REHIGATE: FUNCTION-PRESERVING CAPACITY AUGMENTATION VIA HISTORY-CONDITIONED RESIDUAL CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale pre-training is expensive, motivating approaches that reuse partially trained models rather than train every target model from scratch. Function-preserving model growth offers a natural solution by expanding the backbone while keeping its function unchanged at insertion. However, forward preservation alone does not guarantee that the added capacity provides useful optimization directions. We propose **ReHiGate** (*Residual History Gating*), which keeps the backbone intact and instead introduces a history-conditioned residual controller. ReHiGate reads a sparse set of preceding-layer states and predicts token- and channel-wise gates for the attention and MLP residual branches, using cross-layer history as control rather than directly mixing it into the representation stream. Its identity-centered initialization exactly recovers the inherited computation while retaining active first-order controller directions, and insertion-time functional diagnostics show that these directions are less aligned with the trained backbone than those introduced by parameter-matched width and depth growth. We evaluate ReHiGate on 124M- and 1.5B-parameter Transformers with continuation endpoints at 10B and 100B tokens, respectively. Across both scales, ReHiGate outperforms plain continuation and parameter-matched width and depth growth at every evaluated insertion point, reducing validation loss by up to 0.044 and 0.017 nats.

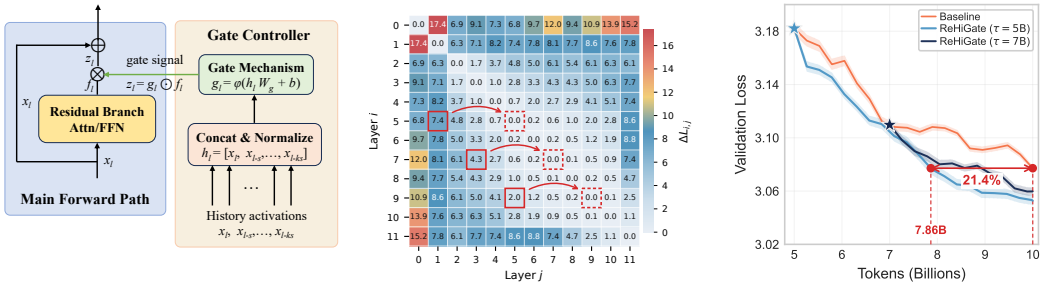
1 INTRODUCTION

Pre-training large language models is computationally expensive (Hoffmann et al., 2022; Kaplan et al., 2020; McCandlish et al., 2018). A common way to reduce this cost is to reuse partially trained models instead of training every target architecture from scratch. Staged training and model-growth methods follow this idea (Shen et al., 2022; Gong et al., 2019; Wang et al., 2023; Du et al., 2024; Yao et al., 2023). They expand a smaller or partially trained Transformer and continue optimization from the inherited state. Such reuse requires introducing new capacity without discarding what the model has already learned.

Most model-growth methods introduce this capacity through function-preserving width or depth expansion. These constructions keep the model output unchanged at insertion, but they do not guarantee that the newly added parameters provide useful optimization directions. Replicated parameters can receive highly correlated updates, while exact identity constructions can leave some new directions initially inactive (Chen et al., 2015; Evci et al., 2022). Existing methods address these issues through techniques such as symmetry-breaking perturbations and gradient-aware initialization (Chen et al., 2015; Evci et al., 2022). More generally, preserving the model function at insertion does not ensure useful update signals for the newly added parameters once training resumes. The way new capacity is introduced therefore becomes important.

