Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Closed-Form Residual-Space Rotation for Offline Transformer Width Growth

Ramasubramanian B

ram@techaarvam.com

Tech-Aarvam

Research Article

Keywords: transformer width growth, model expansion, progressive training, orthogonal operators, function-preserving initialization

Posted Date: August 5th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10580431/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Closed-Form Residual-Space Rotation for Offline Transformer Width Growth

Tech-Aarvam Private Limited; Bengaluru, Karnataka, India; ram@techaarvam.com

Ramasubramanian B

Abstract

Progressive width growth can save compute cost by expanding smaller model checkpoints. Each expansion step is a difficult transition that needs to balance two competing goals: preserving the behaviors previously learned by the model and improving the new block's learning efficacy.

We propose an offline growth recipe with three components. First, the Small Residual Integration Operator (SRIO) applies a closed-form orthogonal rotation at both the input and output residual boundaries. The operator at the block inputs rotates the new subspace's all-ones component toward the old subspace, improving the old weight's read-strength for the new inputs. The output operator is the inverse.

Second, we use block-specific static expansion policies for the attention and the feed-forward network (FFN) blocks: convex expansion for attention and tiled-copy for FFN. Additionally, we add a scalar warmup gate that gradually introduces the newly added attention and FFN outputs. The block-specific expansion and the SRIO operator have the strongest positive impact amoung the three part recipe.

In matched 256->384 width growth experiments, the complete recipe reaches a training loss of 4.2111, compared to 4.2527 for the no-operator baseline and 4.2606 for the LiGO controls reimplemented in the same codebase. Multi-stage runs further demonstrate repeated growth from 24M to 120M parameters. The SRIO and the block-specific expansion are folded into the grown model weights offline and therefore have no persistent runtime operator during training or inference; only the scalar warmup gate is active briefly after growth.

Keywords: transformer width growth, model expansion, progressive training, orthogonal operators, function-preserving initialization

1 Introduction

Progressive model training starts with a smaller model size and expands over many stages to the desired size. The progressive growth strategy can reduce compute and wall clock time to reach the same training loss. (T. Chen, Goodfellow, and Shlens 2016; Gong et al. 2019; Gu et al. 2021; C. Chen et al. 2022; P. Wang et al. 2023; Yao et al. 2024; Y. Wang et al. 2024; Du et al. 2024; Samragh et al. 2024)

Existing width-growth methods primarily address how the grown blocks are initialized using a function-preserving transformation or learned growth operators. We examine the related problem of old and new subspace interaction. If the blocks are too isolated,

pretrained old weights may contribute very little to processing the new inputs; i.e., the interaction between the old and new subspaces is limited. If they are mixed without control, the expansion may introduce a larger disturbance to the learned function. The width-growth recipe has the following components:

- Closed-form SRIO: an offline orthogonal transformation between old and new subspaces.
- Block-specific weight expansion: Attention blocks use a convex expansion, while FFN blocks use a tiled copy.
- Post-growth runtime output warmup: An optional scalar gate gradually introduces the newly added attention and FFN outputs.

SRIO acts only in the two-dimensional plane spanned by the new and old all-ones components (equivalently, their mean directions).

SRIO and the block-specific expansion policy provide clear improvements in the matched comparisons, while the gate is configuration-dependent. In controlled experiments growing the model size with a residual width of 256 to a width of 384 (d256 -> d384), SRIO consistently improves over the corresponding no-operator configurations. The complete recipe reaches a training loss of 4.2111, while the no operator configuration reaches 4.2527. This is also lower than LiGO control runs, with a reimplementation of LiGO in the same codebase.

1.1 Main Ideas

- Model width expansion is offline. The best performing among our experiments uses separate expansion policies for the attention and the FFN blocks.
- Our experiments included many variants of the Residual Integration Operator - (None, Small, medium, large - NRIO, SRIO, MRIO, LRIO). We settled on SRIO-SO, the static orthogonal closed-form variant with a compact 2×2 row-sum representation. A single hyperparameter ϵ determines the row sums and, equivalently, the 2D rotation angle. Our LRIO-LI variant uses an identity-initialized operator and a LiGO-like 100-step learning phase.
- We also use the same operator (SRIO_R) to transform the layer-normalization γ and β parameters. This helped increase the gradient RMS ratios in favor of the new block. Our matched comparison runs kept the source checkpoint, growth boundary, and ϵ fixed. The median new/old RMS ratios increased in all comparisons and in eight of thirteen was larger than one. Results are in Section 4.7.
- In addition to the above offline strategies, we also added a scalar warmup gate (`new_dim_alpha_gate`) to start from zero and gradually increase the new portion of the residual outputs at both the attention and the FFN over a short initial duration of 2 shards. This initially favors function preservation, while gradually introducing the new portion of the outputs.

Our training protocol uses a long single epoch of unique tokens. In most of our experiments, we trained each stage to the Chinchilla-optimal token-count target of 20 times the current stage's parameter count (Hoffmann et al. 2022). Training data include OpenWebText, FineWeb, and FineWeb-Edu. Given the unique-token single-epoch regime, we primarily use training loss to draw inferences; it agrees directionally with validation loss and accuracy.

2 Solution Architecture

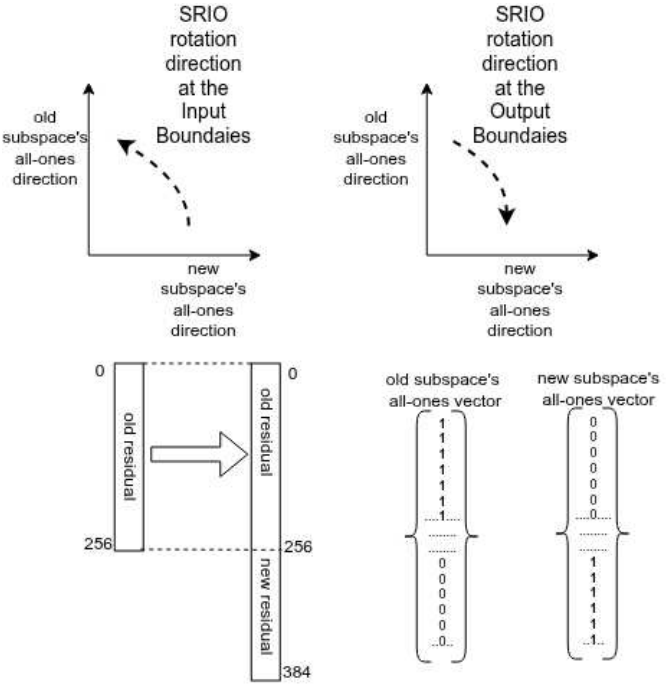


Figure 1. SRIO rotation between the old and new subspaces' all-ones directions at residual boundaries.

The model size growth is offline. The blocks in the model are expanded statically. Section 2.3 describes the specific offline expansion strategies that were evaluated as baseline. We first describe the RIO (Residual Integration Operator).

2.1 SRIO: Small Residual Integration Operator

After the model expansion, the RIO operator is applied offline at each residual boundary. The operator is a rotation on the plane spanning the new and old subspace's all-ones directions (Figure 1).

We express this rotation using the compact row-sum representation $SRIO_{2\times2}$. The row-sum matrix is expanded into the full $d_{new} \times d_{new}$ orthogonal matrix SRIO_R using (5) and (6).

SRIO_R is attached at outputs and rotates the old subspace component along u_{old} toward u_{grow}. SRIO_R^T is attached at inputs and rotates the new subspace component along u_{grow} toward u_{old}.

2.1.1 Derivation of SRIO

Rotation plane and mean components

The old subspace and the new subspaces are of d_{old} and d_{new} dimensions respectively. $d_{grow} = d_{new} - d_{old}$

Let $a = \sqrt{\frac{d_{old}}{d_{grow}}}$

The old all ones unit-vector $u_{old} = \frac{1}{\sqrt{d_{old}}} \begin{bmatrix} \mathbf{1}_{d_{old}} \\ \mathbf{0}_{d_{grow}} \end{bmatrix}$

The new all ones unit-vector $u_{grow} = \frac{1}{\sqrt{d_{grow}}} \begin{bmatrix} \mathbf{0}_{d_{old}} \\ \mathbf{1}_{d_{grow}} \end{bmatrix}$

Input to the operator (which is the output of the model block) x is written as

$$x = \mu_{old}\mathbf{1}_{old} + \mu_{grow}\mathbf{1}_{grow} + x^{\perp}_{old} + x^{\perp}_{grow},$$

where

$$\mathbf{1}^{\top}_{d_{old}} x^{\perp}_{old} = 0, \qquad \mathbf{1}^{\top}_{d_{grow}} x^{\perp}_{grow} = 0,$$

and

$$\mu_{old} = \frac{1}{d_{old}}\mathbf{1}^{\top}_{d_{old}} x_{old}, \qquad \mu_{grow} = \frac{1}{d_{grow}}\mathbf{1}^{\top}_{d_{grow}} x_{grow}.$$

The transformed input is

$$x' = x^{\perp}_{old} + x^{\perp}_{grow} + \left(c\mu_{old} - \frac{s}{a}\mu_{grow}\right)\mathbf{1}_{old} + \left(as\mu_{old} + c\mu_{grow}\right)\mathbf{1}_{grow}. \qquad (1)$$

The inverse transformation is the following:

$$x = x'^{\perp}_{old} + x'^{\perp}_{grow} + \left(c\mu'_{old} + \frac{s}{a}\mu'_{grow}\right)\mathbf{1}_{old} + \left(-as\mu'_{old} + c\mu'_{grow}\right)\mathbf{1}_{grow}. \qquad (2)$$

Row sum representation and component gains

The coefficients in (1) are represented as

$$\text{output_SRIO} = SRIO_{2\times 2} = \begin{bmatrix} c & -s/a \\ as & c \end{bmatrix}. \qquad (3)$$

The coefficients in (2) are represented as

$$\text{input_SRIO} = \begin{bmatrix} c & s/a \\ -as & c \end{bmatrix}. \qquad (4)$$

Note that output_SRIO^T $\neq$ input_SRIO.

