	65.13	66.27
					–	63.05	66.78
					–	65.14	66.58
1	16	✓	✓	✓	0.00030	63.90	58.05
					0.00030	64.20	58.23
					–	56.66	67.10
					–	64.13	57.90
3	8	✓	✓	✓	0.00030	64.54	69.11
					0.00030	61.20	69.15
					–	65.04	69.90
					–	62.45	69.40
3	16	✓	✓	✓	0.00030	63.63	67.05
					0.00030	65.28	67.26
					–	57.58	70.02
					–	64.92	68.06
6	8	✓	✓	✓	0.00030	65.74	69.72
					0.00030	66.58	69.69
					–	65.63	71.13
					–	66.75	69.57
12	2	✓	✓	✓	0.00030	65.09	70.98
					0.00030	64.24	71.13
					–	66.01	71.15
					–	65.55	71.33
12	4	✓	✓	✓	0.00030	64.67	71.03
					0.00030	66.13	69.91
					–	65.11	71.92
					–	65.61	70.15
24	1	✓	✓	✓	0.00030	66.36	70.93
					0.00030	67.06	71.15
					–	65.11	71.45
					–	66.04	71.84
24	2	✓	✓	✓	0.00030	64.89	71.48
					0.00030	65.19	72.08
					–	65.70	71.56
					–	65.72	72.25

Table 6: Additional results of relaxed iterative models with different LoRA parameterizations. See Sec. 5.3 for details.