We propose **ReHiGate** (*Residual History Gating*), illustrated in Figure 1(a), which introduces new capacity through a history-conditioned residual controller rather than by expanding the Transformer backbone. This gives the added parameters a different functional role from the replicated or identity-initialized structures used in conventional model growth. For each block, ReHiGate reads a sparse set of preceding-layer states and uses them to predict token- and channel-wise gates for the attention and MLP residual branches (He et al., 2016; Vaswani et al., 2017; Srivastava et al., 2015). The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) ReHiGate overview (b) Layer-swap sensitivity (c) Token saving

Figure 1: **Overview and empirical motivation of ReHiGate.** **(a)** ReHiGate conditions on a sparse set of preceding-layer activations to predict gates for the current attention and MLP residual branches. **(b)** Layer-swap sensitivity on the 124M model reveals a broad low-sensitivity middle block, indicating substantial functional redundancy with graded layer-specific variation; red boxes highlight representative stride-4 history connections. **(c)** ReHiGate reaches the baseline 10B validation loss earlier across insertion points; with insertion at 5B tokens, it matches the baseline after 7.86B tokens, saving 21.4% of training tokens.

historical states serve only as conditioning signals for residual modulation and are not directly mixed into the representation stream. This design is motivated by the observation that intermediate representations are redundant across depth, yet remain layer-specific. Layer-swap analysis at both 124M and 1.5B scales(Figure 1(b); Appendix D) shows broad low-sensitivity regions among middle layers, while sensitivity still varies across depth. This suggests that sparse preceding-layer states can retain useful cross-depth variation without repeatedly reading highly redundant neighboring representations (Jawahar et al., 2019; Tenney et al., 2019; Rogers et al., 2020).

ReHiGate is inserted without changing the inherited backbone weights, optimizer state, or training schedule. An identity-centered initialization sets the gates to one at insertion, exactly recovering the inherited computation while retaining active first-order controller directions. We further compare the insertion-time functional geometry of ReHiGate with parameter-matched width and depth growth. ReHiGate introduces functional update directions that are consistently less aligned with those of the trained backbone. Across 124M- and 1.5B-parameter Transformers, ReHiGate outperforms plain continuation and parameter-matched model growth at every evaluated insertion point, reducing validation loss by up to 0.044 and 0.017 nats, respectively.

The contributions of this work can be summarized as follows: **(1) History-conditioned residual augmentation.** We propose ReHiGate, which introduces new capacity through history-conditioned residual modulation rather than width or depth expansion. Preceding-layer states are used to control the attention and MLP residual branches without being directly mixed into the representation stream. **(2) Function-preserving insertion with active and distinct update directions.** ReHiGate exactly preserves the inherited computation at insertion while retaining active first-order controller directions. Functional diagnostics show that these directions are less aligned with the trained backbone than those introduced by parameter-matched width and depth growth. We further show that cross-layer history admits a compact low-rank representation, supporting the factorized controller design. **(3) Consistent gains across scales.** Across 124M- and 1.5B-parameter Transformers, ReHiGate consistently outperforms parameter-matched model growth over a broad range of insertion points, reducing validation loss by up to 0.044 and 0.017 nats, respectively.

2 Related Work

Model growth and trained-state continuation Model growth reuses a smaller trained model to accelerate the pre-training of a larger one. Building on Net2Net's function-preserving weight-splitting framework (Chen et al., 2015), subsequent work has extended this principle to modern Transformers through progressive stacking, width expansion, learned growth operators, and strictly function-preserving variants (Gong et al., 2019; Chen et al., 2022; Wang et al., 2023; Du et al., 2024; Yao et al., 2023), often reporting substantial pre-training savings. These approaches increase capacity

by modifying the backbone architecture itself, while additional techniques such as gradient-informed initialization (Evci et al., 2022) improve the optimization of newly introduced parameters. ReHiGate instead augments a trained model through residual modulation rather than expanding the backbone itself.