We represent the block input read-strengths and the output write-strength using the variables below. The gains are: Input-new gain (RT_new $= c - as$), Input-old gain (RT_old $= c + \frac{s}{a}$), Output-new gain (R_new $= c + as$), and Output-old gain (R_old $= c - \frac{s}{a}$).

Constraints and closed-form solution

We use the following two constraints to reach a closed-form solution for the $SRIO_{2\times2}$

- orthogonality of SRIO_R $\in \mathbb{R}^{d_{new}\times d_{new}}$
- New block gain at input RT_new = ϵ. ϵ is small.

A lower ϵ rotates the newly grown subspace's all-ones component more aggressively toward the old subspace. Informally, this action can be seen as preferential use of the old weights for both new and old residual inputs. However, the action of new weights on both old and new inputs is throttled by the small ϵ.

The ϵ is a configurable knob. Appendix Section A.9 provides loose guidelines on the choice of epsilon for various growth ratios $\frac{d_{old}}{d_{grow}}$.

Orthogonality implies

$$c^2 + s^2 = 1$$

Equating RT_new to ϵ gives $c - as = \epsilon$. We solve the quadratic equation to get the values for c and s.

$$c = \epsilon + as$$

$$s = \frac{-\epsilon a + \sqrt{a^2 + 1 - \epsilon^2}}{a^2 + 1}$$

Dense expansion

The dense operator is

$$\text{SRIO_R} = \begin{bmatrix} OO & N2O \\ O2N & NN \end{bmatrix} \qquad (5)$$

$$
\begin{aligned}
OO &= \mathbf{I}_{d_{old}} + \frac{oo-1}{d_{old}}\,\mathbf{1}\mathbf{1}^T, \\
N2O &= \frac{n2o}{d_{grow}}\,\mathbf{1}\mathbf{1}^T, \\
O2N &= \frac{o2n}{d_{old}}\,\mathbf{1}\mathbf{1}^T, \\
NN &= \mathbf{I}_{d_{grow}} + \frac{nn-1}{d_{grow}}\,\mathbf{1}\mathbf{1}^T.
\end{aligned}
\tag{6}
$$

Each entry of $N2O$ is $-s/\sqrt{d_{old}d_{grow}}$, and each entry of $O2N$ is $s/\sqrt{d_{old}d_{grow}}$.

2.1.2 Operation at each block boundary

At inputs we use SRIO_R^T, e.g., $W\ R^T$. At outputs we use SRIO_R, e.g., $R\ W$.

Figure 2 shows the attachments using mathematical column-vector notation rather than implementation-specific PyTorch notation.

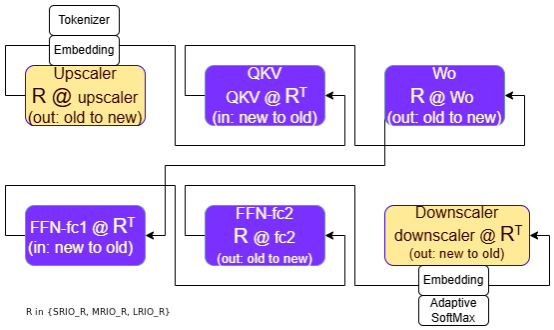


Figure 2. Attachment point for the Residual Integration Operators.

2.1.3 LayerNorm γ and β transformation

In addition, SRIO_R transforms the layer normalization parameters γ and β, taking inspiration from LiGO (P. Wang et al. 2023).

Layer normalization is a nonlinear operation. The attachment at the pre-LayerNormalization points unlike the other boundaries is not a geometric linear transformation, instead, it aims to boost the gradients to the newly grown parameters. This attachment boosts the measured new to old gradient RMS ratios. Results are reported in Section 4.7 and Appendix Table 10.

2.2 Scalar warmup gate

A scalar warmup gate (`new_dim_alpha_gate`) gates the new dimensions of the output from attention as well as FFN (Bachlechner et al. 2020; De and Smith 2020; Touvron et al. 2021; Karras et al. 2018). `new_dim_alpha_gate` is directly at the output of the attention and FFN before the residual addition. The old subspace outputs are passed through. The new outputs start fully gated, then warm up over 2 shards. Each shard in our setup is 20 million

tokens. Warmup immediately after expansion is over a 40M-token training phase in all the experiments that show `new_dim_alpha_gate` as ON.

2.3 Expansion strategies

We found that attention and FFN benefited from using different expansion strategies.

1. Convex init for the attention blocks (QKV, Wo, upscale/downscale)
2. Tiled copy for feed forward network (FFN) (Both layers FFN-up and FFN-down)

FFN-up is the first layer that transforms from the input dimensions, d_{new}, to the hidden dimensions, $4 \times d_{new}$. FFN-down follows the GELU and transforms from the hidden dimensions to d_{new}. FFN-up is also often referred to as fc1 and FFN-down as fc2.

2.3.1 Convex Init for the attention blocks

Following LiGO's factorized linear growth-operator form (P. Wang et al. 2023), we use offline convex-coefficient matrices. Convex combinations of existing columns create new input columns, and convex combinations of existing rows create new output rows.

$$W_{new} = B_{out} W_{old} B_{in}^{T}$$

B_{in} grows the input-column side through right multiplication by B_{in}^{T}, and B_{out} grows the output-row side through left multiplication. This operation grows $W_{old} \in \mathbb{R}^{d_{old} \times d_{old}}$ to $W_{new} \in \mathbb{R}^{d_{new} \times d_{new}}$.

The matrices take the form

$$B_{out} = \begin{bmatrix} I \\ C_{out} \end{bmatrix}, \qquad B_{in} = \begin{bmatrix} I \\ C_{in} \end{bmatrix},$$

where $B_{in}, B_{out} \in \mathbb{R}^{d_{new} \times d_{old}}$ and I is the $d_{old} \times d_{old}$ identity matrix. The random coefficients in each row of C_{in} and C_{out} are nonnegative and sum to one.

2.3.2 FFN path-preserving copy

Inspired by copy-based width-expansion methods such as Net2Net (T. Chen, Goodfellow, and Shlens 2016) and bert2BERT (C. Chen et al. 2022), for both fc1 and fc2 of the FFN we place a tiled and cropped copy of the old weight in the new-new portion. Let $\mathcal{T}_{r \times c}(W)$ denote W tiled as needed and cropped to shape $r \times c$. The two rectangular FFN weights expand as

$$W_{\text{fc1,new}} = \begin{bmatrix} W_{\text{fc1,old}} & \mathbf{0}_{4d_{old} \times d_{grow}} \\ \mathbf{0}_{4d_{grow} \times d_{old}} & \mathcal{T}_{4d_{grow} \times d_{grow}}(W_{\text{fc1,old}}) \end{bmatrix}.$$

$$W_{\text{fc2,new}} = \begin{bmatrix} W_{\text{fc2,old}} & \mathbf{0}_{d_{old} \times 4d_{grow}} \\ \mathbf{0}_{d_{grow} \times 4d_{old}} & \mathcal{T}_{d_{grow} \times 4d_{grow}}(W_{\text{fc2,old}}) \end{bmatrix}.$$

GELU acts element-wise on the outputs of fc1. Each row of fc1 defines one feature selector followed by its corresponding GELU activation. Empirically, different expansion policies worked better for FFN and attention. We attribute this to fundamentally different operations, despite both having nonlinearities - GELU applies independently to each selector row and output feature, while attention applies softmax to $QK^{\top}$ to form weights for linear combinations of the value vectors.

2.4 Residual Integration Operator - Baseline/control alternatives

Residual Integration Operator - Evaluation Variants

The RIO variants and naming convention evaluated are given below.

- NRIO stands for No Residual Integration Operator
- SRIO stands for Small Residual Integration Operator. 2×2 - Small RIO.
- MRIO stands for Medium Residual Integration Operator. $k \times k$ - Current growth stage is k.
- LRIO stands for Large Residual Integration Operator. $d_{new} \times d_{new}$ - full residual wide RIO.

RIO - Initialization Alternatives and their naming

- SO: Static-Orthogonal - the main proposed closed-form solution in Section 2.1.1.
- SD: Static-Diagonal - off-diagonal scaling with diagonal as 1; expansion uses the same mechanism as Section 2.1.1.
- LI: Learning-Identity - identity initialized and LiGO-like 100-step runtime learning.
- SBKV: Static-Best-Known-Value - hard-coded values from previous learning or similar experimental runs, used for reproduction runs.

The naming convention combines the operator type and the initialization variant for the operator. For example, the main solution is SRIO-SO: Small Residual Integration Operator with static orthogonal initialization.

In the learning modes, whether the orthogonality constraint is enabled or disabled is configurable. Following prior soft-orthogonality regularization (Bansal, Chen, and Wang 2018), when enabled we add the following orthogonality loss. Section 4.6 shows that this lowered loss.

$$L_{ortho} = \frac{\lambda}{d} \| R^T R - I_d \|_F^2 \qquad (7)$$

2.4.1 MRIO: Medium Residual Integration Operator

If the current growth stage is k, there are m=k-1 older regions and one new region. SRIO calls out two subspaces old and new. In MRIO, there are k subspaces.