Residual pathways and cross-layer information flow Residual connections are central to modern deep networks (He et al., 2016; Vaswani et al., 2017), and a substantial body of work has explored how to control or enrich the signal flowing through them. Early gating mechanisms were introduced to stabilize optimization and shape per-layer contributions (Srivastava et al., 2015; Bachlechner et al., 2021; Wang et al., 2024; Parisotto et al., 2020), while more recent work brings gating inside attention itself (Qiu et al., 2025; Lin et al., 2025). In parallel, a growing line of methods lets each layer draw on more than the immediately preceding state through depth-weighted averaging, depth-wise cross-attention, multiple dense streams, preserved value information, or learned cross-depth connection strengths (Pagliardini et al., 2024; Heddes et al., 2025; Xiao et al., 2025; Zhou et al., 2024; Menghani et al., 2024; Zhu et al., 2024). These methods typically either condition modulation on the current representation or incorporate historical states directly into the representation pathway. ReHiGate instead treats cross-layer history as a control signal for residual modulation, rather than mixing historical states directly into the representation pathway.

3 METHOD

This section describes ReHiGate through three components: the residual modulation interface, the history-conditioned gate controller, and the function-preserving insertion procedure.

3.1 RESIDUAL MODULATION INTERFACE

Let $x_l \in \mathbb{R}^{T\times d}$ denote the residual stream entering Transformer block l, where T is the sequence length and d the hidden size. We absorb normalization and most internal projections into the branch functions, except for the attention output projection $W_{o,l}$, which is kept explicit because ReHiGate's gate is inserted between attention mixing and this projection. A standard Transformer block can then be written as

$$a_l = \mathrm{Attn}_l(x_l), \qquad u_l = x_l + a_l, \tag{1}$$

$$m_l = \mathrm{MLP}_l(u_l), \qquad x_{l+1} = u_l + m_l, \tag{2}$$

where the attention contribution a_l and the MLP contribution m_l enter the residual stream with fixed unit coefficients.

ReHiGate modulates these residual contributions with learned, input-dependent gates while leaving the branch computations otherwise unchanged. The attention gate is applied after attention mixing and before the output projection—a configuration we refer to as **post-attention placement**—while the MLP gate directly modulates the MLP residual contribution:

$$\bar{a}_l = \mathrm{AttnMix}_l(x_l), \qquad a_l = W_{o,l}(g_l^a \odot \bar{a}_l)\,, \qquad u_l = x_l + a_l, \tag{3}$$

$$m_l = \mathrm{MLP}_l(u_l), \qquad x_{l+1} = u_l + g_l^m \odot m_l. \tag{4}$$

Here $\bar{a}_l$ denotes the mixed attention output before the final projection $W_{o,l}$, and $g_l^a, g_l^m \in \mathbb{R}^{T\times d}$ are token- and channel-wise gates with $\odot$ denoting element-wise multiplication. The two branches are gated independently, allowing the controller to modulate the attention and MLP contributions separately—a choice we ablate in Section 5.4.

3.2 HISTORY-CONDITIONED GATE CONTROLLER

For each layer, ReHiGate constructs a cross-depth history input from residual-stream states available before the current branch computation and uses it only to condition the gate. We consider three history access patterns—full history, sliding-window history, and stride-s history—and adopt **stride-s history** as our main configuration, treating the others as ablations (Section 5.4). Motivated by the redundancy among nearby layers and the remaining variation across depth, striding avoids repeatedly

reading highly redundant neighboring states while retaining coverage over earlier depths. The stride-s history input is

$$h_l^{(s)} = \mathrm{Concat}_d\left(x_l,\ x_{l-s},\ x_{l-2s},\ \ldots\right) \in \mathbb{R}^{T\times k_l d}, \tag{5}$$

where only valid layer indices are kept and k_l denotes the number of valid sources used by layer l.

From $h_l^{(s)}$, ReHiGate predicts branch-specific element-wise gates:

$$g_l^b = \frac{1}{2} + \sigma\left(h_l^{(s)} W_{g,l}^b + b_{g,l}^b\right), \qquad W_{g,l}^b \in \mathbb{R}^{k_l d\times d}, \quad b_{g,l}^b \in \mathbb{R}^d, \quad b \in \{a, m\}. \tag{6}$$

The shifted-sigmoid parameterization centers the gate at one, making the controller compatible with an identity-preserving insertion, which we develop in Section 3.3.