For each subspace, we arrive at rotation/row-sum targets - the same ideas as SRIO, but instead of 1 rotation angle, we have m rotation angles.

In SRIO we use an input-side hyperparameter constraint as RT_new $= \epsilon$. Instead, for MRIO, we use output-side targets, i.e., R_new and R_old$_i$. MRIO is mindful of there being k stages and allows picking aged targets. For example, the direction of rotation for the newest old direction and the older stages are opposite, and the hypothesis behind MRIO is that aged targets and finer control help.

The final age-adjusted operator is

$$\text{MRIO_R} = MRIO_{\text{age}} \times \text{SRIO_R}.$$

$MRIO_{\text{age}}$ is also kept orthogonal. The detailed derivation is in Appendix Section A.5. MRIO showed marginally lower loss compared to SRIO under matched comparisons. However, prolonged training narrowed this gap and SRIO ended lower or similar.

2.4.2 LRIO: Large Residual Integration Operator

LRIO is initialized as the identity matrix of order $d_{new} \times d_{new}$ and learned over a LiGO-like 100-step runtime phase. We also used a configurable orthogonality penalty as described in Equation 7.

2.4.3 Baseline expansion policies

We use baseline/control expansion policies to check whether the observed gains are specific to the proposed expansion recipe or hold more broadly across static, learned, spectral, and copy-based variants. The detailed policy definitions and control settings can be found in Appendix Section A.1. Their comparison is shown in the loss curves in Section 4.3.

3 Experiment setup and methodology

We use OpenWebText, FineWeb, and FineWeb-Edu from Hugging Face. One full progression also used a much smaller number of conversational, programming, and tool-call shards. Data are tokenized offline using `tiktoken` with the GPT-2 byte-level BPE vocabulary of 50,257 tokens (Radford et al. 2019).

Training is indexed by source-file shards capped at 20 million tokens. The completed full progressions used 500 shards (10.0 billion source tokens) for NRIO and 520 shards (approximately 10.195 billion source tokens) for SRIO and MRIO; the latter manifests contain two shorter shards. The per-stage shard counts are reported in Table 2.

Each growth stage increases the inner residual width across all transformer blocks. All models use eight transformer layers and four attention heads. The input embedding width remains 256, with linear upscaling to the current residual width. The total parameter count ranges from 24.3M at $d = 256$ to 119.2M at $d = 1024$. Expansion is performed offline by transforming a saved checkpoint according to the selected expansion policies.

Seeds are applied to both offline expansion and training. Appendix Table 8 shows run-to-run loss variability from three-seed comparison runs. CuDNN deterministic algorithms are not enforced.

The context length is 256. Stages above $d = 256$ use batch size 128 and learning rate 0.00018; $d = 256$ uses batch size 256 and learning rate 0.0003. Parameters, gradients, and Adam states use float32; mixed-precision computation uses BF16. Training uses Adaptive softmax (Grave et al. 2017), fused QKV, and scaled dot-product attention. Shared training settings and the LiGO control configuration are listed in Appendix Table 7.

4 Results

4.1 Residual Integration Operator comparison

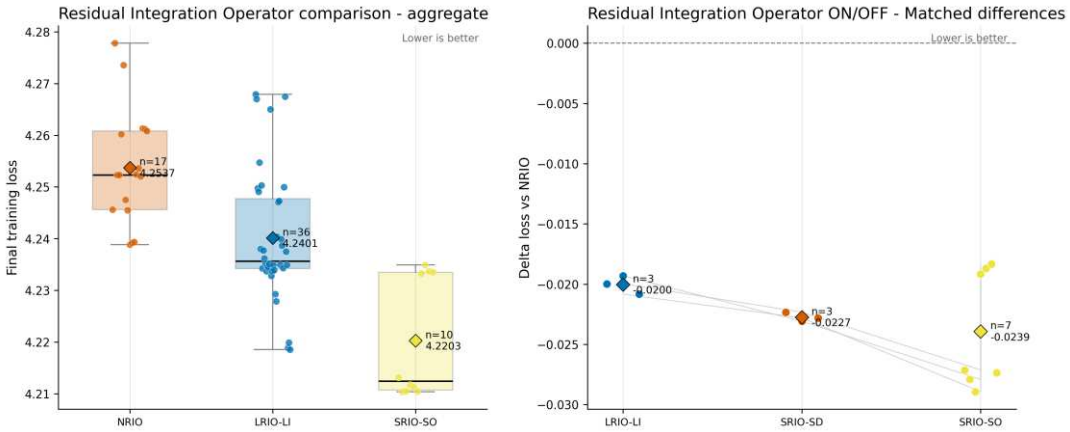


Figure 3. Residual integration operator ON/OFF comparisons, left: aggregate, right: control matched.

Table 1. Matched static orthogonal (SO) and learned orthogonal (LO) comparison.

Growth	ϵ	SO loss	LO loss	LO−SO
256→512	0.50	4.1280	4.1266	-0.0014
512→768	0.50	3.8933	3.8927	-0.0007

Left panel in Figure 3 is an aggregate of all available runs (various expansion policies and configurations excluding only extreme cases) with a stage 1 width of 256 and stage 2 width of 384. Stage 1 was trained for 30 shards and stage 2 for another 40 shards, for a net 70 shards. The excluded extremes are zero initialization policy and alpha (spectral multiplier) of 10x.

Right panel uses matched pairs. Settings including Expansion policies, scaling multipliers and `new_dim_alpha_gate` are kept fixed. The operator and its initialization options are varied and evaluated.

The operator loss ordering is NRIO > LRIO > SRIO (lower is better). Within SRIO, the initialization and static/learning ordering is SD > SO > LO. The LO, the learning-enabled orthogonal closed-form variant, reached the lowest loss.

LO improvement over SO is small and of a comparable magnitude to the run-to-run variations shown in Appendix Table 8. We consider LO and SO as equivalent due to the small difference. The LO variant is viable, given the addition is only 4 learnable parameters and has a negligible run-time penalty.

4.2 Residual Integration Operator curve fit

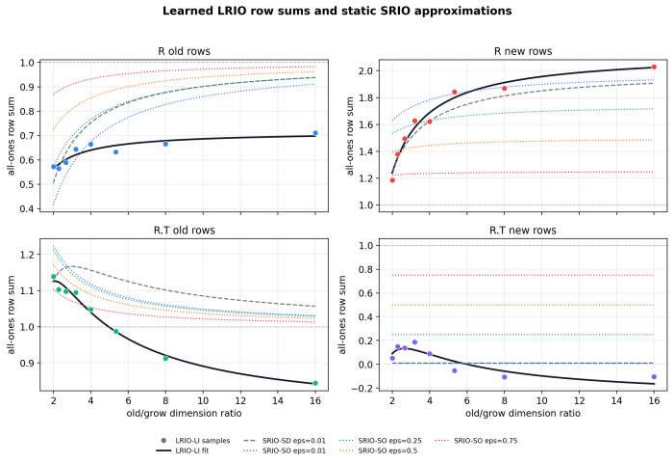


Figure 4. LRIO row-sum curve fit with static SRIO closed-form approximations across old/grow ratios and epsilon settings.

LRIO is full dense and identity-initialized, and this operator goes through the LiGO-inspired 100-step learning phase before it is folded into the weights of the model. We ran LRIO for various stage-to-stage growth ratios. Figure 4 shows the mean row-sum for the old and new subspaces at the input and output. The figure also shows the static closed-form SRIO variants and the Bayesian information criterion (BIC)-selected curve fit. Each curve corresponds to one hyperparameter setting.

Section 2.1.1 shows the detailed derivation of the winning closed-form SRIO-SO variant.

4.3 Expansion Policy comparison

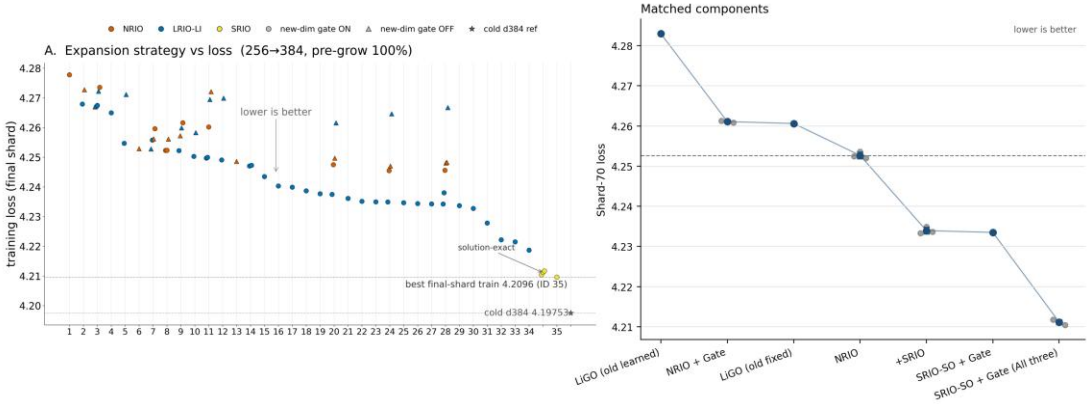


Figure 5. Loss curves and matched component comparison ($d256 \rightarrow d384$).

In the left panel of Figure 5 each point in the x-axis is a particular expansion policy. The experiment IDs are mapped to their expansion policies and algorithms in Appendix Section A.1.