The full projection $W_{g,l}^b$ is an unconstrained map from concatenated historical evidence to the gate space, with $k_l d^2$ parameters per branch. Our spectral analysis shows that the concatenated history is highly redundant and that its dominant variation can be summarized in a much smaller subspace (Section 5.3 and Appendix D). We therefore parameterize the history-to-gate map through a rank-r bottleneck:

$$g_l^b = \frac{1}{2} + \sigma\left(h_l^{(s)} U_l^b V_l^b + b_{g,l}^b\right), \qquad U_l^b \in \mathbb{R}^{k_l d\times r}, \quad V_l^b \in \mathbb{R}^{r\times d}, \tag{7}$$

with $W_{g,l}^b = U_l^b V_l^b$ and $\mathrm{rank}(W_{g,l}^b) \le r$. The controller therefore summarizes the cross-layer history through a learned r-dimensional subspace before producing the modulation signal.

3.3 Function-preserving insertion

We next describe how ReHiGate is inserted into a partially trained model while exactly preserving the inherited computation and retaining active first-order controller directions.

Exact identity with active first-order directions. Given a partially trained model at token budget τ, we keep the inherited backbone unchanged and initialize only the newly introduced controller. For each low-rank gate, U_l^b is initialized with an orthogonal basis, while $V_l^b = 0$ and $b_{g,l}^b = 0$. Since the shifted sigmoid is centered at one,

$$g_l^b = \frac{1}{2} + \sigma\left(h_l^{(s)} U_l^b V_l^b + b_{g,l}^b\right) = \mathbf{1}, \tag{8}$$

so ReHiGate exactly recovers the inherited backbone computation at insertion.

Exact function preservation, however, does not make the controller first-order inactive. Let $z_l^b = h_l^{(s)} U_l^b$. Because $\sigma'(0) = \frac{1}{4}$, the gradient of the loss with respect to the output factor is

$$\nabla_{V_l^b}\mathcal{L} = \frac{1}{4}\left(z_l^b\right)^\top \nabla_{g_l^b}\mathcal{L} = \frac{1}{4}\left(h_l^{(s)} U_l^b\right)^\top \nabla_{g_l^b}\mathcal{L}. \tag{9}$$

The derivation is given in Appendix A. This gradient is generically nonzero whenever the projected history and the upstream gate gradient are nonzero. Together, the exact identity condition and the nonzero first-order gradient give the following insertion property.

Proposition 1 (Identity preservation with active first-order directions) *Consider the ReHiGate controller $g_l^b = \frac{1}{2} + \sigma(h_l^{(s)} U_l^b V_l^b + b_{g,l}^b)$ with $V_l^b = 0$ and $b_{g,l}^b = 0$ at insertion. Then:*

(i) *$g_l^b \equiv \mathbf{1}$, so the augmented model exactly recovers the inherited backbone computation;*

(ii) *the gradient with respect to V_l^b is $\nabla_{V_l^b}\mathcal{L} = \frac{1}{4}(h_l^{(s)} U_l^b)^\top \nabla_{g_l^b}\mathcal{L}$, and is therefore nonzero whenever the projected history and upstream gate gradient are nondegenerate.*

Hence exact function preservation does not eliminate the controller's first-order adaptation signal.

Beyond the exact-identity setting used in our main experiments, the same parameterization also admits a controlled near-identity regime. With a nonzero perturbation scale τ_g, the expected squared residual deviation scales as $\mathcal{O}(\tau_g^2)$, while the gate-logit variance remains independent of the hidden

width under fan-in scaling. The corresponding results for both low-rank and dense controllers are given in Appendix A.

Optimizer state and controller-only warmup. Backbone optimizer moments and the learning-rate schedule are restored from the partially trained state, while the newly introduced controller starts with fresh optimizer state. We apply a short warmup only to the controller parameter group. Letting $\eta_b(t)$ denote the inherited backbone learning rate at continuation step t, ρ the controller-to-backbone learning-rate ratio, and T_g the controller warmup duration, the controller learning rate is

$$\eta_g(t) = \rho\, \eta_b(t)\, \min\left(1, \frac{t+1}{T_g+1}\right). \tag{10}$$

The backbone therefore continues under its inherited schedule, while the new controller is introduced with a short learning-rate warmup.

4 Experimental Setup

We evaluate ReHiGate as a continuation intervention on 124M- and 1.5B-parameter GPT-style Transformers. In each setting, a backbone is first trained to an intermediate token budget τ, after which the intervention is applied and training continues to a fixed endpoint under a matched post-τ token budget. For the 124M backbone, we sweep $\tau \in \{1\text{B}, \ldots, 8\text{B}\}$ and continue to 10B tokens. For the 1.5B backbone, we evaluate insertion points from 50B to 90B tokens and continue to 100B tokens. We compare ReHiGate with plain continuation, parameter-matched depth and width growth, and a large-batch continuation control. Training uses standard GPT-style settings; full backbone, data, optimizer, and continuation configurations are given in Appendix B.1.

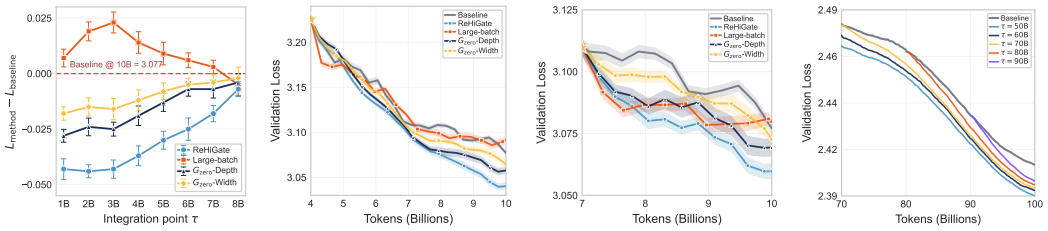
ReHiGate configuration. ReHiGate uses stride-4 history, post-attention placement, and gates on both attention and MLP residual branches, with the low-rank controller of Section 3.2 at rank $r = 256$. This adds 14.17M parameters ($\sim$11.4% of the backbone) at 124M scale and 145.6M parameters ($\sim$9.4%) at 1.5B scale. At insertion, backbone weights, AdamW moments, and the learning-rate schedule are preserved; only the controller parameters are newly introduced under the exact identity initialization of Section 3.3, followed by a short controller-only warmup. Exact controller optimizer settings are given in Appendix B.2.

Continuation baselines. All continuation methods start from the same trained state at τ and process the same remaining token budget. *Plain continuation* resumes the original backbone unchanged. *Large-batch continuation*, evaluated at 124M scale, increases the effective batch size by 25%, providing an optimization-only control without adding parameters. *Model-growth continuation* applies function-preserving growth operators in the G_{zero} style of Du et al. (2024). At 124M scale, we use G_{zero}-Depth (12→14 layers) and G_{zero}-Width ($d = 768 \to 816$), both adding approximately 14.17M parameters ($\sim$11.4% of the backbone) to match ReHiGate's added-parameter budget. At 1.5B scale, we use G_{zero}-Depth (52→57 layers) and G_{zero}-Width ($d = 1536 \to 1608$), adding 141.56M ($\sim$9.1%) and 144.87M ($\sim$9.4%) parameters, respectively, approximately matching ReHiGate's 145.6M ($\sim$9.4%) added-parameter budget. Further details and exact initialization of each variant are given in Appendix B.3.

Architecture ablations. Ablations vary one design axis at a time—controller rank, history pattern (stride and window), gate placement on the attention side, branch scope, and gate functional form—while holding the backbone and controller optimization fixed. Ablations follow the 124M 5B insertion protocol with the reference configuration above.