The matched component data in the right panel are tabulated in Appendix Table 9. The right panel shows matched runs where the three parts of the solutions are shown separately and together and also for reference the LiGO reimplementation. This shows that the combined three part strategy reached the best loss among these experiments.

We ran several experiments across many configurations and seed combinations. Some of the experiment arms also had matched variants for the Residual Integration Operator and the `new_dim_alpha_gate` settings. In this set of experiments stage-1's residual width is 256 and the expanded stage-2's residual width is 384. The lowest loss points at the right-most use the SRIO operator variants. SRIO-LO and SRIO-SO are the best performing across the various baseline runs.

4.4 Multi-stage progressive sample runs

The goal of these runs was to rule out issues or unexpected behavior during multi-stage model growth. We grew the model to 120M parameters using different growth factors at each stage, increasing its width from (d=256) to (d=1024). Because these runs were intended primarily as functional checks, we also used a different dataset shuffle and mixture, along with different growth-stage and operator combinations. With a context length of 256, the resulting loss and accuracy were within the expected range for a 120M-parameter model.

Table 2 shows the shard count per stage, training loss, LAMBADA loss, and generic next-token top-1 accuracy.

The LAMBADA dataset is not used in training. A separate evaluation run measures its full-dataset mean loss and top-1 accuracy.

Table 2. Multi-stage progressive sample runs.

Run	Stage shards	Train loss	LAMBADA loss	LAMBADA top-1
Progressive (expansion: diag_const, NRIO)	30:40:50:70:80:105:125	3.5513	4.3327	31.11%
Progressive (expansion: attn-convex+ffn-tiled-copy, SRIO)	20:30:35:105:330	3.5264	4.1835	31.69%
Progressive (expansion: attn-convex+ffn-tiled-copy, MRIO)	20:30:35:105:330	3.5275	4.1701	31.64%

4.5 Scalar warmup gate

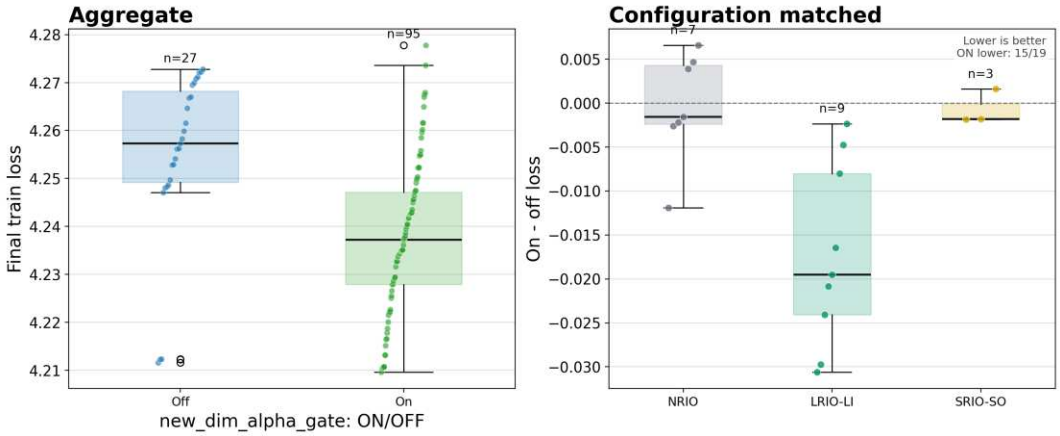


Figure 6. Left: `new_dim_alpha_gate` ON/OFF aggregate. Right: matched comparisons by NRIO, LRIO, and SRIO

Figure 6 shows `new_dim_alpha_gate` ON/OFF comparisons. Left side plot is an aggregate across various expansion policies and configurations. This shows that the gate being on was generally at a lower loss.

The right side filters matched pairs, where the configuration and the expansion policies are fixed and the `new_dim_alpha_gate` is varied. It also shows the delta from the baseline with the gate OFF. More negative is better: a more negative delta indicates a larger benefit relative to gate OFF.

The gate was not useful for the LiGO-layerwise expansion policy (one of the NRIO cases). The LiGO-layerwise is a reimplementation of LiGO in our code base. NRIO data points and their losses are presented in Appendix Table 17. All matched data points and their losses are presented in Appendix Table 16.

4.6 Orthogonality

Equation 7 describes the orthogonality penalty. Table 3 reports shard-70 training loss for matched $\lambda = 0$ and $\lambda = 1$ runs; lower is better. Adding the penalty lowered loss in both matched pairs. The aggregate and matched-pair distributions are shown in Appendix Figure 8.

Table 3. Matched orthogonality-penalty comparison.

n_0	$\lambda = 0$ train	n_1	$\lambda = 1$ train
4	4.2337	1	4.2293
1	4.2250	1	4.2215

4.7 LayerNorm γ and β transformation

Table 4 compares various options for how γ and β in the layer normalization are transformed. Transforming γ and β using SRIO_R as SRIO_R $\cdot \gamma$ and SRIO_R $\cdot \beta$ helped improve the loss.

Residual and gradient RMS ratios between the new and the old subspaces are shown in Appendix Table 15 and Appendix Table 10.

On the same 256-token window, the residual and gradient ratios show that the intended effect is visible immediately after expansion.

The offline LayerNorm γ and β transformation is described in Section 2.1.3.

Transforming both γ and β gave only a marginal advantage over transforming γ alone. The loss difference was comparable to seed-to-seed variation.

Table 4. LayerNorm γ and β transformation comparisons at shard 70.

Matched setting	Variant	Code	LayerNorm transform	Train	Δ Train
LRIO-LI	$R\gamma, R\beta$	`residual`	use $R\gamma$ and $R\beta$	4.2278	+0.0000
	γ, β (no R)	`none`	preserve γ and β	4.2428	+0.0150
LRIO-LI	$R\gamma, R\beta$	`residual`	use $R\gamma$ and $R\beta$	4.2215	+0.0000
	$R\gamma$ only	`gamma_only`	use $R\gamma$; preserve β	4.2218	+0.0003
SRIO-SO	$R\gamma, R\beta$	`residual_static_fold`	use $R\gamma$ and $R\beta$	4.2104	+0.0000
	$R\gamma$ only	`gamma_only`	use $R\gamma$; preserve β	4.2105	+0.0001
SRIO-LO	$R\gamma, R\beta$	`residual`	use $R\gamma$ and $R\beta$	4.2098	+0.0000
	$R\gamma$ only	`gamma_only`	use $R\gamma$; preserve β	4.2097	-0.0001

4.8 FFN and attention expansion policies

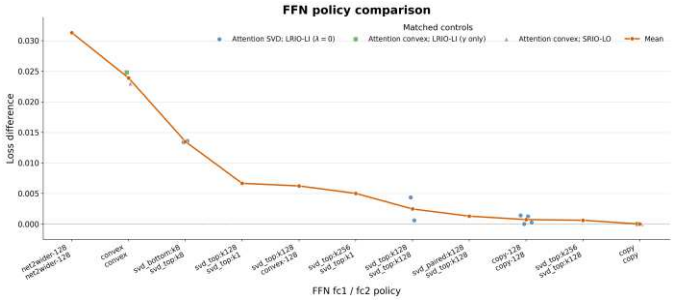


Figure 7. Matched FFN expansion-policy loss differences.

Figure 7 varies the FFN fc1/fc2 expansion policy. Within each matched set, the attention expansion policy, RIO configuration, `new_dim_alpha_gate`, and other training settings are fixed. Each marker is the loss difference from the lowest-loss FFN policy in that set; lower is better. The line connects the mean for each FFN policy across matched sets. The underlying comparisons are listed in Appendix Table 18.

5 Prior Work

Progressive model growth reuses a smaller model through function-preserving, copied or learned transformations. Net2Net and Network Morphism introduced function-preserving width and depth transformations (T. Chen, Goodfellow, and Shlens 2016; Wei et al. 2016), while progressive stacking grows depth by duplicating layers (Gong et al. 2019). bert2BERT, LEMON and HyperCloning apply related transformations to pretrained language models (C. Chen et al. 2022; Y. Wang et al. 2024; Samragh et al. 2024). CompoundGrow jointly schedules input length, width and depth (Gu et al. 2021). Masked Structural Growth uses external masks to make growth strictly function-preserving while progressively introducing new structures (Yao et al. 2024), while Du et al. (2024) study when growth should occur.

LiGO is the most direct learned-operator baseline for this work (P. Wang et al. 2023). It parameterizes growth through factorized linear operators whose convex-coefficient form covers several copying and averaging policies. The operators are learned for 100 steps before being folded into the expanded model. RIO instead derives a compact orthogonal operator in closed form and applies it offline.

ReZero, SkipInit and LayerScale provide precedents for scalar, per-layer and per-channel residual modulation (Bachlechner et al. 2020; De and Smith 2020; Touvron et al. 2021). Soft orthogonality regularization (Bansal, Chen, and Wang 2018) motivates the orthogonality penalty evaluated with LRIO. GradMax uses an SVD construction to initialize new neurons while maximizing their gradient norm (Evci et al. 2022); it motivates the SVD-based expansion controls used in our comparisons.

Effective rank, spectral dynamics and weight statistics have also been studied as descriptions or predictors of optimization and model quality (Roy and Vetterli 2007; Yunis et al. 2024; Martin, Peng, and Mahoney 2021; Unterthiner et al. 2020; Jastrzębski et al. 2020).

6 Limitations and Future Work

The model sizes we work with range from 24M to 120M parameters. We also used a small context length of 256. GPT-2 tokenizer is the only vocabulary utilized. The training datasets are OpenWebText, FineWeb, and FineWeb-Edu. While the closed-form solution is scale agnostic, explicit verification with other datasets and vocabularies or at the billion-parameter scale is not covered in our studies and is marked for future work. Our training uses exactly one epoch, with a ballpark Chinchilla-optimal count of unique tokens.

Validation loss and training loss tracked closely and directionally stayed very correlated, as we would expect from the one-epoch unique-token approach. We have made hyperparameter recommendations in Appendix Section A.9 to be mindful of the residual skew per region corresponding to each growth stage. The impact of such remnant residual skews on SFT and post-training needs further study. The present work has also focused on width expansion. The impact on depth growth and specific depth-growth strategies aligned with this work is marked for future work.

While the overall difference between MRIO and SRIO is small, MRIO + aggressive ϵ had a marginal loss advantage compared to SRIO, which reversed only at approximately 120% of the Chinchilla-optimal token count. We also noticed differences between these operators in their response to a new dataset. These observations may require further studies.

7 Conclusion

We studied the old and new subspace interaction after width expansion. SRIO improves this interaction through a closed form orthogonal rotation in the plane spanning the old and new subspace all-ones directions. In matched experiments, SRIO improved over no-operator baseline and was comparable to the learned orthogonal initialized variant.

Combined with the convex init for attention and tiled copy for FFN, the method reached a training loss of 4.2111, comapred with 4.2527 for the matched no-operator variants and 4.2606 for the LiGO control from the LiGO reimplementation in our codebase. The SRIO and expansion are folded offline, and therefore do not require a runtime persistent operator. The method was also applied over repeated growth stages from 24M to 120M, demonstrating multi-stage feasibility.

Statements and Declarations

Funding

No external funding was received.

Competing interests

The author is the founder and a director of Tech-Aarvam Private Limited, under whose affiliation this work was conducted. The author declares no other competing interests.

Data availability

The study used the publicly available OpenWebText, FineWeb, and FineWeb-Edu datasets. The processed experimental results supporting the findings are available from the corresponding author upon reasonable request.

Code availability

The implementation and runner configuration are available at https://github.com/techaarvam/srio_model_growth.

Author contributions

Ramasubramanian B conceived and designed the study, developed the method and software, conducted the experiments, analyzed the results, and wrote and approved the manuscript.

A Appendix Findings

A.1 Baseline/control expansion policies

While the section above provides the specific top solution we present as the best performer, we evaluated against many configuration settings and expansion policies. This section captures definitions and baseline implementation details, which will help readers interpret the baseline comparison charts with the required context.

A.1.1 Expansion policy baselines

We evaluated a large array of static expansion policies (T. Chen, Goodfellow, and Shlens 2016; P. Wang et al. 2023; Y. Wang et al. 2024; Evci et al. 2022). Each policy defines how an expanded weight matrix is initialized, taking W of size $d_{\text{old}} \times d_{\text{old}}$ to $d_{\text{new}} \times d_{\text{new}}$. We write $\Delta = d_{\text{new}} - d_{\text{old}}$, $\sigma_{\text{med}} = \text{median}_i \sigma_i(W)$, and $s = \alpha\, \sigma_{\text{med}}$.

Table 6 summarizes the baseline expansion policies used in the comparison runs.

Table 5 maps the IDs on the x-axis in Figure 5 to their expansion policies.

Table 5. Lookup-table for the loss curve figure. ID to expansion policy mapping

ID	Policy	ID	Policy
1	`QKVWoFFN-centroid`	19	`QKVWo-svd_FFN-copy k32`
2	`QKVWoFFN-diag-const`	20	`QKVWoFFN-ortho`
3	`QKVWoFFN-diag-rand`	21	`QKVWo-svd_FFN-copy k64`
4	`QKVWo-svd_FFN-net2wider`	22	`QKVWo-svd_FFN-copy k96`
5	`QKVWo-svd-bottom_FFN-copy`	23	`QKVWo-svd_FFN-paired`
6	`ligo_nogrow`	24	`QKVWoFFN-copy`
7	`ligo_layerwise_oldlearnable`	25	`QKVWo-svd_FFN-copy k192`

ID	Policy	ID	Policy
8	`random_orthogonal_nonconvex_coeff`	26	`QKVWo-svd_FFN-copy k256`
9	`ligo_layerwise_oldfixed`	27	`QKVWo-svd_FC1k256_FC2k128`
10	`QKVWo-svd-quad_FFN-copy`	28	`QKVWo-svd_FFN-copy k128`
11	`QKVWoFFN-dense-rand`	29	`QKVWo-svd_FFN-copy k160`
12	`QKVWo-svd_FFN-copy k15`	30	`QKVWo-svd_FFN-copy + gate-ha+woha`
13	`QKVWoFFN-spectral`	31	`QKVWo-svd_FFN-copy`
14	`QKVWo-svd_FFN-lowrank`	32	`QK-svd_VWo-convex_FFN-copy LRIO-LI`
15	`QKVWo-convex_FFN-convex`	33	`QKV-svd_Wo-convex_FFN-copy`
16	`QKVWo-svd_FC2-lowrank`	34	`QKVWo-convex_FFN-copy`
17	`QKVWo-svd_FC2-convex`	35	`QKVWo-convex_FFN-copy SRIO-LO`
18	`QKVWo-svd_FC1k256_FC2k1`		

Table 6. Definitions of the expansion-policy baselines.

Policy name	Equation	Explanation
LiGO / convex template	$W_{\text{expanded}} = B_o W B_i^{\top}\ B_o = [I^{\top}, C_o^{\top}]^{\top}, \quad B_i = [I^{\top}, C_i^{\top}]^{\top}$	Convex row-stochastic coefficient blocks fill the old–new cross blocks from old weights.
LiGO-layerwise	$W_{\text{new}}^{(\ell)} = B_o^{(\ell)} W^{(\ell)} B_i^{(\ell)\top}$	Per-layer LiGO operators learned over 100 steps.
LiGO-shared-coeffs	$W_{\text{new}}^{(\ell)} = B_o W^{(\ell)} B_i^{\top}$ for all ℓ	One learned LiGO operator is shared across layers.
convex-init	$W_{\text{expanded}} = B_o W B_i^{\top}$, with sampled static C_o, C_i	The LiGO block form is sampled once offline and not learned.
Block-diagonal template	$W_{\text{expanded}}^{oo} = W, \quad W_{\text{expanded}}^{on} = W_{\text{expanded}}^{no} = 0, \quad W_{\text{expanded}}^{nn} = N$	Old block is preserved, cross blocks are zero, and only the new–new block is initialized.
`svd_top`	$N = \alpha\tau B / \| B \|_F\ B = (U_{[:,K]} \Sigma_K V_{[K,:]}^{\top})_{1:\Delta,1:\Delta}\ \tau = \| W \|_F\ \Delta / d_{\text{old}}$	The new block is a crop of the top-k SVD reconstruction, RMS-matched to W.

Policy name	Equation	Explanation
`svd_bottom`	Same as `svd_top`, with K selecting the bottom-k singular values.	Uses the smallest singular directions instead of the largest ones.
copy	$N_{ij} = \alpha W_{(i \bmod d_{\text{old}}),(j \bmod d_{\text{old}})}$	Tiles the old matrix and crops the repeated block to $\Delta \times \Delta$.
net2widenet	$j \mapsto j \bmod H_{\text{old}}$, with outgoing weights divided by replication count.	FFN hidden-width growth by deterministic function-preserving neuron replication.
dense-rand	$N = s\Delta^{-1/2}G, \quad G_{ij} \sim \mathcal{N}(0,1)$	Dense Gaussian new block scaled by the median singular scale.
diag-rand	$N = s\text{diag}(g), \quad g_i \sim \mathcal{N}(0,1)$	Random diagonal-only new block.
diag-const	$N = sI_\Delta$	Constant diagonal new block.
spectral	$N = U\text{diag}(\tilde{\sigma})V^\top$ $\tilde{\sigma}_i = \alpha\sigma_{\text{lo}}(\sigma_{\text{hi}}/\sigma_{\text{lo}})^{(i-1)/(\Delta-1)}$ $\sigma_{\text{lo}} = P_{10}\big(\sigma(W)\big), \quad \sigma_{\text{hi}} = P_{90}\big(\sigma(W)\big)$	QR-orthonormal factors with a geometric singular-value profile from the old matrix spectrum.
ortho	$N = sQ, \quad Q^\top Q = I_\Delta$	Semi-orthogonal new block with median-spectrum scale.

For baseline comparison, we implemented LiGO growth algorithm in our code base. However, when we mark an experiment as convex, we mean the static, i.e., offline growth using random convex coefficients. The runs for the LiGO reimplementation are marked as LiGO-layerwise or just ligo.

A.2 Reproducibility settings

Table 7 records the shared training configuration and the LiGO control settings.

Table 7. Reproducibility settings.

Setting	Value
Shared training configuration	
Architecture	8 transformer layers; 4 attention heads; embedding width 256; residual width 256–1024; 24.3M–119.2M parameters.
Context and position	256 tokens; learned absolute positional embedding of width 256.
Batch size	256 at $d = 256$; 128 after width growth.