5 Results

We first evaluate ReHiGate across model scales and insertion points and compare it with parameter-matched model growth (§5.1). We then examine the insertion-time functional directions introduced by different augmentation schemes (§5.2). Next, we study the controller-capacity sweep and the low-rank structure of cross-layer history (§5.3). Finally, we ablate the main architectural choices in ReHiGate (§5.4).



(a) 124M final-loss improvement over baseline

(b) 124M token-aligned curves, $\tau = 4$B

(c) 124M token-aligned curves, $\tau = 7$B

(d) 1.5B token-aligned curves to 100B tokens

Figure 2: **Continuation comparison across insertion points.** ReHiGate is evaluated under matched post-τ token budgets. **(a)** Final validation loss at the 10B horizon in the 124M setting, reported as improvement over baseline ($\Delta L = L_{\text{method}} - L_{\text{baseline}}$, lower is better) across all eight insertion points $\tau \in \{1\text{B}, \ldots, 8\text{B}\}$. ReHiGate improves over the baseline at every insertion point and achieves larger gains than the parameter-matched G_{zero} growth variants. **(b,c)** Token-aligned validation loss versus processed tokens for two representative 124M insertion points, $\tau = 4$B and $\tau = 7$B. **(d)** At 1.5B scale, ReHiGate insertion between 50B and 90B tokens consistently improves the 100B-token endpoint over plain continued pre-training.

5.1 ReHiGate improves continued pre-training

Figure 2 and Table 1 summarize continuation performance across model scales and insertion points.

ReHiGate consistently outperforms continuation and matched model growth across scales. At 124M scale, ReHiGate achieves the lowest endpoint loss at every insertion point. Relative to plain continuation at 3.077, it reduces validation loss by up to 0.044 nats, reaching 3.033 at the 2B insertion point. The parameter-matched width and depth growth baselines also improve over plain continuation, showing that the added parameters are useful, but ReHiGate remains better by 0.003–0.020 nats across the full insertion sweep. The same pattern holds at 1.5B scale. ReHiGate reduces the 100B-token loss from 2.407 to 2.390–2.398, corresponding to gains of 0.009–0.017 nats, and outperforms the stronger of the two matched growth baselines by 0.007–0.011 nats at every insertion point. Together, these results show that matching the amount of added capacity does not recover the gain of ReHiGate.

The advantage becomes smaller when little continuation budget remains. The effect is most visible for late insertion, where fewer training tokens remain for the newly introduced controller. In the 124M setting, the gap between ReHiGate and the large-batch control is 0.050–0.066 nats at early insertion points but falls to 0.002 nats at $\tau = 8$B. At 1.5B scale, the gain over plain continuation similarly decreases from 0.017 nats at 50B insertion to 0.009 nats at 90B. This trend indicates that ReHiGate benefits from sufficient post-insertion optimization budget.

5.2 ReHiGate exposes less backbone-aligned functional directions

The matched-budget results in Section 5.1 show that adding the same number of parameters through width or depth growth does not recover the gain of ReHiGate. Proposition 1 establishes that ReHiGate retains active first-order controller directions at insertion, but does not determine whether these directions duplicate those already available to the trained backbone. We therefore compare the insertion-time functional directions introduced by the new parameters with those of the inherited backbone.

For each augmentation scheme, we construct functional tangent sketches for the newly introduced parameters and the inherited backbone on the same inputs. We measure their alignment using kernel CKA, overlap between their leading subspaces, and functional cosine similarity. Lower values on these three metrics indicate less alignment with the backbone. We additionally report the effective rank of the new-parameter sketch to distinguish directional novelty from a simple increase in tangent dimensionality. Further details of the functional sketches and alignment metrics are given in Appendix C.

Table 1: **Final continuation comparison across model scales.** Lower is better. All methods use matched post-insertion token budgets; growth baselines are approximately parameter-matched to ReHiGate.