Setting	Value
Optimizer	Adam; $(\beta_1, \beta_2) = (0.9, 0.999)$; $\epsilon_{\text{Adam}} = 10^{-8}$; weight decay 0.
Learning rate	3×10^{-4} at $d = 256$; 1.8×10^{-4} after growth; constant within each stage; no learning-rate warmup.
Gradient clipping	Global gradient L_2 norm capped at 1.0.
Dropout	0.
Numerics and kernels	FP32 parameters, gradients, and Adam states; BF16 autocast compute; fused QKV projections; PyTorch scaled-dot-product attention.
Gradient checkpointing	Enabled for the full progressive and matched component runs; disabled during LiGO operator fitting and its post-growth control.
Output head	Adaptive softmax; cutoffs 10,000/20,000/30,000/40,000; division value 2; GPT-2 BPE vocabulary of 50,257 tokens.
Training unit	Up to 20M source tokens per shard; window stride 256.
Random seeds	NRIO progression: 42. SRIO/MRIO progressions: training 45 and archived offline expansion 42. Matched component runs: 43/44/45. LiGO control: 45.
LiGO control	
LiGO control	$d256 \rightarrow d384$; 100 offline Adam steps at learning rate 10^{-3}, batch size 256 and context 256; layerwise convex-initialized operators; old block learnable; $R\gamma$, $R\beta$ LayerNorm transformation and output-side Wo tie; orthogonality penalty 0; gate OFF. The 100 batches use the 20M-token slice at offset 20M from `owt_more_0011.npz`.

A.3 Seed variation

Table 8. Matched three-seed loss variation.

Policy	Mean	SD	Range
NRIO	4.239120	0.000261	0.000518
LRIO-LI	4.219084	0.000685	0.001320
SRIO-SD	4.216382	0.000266	0.000478
SRIO-SO	4.211135	0.000649	0.001283

A.4 Matched component comparison

This table reports a controlled experiment set that isolates the three solution components: the RIO operator, scalar warmup gate, and differential expansion policy. The observed advantages of SRIO and the differential expansion policy are clear. In this setting, however, the gate provides little additional benefit when combined with SRIO. By contrast, the focused warmup-gate comparisons show a stronger contribution for LRIO; see Figure 6.

We reimplemented LiGO in our codebase using the publicly available VITA Group implementation as a reference.[1]

Table 9. Matched component comparison.

Variant	RIO	Gate	FFN	n	Loss	Δ
NRIO	NRIO	OFF	convex	3	4.2527 ± 0.0008	+0.0000
NRIO + Gate	NRIO	ON	convex	3	4.2611 ± 0.0002	+0.0084
+SRIO	SRIO-SO	OFF	convex	3	4.2339 ± 0.0009	-0.0187
SRIO-SO + Gate	SRIO-SO	ON	convex	1	4.2335 ± 0.0000	-0.0192
LiGO (old fixed)	NRIO	OFF	convex	1	4.2606 ± 0.0000	+0.0079
LiGO (old learned)	NRIO	OFF	convex	1	4.2830 ± 0.0000	+0.0303
SRIO-SO + Gate (All three)	SRIO-SO	ON	copy	3	4.2111 ± 0.0006	-0.0415

A.5 MRIO construction

Scaling targets

Let the old residual space contain m stage regions indexed from the oldest, $s = 0$, to the youngest-old, $s = m - 1$, with widths d_s. Given a youngest-old target w_{ref} and release parameter β, the raw and normalized row-sum targets are

$$\widetilde{w}_s = 1 - (1 - w_{\text{ref}})\beta^{m-1-s}, \qquad w_s = \frac{\widetilde{w}_s}{\frac{1}{m}\sum_{k=0}^{m-1} \widetilde{w}_k}.$$

w_{ref} is the reference scale for the youngest-old stage $\widetilde{w}_{m-1}$, the most throttled stage. The normalization makes the weights a grading around one: old stages are boosted and the youngest-old is throttled.

The scaling targets are age-based and correspond to each subspace's all-ones direction. Unlike SRIO, where we directly solve a quadratic equation, we first normalize the targets and use a least-squares fit for the read- and write-side scaling targets. We then express the fitted matrix in the orthonormal basis formed by the subspaces' all-ones directions and take its polar factor. This gives $MRIO_{\text{age}}$, the closest orthogonal matrix to $MRIO_{\text{lsq}}$.

Least-squares fit

Let $G_{\text{lsq}} \in \mathbb{R}^{m \times m}$ be the region-level gain matrix, with output regions as rows and input regions as columns. Its diagonal is fixed to one; for $s \neq t$, let $x_{st} = \left(G_{\text{lsq}}\right)_{st}$ be the off-diagonal entries to fit. The fitted targets are

[1] https://github.com/VITA-Group/LiGO

$$\sum_{t \neq s} x_{st} \simeq w_s - 1, \qquad \sum_{s \neq t} \frac{d_s}{d_t} x_{st} \simeq 0.$$

Stacking these targets as $Ax \simeq b$, the minimum-norm least-squares solution is

$$x = A^{+} b.$$

The first target fits the write-side row sum for each old region; the second fits each read-side dimension-weighted column sum to one. Expressing the fit in the orthonormal basis formed by the subspaces' all-ones directions gives

$$\left(MRIO_{\text{lsq}}\right)_{st} = \left(G_{\text{lsq}}\right)_{st} \sqrt{\frac{d_s}{d_t}}.$$

Polar factor

If $MRIO_{\text{lsq}} = U \Sigma V^{\top}$, its nearest proper orthogonal polar factor is

$$MRIO_{\text{age}} = U V^{\top}.$$

The final column of U is sign-flipped before recomputing $MRIO_{\text{age}}$ when required to enforce $\det\left(MRIO_{\text{age}}\right) = +1$.

Dense expansion and composition

Finally, expand $MRIO_{\text{age}}$ to its full-width representation $MRIO_{\text{age}}^{(D)}$, starting from $MRIO_{\text{age}}^{(D)} = I_D$:

$$g_{st} = \left(MRIO_{\text{age}}\right)_{st} \sqrt{\frac{d_t}{d_s}}, \qquad \left(MRIO_{\text{age}}^{(D)}\right)_{ij} = \delta_{ij} + \frac{g_{st} - \delta_{st}}{d_t}, \quad i \in s,\ j \in t.$$

The newly grown region remains identity under $MRIO_{\text{age}}^{(D)}$. The final MRIO composes this age rotation with the current SRIO:

$$\text{MRIO_R} = MRIO_{\text{age}}^{(D)} \text{ SRIO_R}.$$

In our experiments we used w_{ref} values of 0.3 and 0.64 in two different experiment runs. The previous stage's SRIO R_new can guide the choice of w_{ref}. We propose that MRIO aging can be useful when growth to the current stage is early, when the previous stage's R_new was too high, or both.

A.6 Orthogonality-penalty comparison

The left panel is the aggregate across many LRIO policy types and configurations, and the right panel contains pairs with configurations and expansion policies matched.

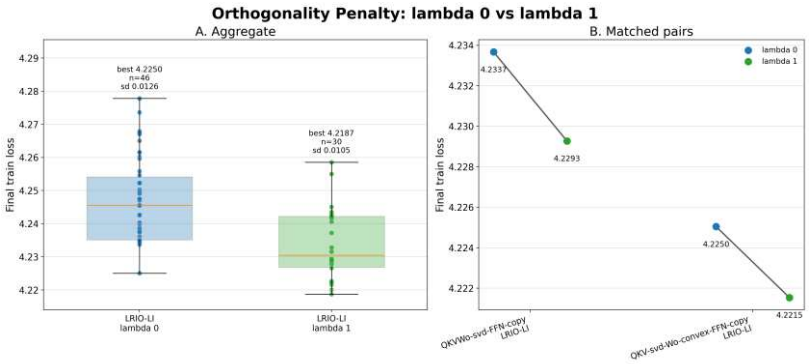


Figure 8. Orthogonality-penalty ON/OFF comparison. Left: aggregate LRIO runs. Right: matched dense LRIO pairs.

A.7 LayerNorm γ and β transformation: gradient ratios

Table 10 reports paired untransformed and $R\gamma, R\beta$ probes. Each row holds the source checkpoint, growth boundary, operator, and ϵ fixed; the SRIO and MRIO rows are not a direct operator comparison. The ratio is the median across the measured matrix and attention-head blocks after one backward pass on the same 256-token window.

Table 10. Matched new/old gradient RMS by LayerNorm γ and β transformation and ϵ.

Operator	Growth	ϵ	Untransformed	$R\gamma, R\beta$
MRIO	$d_{384} \rightarrow d_{512}$	0.01	0.275	0.577
MRIO	$d_{512} \rightarrow d_{768}$	0.01	0.513	5.715
MRIO	$d_{512} \rightarrow d_{768}$	0.17	0.517	6.888
MRIO	$d_{512} \rightarrow d_{768}$	0.50	0.525	2.095
MRIO	$d_{512} \rightarrow d_{768}$	0.95	0.529	0.583
MRIO	$d_{768} \rightarrow d_{1024}$	0.01	0.524	1.679
SRIO-SO	$d_{256} \rightarrow d_{384}$	0.01	0.202	0.602
SRIO-SO	$d_{384} \rightarrow d_{512}$	0.01	0.276	1.133
SRIO-SO	$d_{512} \rightarrow d_{768}$	0.01	0.519	1.930
SRIO-SO	$d_{512} \rightarrow d_{768}$	0.17	0.518	2.894
SRIO-SO	$d_{512} \rightarrow d_{768}$	0.50	0.514	3.917
SRIO-SO	$d_{512} \rightarrow d_{768}$	0.95	0.515	0.667
SRIO-SO	$d_{768} \rightarrow d_{1024}$	0.01	0.232	0.456

A.8 Growth Timing and Modeled Cost

Total shards: 60. Figure 9 shows that growing early reaches a lower loss than cold training at full model size. Shading highlights the measured crossing interval.

Figure 10 shows a modeling of the cost of training at full model size and using five-stage growth from d_{256} to d_{1024}. The cost saving increases, then becomes constant after reaching full size.

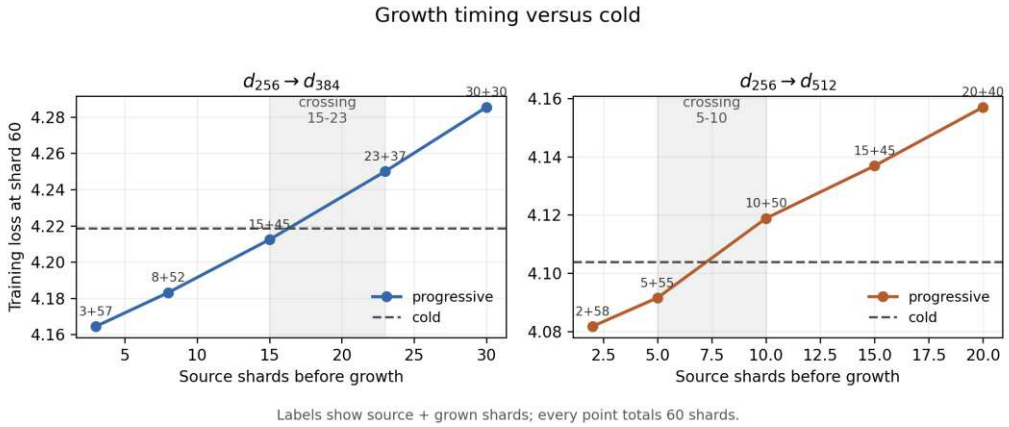


Figure 9. Early vs late growth loss comparison against cold.



Figure 10. Modeled cost and cost saving from progressive training using five growth stages.

A.9 EPS choice and output residual checks

Table 11. SRIO mean-mode row-sum gains by ϵ and growth ratio.

d_{old}/d_{grow}	a	ϵ	R_old	R_new	RT_old	RT_new
1.000	1.000	0.500	0.500	1.323	1.323	0.500
2.000	1.414	0.500	0.724	1.397	1.173	0.500
3.000	1.732	0.500	0.809	1.427	1.118	0.500
4.000	2.000	0.350	0.803	1.557	1.104	0.350
8.000	2.828	0.170	0.861	1.750	1.059	0.170
16.000	4.000	0.100	0.921	1.851	1.030	0.100

We recommend measuring residual and gradient RMS ratios post-fold at the first gradient step and at the end of the stage and monitoring them throughout the progressive expansion stages.

In the evaluated d512 to d576 growth, lower ϵ consistently reduced the final loss.

Residual and gradient RMS ratios stayed elevated with aggressive lower epsilon values, as shown in Appendix Table 14 and Appendix Table 15.

Based on these observations, we have suggested epsilon values in Table 11 for a few growth ratios. Implementation-specific measurements and studies and the stage lengths can impact this choice.

Table 12 compares each region's γ RMS, relative to the original region, at its first checkpoint and the final d1024 checkpoint. Movement toward 1 indicates more uniform region scaling.

Table 12. LayerNorm γ region multipliers at expansion and d1024.

operator	region	norm1 start	norm1 d1024	norm2 start	norm2 d1024
SRIO-SO	exp1 (d384)	5.474	0.852	2.675	0.874
SRIO-SO	exp2 (d512)	13.089	1.202	3.764	1.082
SRIO-SO	exp3 (d768)	2.766	1.205	2.708	1.326
SRIO-SO	exp4 (d1024)	1.574	1.574	1.085	1.085
MRIO	exp1 (d384)	5.474	1.557	2.675	1.734
MRIO	exp2 (d512)	11.559	1.693	3.217	1.584
MRIO	exp3 (d768)	4.898	1.758	3.899	1.672
MRIO	exp4 (d1024)	2.202	2.202	1.606	1.606

Table 13 compares post-LayerNorm old/new RMS immediately after folding and at the end of d384. The post-fold checkpoint uses SRIO-SO with $\epsilon = 0.95$; values are layer-averaged 5% trimmed means from one 256-token LAMBADA window.

Table 13. Post-fold and end-stage residual reads for d256 to d384.

Condition	Site	old [0:256]	new [256:384]	new/old
post-fold	norm1	0.8359	0.2033	0.243
post-fold	norm2	1.0616	0.2727	0.257
end-d384	norm1	0.7035	0.6889	0.979
end-d384	norm2	1.0821	0.8712	0.805

Table 14 reports post-fold old/new residual RMS and new/old ratios across growth sizes and ϵ values. The final column gives the closed-form R_new/R_old contrast for comparison.

Table 14. Post-fold residual reads across ϵ and growth size.

grow	new dim	ϵ	site	old residual	new residual	measured new/old	R_new /R_old
64	320	0.01	norm1	0.531	0.406	0.765	2.642
64	320	0.01	norm2	0.722	0.593	0.821	2.642
64	320	0.25	norm1	0.600	0.375	0.625	2.123
64	320	0.25	norm2	0.798	0.542	0.680	2.123
64	320	0.50	norm1	0.666	0.336	0.504	1.691
64	320	0.50	norm2	0.869	0.480	0.552	1.691
64	320	0.75	norm1	0.727	0.292	0.402	1.325
64	320	0.75	norm2	0.932	0.408	0.437	1.325
64	320	0.95	norm1	0.770	0.253	0.328	1.063
64	320	0.95	norm2	0.976	0.342	0.351	1.063
128	384	0.01	norm1	0.357	0.309	0.865	3.928
128	384	0.01	norm2	0.562	0.424	0.755	3.928
128	384	0.25	norm1	0.489	0.298	0.609	2.686
128	384	0.25	norm2	0.702	0.411	0.585	2.686
128	384	0.50	norm1	0.619	0.268	0.433	1.929
128	384	0.50	norm2	0.840	0.370	0.440	1.929
128	384	0.75	norm1	0.743	0.233	0.314	1.408
128	384	0.75	norm2	0.968	0.319	0.330	1.408
128	384	0.95	norm1	0.836	0.203	0.243	1.076
128	384	0.95	norm2	1.062	0.273	0.257	1.076
256	512	0.01	norm1	0.074	0.396	5.370	141.418
256	512	0.01	norm2	0.160	0.449	2.809	141.418
256	512	0.25	norm1	0.217	0.455	2.099	5.568
256	512	0.25	norm2	0.439	0.531	1.210	5.568
256	512	0.50	norm1	0.454	0.398	0.875	2.646
256	512	0.50	norm2	0.694	0.485	0.698	2.646
256	512	0.75	norm1	0.708	0.297	0.420	1.599
256	512	0.75	norm2	0.954	0.376	0.394	1.599
256	512	0.95	norm1	0.910	0.239	0.263	1.103
256	512	0.95	norm2	1.156	0.305	0.264	1.103

Table 15 reports post-LayerNorm new/old residual ratios at layers 0, 5, 6, and 7 across growth sizes and ϵ values.

Table 15. Layerwise post-fold new/old residual ratios.

grow	new dim	ϵ	site	L0	L5	L6	L7
64	320	0.01	norm1	0.273	0.915	1.215	1.546
64	320	0.01	norm2	0.190	1.008	1.512	1.323
64	320	0.25	norm1	0.180	0.752	1.013	1.273
64	320	0.25	norm2	0.152	0.826	1.250	1.108
64	320	0.50	norm1	0.117	0.610	0.826	1.030
64	320	0.50	norm2	0.128	0.662	1.005	0.907
64	320	0.75	norm1	0.077	0.488	0.660	0.817
64	320	0.75	norm2	0.113	0.518	0.784	0.716
64	320	0.95	norm1	0.059	0.399	0.535	0.663
64	320	0.95	norm2	0.102	0.413	0.618	0.568
128	384	0.01	norm1	0.636	1.068	1.148	1.497
128	384	0.01	norm2	0.201	0.987	1.138	1.183
128	384	0.25	norm1	0.272	0.779	0.905	1.057
128	384	0.25	norm2	0.136	0.761	0.907	0.897
128	384	0.50	norm1	0.144	0.566	0.690	0.755
128	384	0.50	norm2	0.102	0.567	0.695	0.668
128	384	0.75	norm1	0.085	0.415	0.518	0.550
128	384	0.75	norm2	0.087	0.418	0.524	0.492
128	384	0.95	norm1	0.060	0.322	0.404	0.427
128	384	0.95	norm2	0.083	0.322	0.403	0.376
256	512	0.01	norm1	0.292	15.830	36.924	33.897
256	512	0.01	norm2	0.589	5.026	3.343	4.984
256	512	0.25	norm1	1.595	2.804	2.951	4.758
256	512	0.25	norm2	0.185	2.041	1.811	2.535
256	512	0.50	norm1	0.279	1.210	1.351	1.594
256	512	0.50	norm2	0.099	1.076	1.023	1.145
256	512	0.75	norm1	0.104	0.574	0.679	0.716
256	512	0.75	norm2	0.076	0.550	0.582	0.577
256	512	0.95	norm1	0.061	0.351	0.435	0.440
256	512	0.95	norm2	0.083	0.345	0.383	0.366

A.10 Scalar warmup gate comparisons

Table 16 contains the data corresponding to Figure 6 in the main paper. It shows that `new_dim_alpha_gate` is beneficial in all but a few matched policy configurations, including the NRIO LiGO-layerwise configuration.