124M model, 10B-token endpoint

Method	1B	2B	3B	4B	5B	6B	7B	8B
Plain continuation	3.077	3.077	3.077	3.077	3.077	3.077	3.077	3.077
G_{zero}-Width	3.059	3.062	3.061	3.065	3.069	3.072	3.073	3.075
G_{zero}-Depth	3.049	3.053	3.052	3.058	3.064	3.070	3.070	3.073
Large-batch	3.084	3.096	3.100	3.091	3.086	3.083	3.080	3.072
ReHiGate	**3.034**	**3.033**	**3.034**	**3.040**	**3.047**	**3.052**	**3.059**	**3.070**

1.5B model, 100B-token endpoint

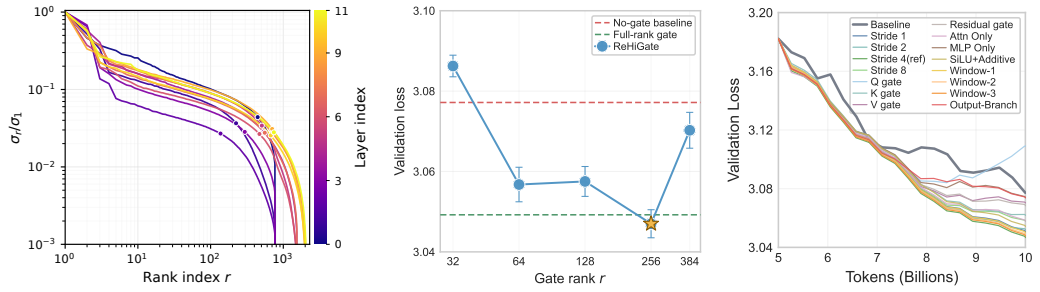
Method	50B	60B	70B	80B	90B
Plain continuation	2.407	2.407	2.407	2.407	2.407
G_{zero}-Width	2.404	2.404	2.405	2.405	2.406
G_{zero}-Depth	2.401	2.402	2.403	2.405	2.405
ReHiGate	**2.390**	**2.391**	**2.394**	**2.396**	**2.398**

Table 2: **Insertion-time functional alignment.** Lower values indicate less alignment with the inherited backbone.

Method	CKA ↓	Subspace ↓	Cosine ↓	Eff. rank
G_{zero}-Depth	0.1564	0.1347	0.2455	27.33
G_{zero}-Width	0.1605	0.1229	0.2277	24.54
ReHiGate	**0.1300**	**0.1094**	**0.1534**	23.44

ReHiGate is consistently less aligned with the inherited backbone across all three measures. Its kernel CKA is 0.1300, compared with 0.1564 and 0.1605 for depth and width growth; the same ordering appears in top-subspace overlap and functional cosine. The difference is particularly clear in functional cosine, which decreases from 0.2455 and 0.2277 for the two growth baselines to 0.1534 for ReHiGate. These results indicate that ReHiGate's active controller directions overlap less with the functional directions already available to the trained backbone.

This difference is not explained by a larger tangent space. ReHiGate has a slightly lower sketch effective rank (23.44) than both depth growth (27.33) and width growth (24.54). Its lower backbone alignment therefore reflects different directions rather than simply more directions. Together with the matched-parameter continuation results in Section 5.1, this suggests that how new capacity is introduced affects the functional directions available for continued training.



(a) spectral decay of stacked history (b) controller rank sweep (c) architectural ablations

Figure 3: **Controller rank, cross-layer spectral structure, and architectural ablations. (a)** Normalized singular-value spectra of stacked history representations. **(b)** Validation loss across controller ranks and the unconstrained projection. **(c)** Validation loss across architectural ablations.

5.3 Low-rank structure of cross-layer history

Table 3 and Figure 3(b) show that ReHiGate is not sensitive to controller rank over a broad intermediate range. Validation loss remains similar for $r \in \{64, 128, 256\}$, reaching 3.057, 3.058, and

3.047, respectively, while the unconstrained full projection reaches 3.049 with twice as many added parameters as the $r = 256$ controller. Performance degrades when the bottleneck becomes too small ($r = 32$, 3.086), while increasing the rank beyond 256 does not yield further improvement ($r = 384$, 3.070). These results indicate that the useful history signal can be captured without an unconstrained history-to-gate projection.