Table 16. Matched comparison – `new_dim_alpha_gate`: ON/OFF.

Matched cell	Integration op	Off train	On train	Δ train
SVD top k128	LRIO-LI	4.2668	4.2361	-0.0306
copy	LRIO-LI	4.2647	4.2349	-0.0297
ortho	LRIO-LI	4.2616	4.2375	-0.0241
SVD top k15	LRIO-LI	4.2699	4.2491	-0.0208
dense rand	LRIO-LI	4.2695	4.2500	-0.0195
svd bottom k15	LRIO-LI	4.2711	4.2547	-0.0165
dense rand	NRIO	4.2721	4.2602	-0.0119
svd quad fold k128	LRIO-LI	4.2583	4.2503	-0.0080
diag rand	LRIO-LI	4.2723	4.2675	-0.0048
SVD top k128	NRIO	4.2482	4.2456	-0.0026
LiGO layerwise	LRIO-LI	4.2564	4.2540	-0.0023
ortho	NRIO	4.2497	4.2475	-0.0022
convex residual	SRIO-SO	4.2123	4.2104	-0.0019
convex γ only	SRIO-SO	4.2123	4.2105	-0.0018
copy	NRIO	4.2470	4.2455	-0.0016
base residual	SRIO-SO	4.2115	4.2131	+0.0016
LiGO layerwise	NRIO	4.2567	4.2606	+0.0039
centroid	NRIO	4.2561	4.2608	+0.0047
diag rand	NRIO	4.2670	4.2736	+0.0066

Enabled lower in 15 of 19 pairs; mean Δ train -0.0085.

Table 17 isolates the NRIO matched pairs from Table 16. Enabling `new_dim_alpha_gate` lowers training loss in four of seven matched policy cells; LiGO-layerwise is one of the exceptions.

Table 17. NRIO matched comparison – `new_dim_alpha_gate`: ON/OFF.

Policy	$n_{\text{off}}/n_{\text{on}}$	Off train	On train	Δ train
dense rand	1/1	4.2721	4.2602	-0.0119
SVD top k128	2/1	4.2482	4.2456	-0.0026
ortho	1/1	4.2497	4.2475	-0.0022
copy	1/1	4.2470	4.2455	-0.0016
LiGO layerwise	2/2	4.2567	4.2606	+0.0039
centroid	1/3	4.2561	4.2608	+0.0047
diag rand	1/1	4.2670	4.2736	+0.0066

Enabled lower in 4 of 7 NRIO matched cells.

A.11 Matched FFN expansion policies

Table 18 contains the data shown in Figure 7. The FFN expansion policy is varied while the attention expansion policy, RIO configuration, `new_dim_alpha_gate`, and other training settings are fixed within each matched set. Loss differences are measured from the lowest loss in each matched set; lower is better.

Table 18. Matched FFN expansion-policy loss deltas.

FFN policy	Runs	Matched sets	Mean Δ	Std Δ
`fc1:copy / fc2:copy`	2	2	0.0000	0.0000
`fc1:svd-top:k256 / fc2:svd-top:k128`	1	1	0.0006	0.0000
`fc1:copy:k128 / fc2:copy:k128`	4	1	0.0007	0.0007
`fc1:svd-paired:k128 / fc2:svd-top:k128`	1	1	0.0013	0.0000
`fc1:svd-top:k128 / fc2:svd-top:k128`	2	1	0.0025	0.0027
`fc1:svd-top:k256 / fc2:svd-top:k1`	1	1	0.0050	0.0000
`fc1:svd-top:k128 / fc2:convex:k128`	1	1	0.0062	0.0000
`fc1:svd-top:k128 / fc2:svd-top:k1`	1	1	0.0067	0.0000
`fc1:svd-bottom:k8 / fc2:svd-top:k8`	2	1	0.0135	0.0002
`fc1:convex / fc2:convex`	2	2	0.0239	0.0013
`fc1:net2wider:k128 / fc2:net2wider:k128`	1	1	0.0313	0.0000

References

Bachlechner, Thomas, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. 2020. “ReZero Is All You Need: Fast Convergence at Large Depth.” https://arxiv.org/abs/2003.04887.

Bansal, Nitin, Xiaohan Chen, and Zhangyang Wang. 2018. “Can We Gain More from Orthogonality Regularizations in Training Deep Networks?” In *Advances in Neural Information Processing Systems*. Vol. 31. https://arxiv.org/abs/1810.09102.

Chen, Cheng, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. 2022. “bert2BERT: Towards Reusable Pretrained Language Models.” In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2134–48. https://doi.org/10.18653/v1/2022.acl-long.151.

Chen, Tianqi, Ian Goodfellow, and Jonathon Shlens. 2016. “Net2Net: Accelerating Learning via Knowledge Transfer.” In *International Conference on Learning Representations*. https://arxiv.org/abs/1511.05641.

De, Soham, and Samuel L. Smith. 2020. “Batch Normalization Biases Residual Blocks Towards the Identity Function in Deep Networks.” In *Advances in Neural Information Processing Systems*. Vol. 33. https://arxiv.org/abs/2002.10444.

Du, Wenyu, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. 2024. “Stacking Your Transformers: A Closer Look at Model Growth for Efficient LLM Pre-Training.” In *Advances in Neural Information Processing Systems*. Vol. 37. https://proceedings.neurips.cc/paper_files/paper/2024/hash/143ea4a156ef64f32d4d905206cf32e1-Abstract-Conference.html.

Evci, Utku, Bart van Merrienboer, Thomas Unterthiner, Fabian Pedregosa, and Max Vladymyrov. 2022. “GradMax: Growing Neural Networks Using Gradient Information.” In *International Conference on Learning Representations*. https://arxiv.org/abs/2201.05125.

Gong, Linyuan, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. “Efficient Training of BERT by Progressively Stacking.” In *Proceedings of the International Conference on Machine Learning*, 97:2337–46. https://proceedings.mlr.press/v97/gong19a.html.

Grave, Édouard, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. “Efficient Softmax Approximation for GPUs.” In *Proceedings of the 34th International Conference on Machine Learning*, 70:1302–10. Proceedings of Machine Learning Research. https://proceedings.mlr.press/v70/grave17a.html.

Gu, Xiaotao, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2021. “On the Transformer Growth for Progressive BERT Training.” In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 5174–80. https://doi.org/10.18653/v1/2021.naacl-main.406.

Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, et al. 2022. “Training Compute-Optimal Large Language Models.” In *Advances in Neural Information Processing Systems*. Vol. 35. https://arxiv.org/abs/2203.15556.

Jastrzębski, Stanisław, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. 2020. “The Break-Even Point on Optimization Trajectories of Deep Neural Networks.” In *International Conference on Learning Representations*. https://arxiv.org/abs/2002.09572.

Karras, Tero, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. “Progressive Growing of GANs for Improved Quality, Stability, and Variation.” In *International Conference on Learning Representations*. https://arxiv.org/abs/1710.10196.

Martin, Charles H., Tongsu Peng, and Michael W. Mahoney. 2021. “Predicting Trends in the Quality of State-of-the-Art Neural Networks Without Access to Training or Testing Data.” *Nature Communications* 12 (1): 4122. https://arxiv.org/abs/2002.06716.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. “Language Models Are Unsupervised Multitask Learners.” OpenAI.

Roy, Olivier, and Martin Vetterli. 2007. “The Effective Rank: A Measure of Effective Dimensionality.” In *Proceedings of the European Signal Processing Conference*.

Samragh, Mohammad, Iman Mirzadeh, Keivan Alizadeh Vahid, Fartash Faghri, Minsik Cho, Moin Nabi, Devang Naik, and Mehrdad Farajtabar. 2024. “Scaling Smart: Accelerating Large Language Model Pre-Training with Small Model Initialization.” https://arxiv.org/abs/2409.12903.

Touvron, Hugo, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. 2021. “Going Deeper with Image Transformers.” In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. https://arxiv.org/abs/2103.17239.

Unterthiner, Thomas, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. 2020. “Predicting Neural Network Accuracy from Weights.” https://arxiv.org/abs/2002.11448.

Wang, Peihao, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David D. Cox, Zhangyang Wang, and Yoon Kim. 2023. “Learning to Grow Pretrained Models for Efficient Transformer Training.” In *International Conference on Learning Representations*. https://openreview.net/forum?id=cDYRS5iZ16f.

Wang, Yite, Jiahao Su, Hanlin Lu, Cong Xie, Tianyi Liu, Jianbo Yuan, Haibin Lin, Ruoyu Sun, and Hongxia Yang. 2024. “LEMON: Lossless Model Expansion.” In *International Conference on Learning Representations*. https://openreview.net/forum?id=3Vw7DQqq7U.

Wei, Tao, Changhu Wang, Yong Rui, and Chang Wen Chen. 2016. “Network Morphism.” In *Proceedings of the International Conference on Machine Learning*. Vol. 48. https://arxiv.org/abs/1603.01670.

Yao, Yiqun, Zheng Zhang, Jing Li, and Yequan Wang. 2024. “Masked Structural Growth for 2x Faster Language Model Pre-Training.” In *International Conference on Learning Representations*. https://proceedings.iclr.cc/paper_files/paper/2024/hash/a8320b6b9d95798dc286a867c44742a1-Abstract-Conference.html.

Yunis, David, Kumar Kshitij Patel, Samuel Wheeler, Pedro Savarese, Gal Vardi, Karen Livescu, Michael Maire, and Matthew R. Walter. 2024. “Approaching Deep Learning Through the Spectral Dynamics of Weights.” https://arxiv.org/abs/2408.11804.