Figure 3(a) provides a complementary view of the history representation itself. We stack N sampled history vectors $h_l^{(s)}$ into $H_l \in \mathbb{R}^{N \times k_l d}$ and compute its singular-value spectrum. Across layers, the spectrum decays rapidly, with most of the energy concentrated in a relatively small set of leading directions. The rank required to retain 95% of the spectral energy ranges from 470 to 742, substantially below the ambient history dimension for layers that aggregate multiple states.

Table 3: **Controller-rank sweep at the 5B insertion point.** All variants use the reference ReHiGate architecture; only the controller rank changes. "Full" denotes the unconstrained history-to-gate projection.

Variant	Added params	% backbone	Val loss
Baseline	–	–	3.077
$r = 32$	1.79M	1.4%	3.086
$r = 64$	3.56M	2.9%	3.057
$r = 128$	7.10M	5.7%	3.058
$r = 256$	14.17M	11.4%	**3.047**
$r = 384$	21.25M	17.1%	3.070
Full	28.33M	22.8%	3.049

The spectral decay is consistent with the controller sweep: although the full history representation contains a longer spectral tail, continuation performance saturates at a substantially smaller controller rank. This suggests that the directions most relevant for residual modulation occupy a more compact subspace than the full history representation. The factorized controller therefore exploits redundancy already present in cross-layer history rather than relying on an unconstrained projection. Appendix D further decomposes this redundancy into per-state dimensionality and cross-lag subspace overlap, and examines its stability across training.

5.4 Architectural ablations

Holding the controller fixed at the reference rank-256 parameterization, Table 4 examines the main architectural choices in ReHiGate.

Moderate striding provides an effective history pattern. Stride-4 reaches a validation loss of 3.047 with 14.17M added parameters. Denser stride-1 and stride-2 histories do not improve performance despite using substantially more parameters, reaching 3.051 and 3.052, respectively, while the sparser stride-8 variant degrades to 3.058. Sliding-window variants are competitive but require equal or larger parameter budgets, with losses of 3.050, 3.049, and 3.048 for window sizes 1, 2, and 3. A swap-guided rule based on the layer-swap sensitivity in Figure 1(b) reaches 3.045, slightly improving over fixed stride-4 and suggesting that layer-dependent history selection can provide further gains. Together, these results support sparse cross-depth access while avoiding dense use of highly redundant neighboring states.

Post-attention placement performs best among the tested gate locations. Gating the query, key, or value projections yields losses of 3.109, 3.062, and 3.070, respectively, while gating the residual update reaches 3.069. All are worse than the post-attention placement at 3.047, with query gating even degrading below the ungated baseline. These results favor modulating the residual contribution after attention mixing rather than perturbing the internal attention computation.

Both residual branches benefit from modulation, and the simple multiplicative gate is sufficient. Restricting ReHiGate to attention-only or MLP-only gating gives losses of 3.058 and 3.074, respectively, compared with 3.047 when both branches are gated. The additive–SiLU variant reaches 3.055, while applying RMSNorm to the branch output reaches 3.074. The reference shifted-sigmoid multiplicative gate therefore provides the strongest performance among the tested gate forms.

6 Limitations

Our experiments evaluate ReHiGate at a fixed set of insertion points rather than selecting the intervention time adaptively. In addition, the newly introduced controller requires sufficient continued

Table 4: **Architectural ablations for the gate design.** All variants follow the 5B insertion protocol and change one design axis from the rank-256 reference configuration. The controller-rank sweep is reported separately in Table 3.

Variant	Changed axis	Added params	% of backbone	Val loss
Reference	stride-4, attn-post, both, mult. gate	14.17M	11.4%	*3.047*
	History pattern			
Stride 1	history stride 1	35.41M	28.6%	3.051
Stride 2	history stride 2	21.25M	17.1%	3.052
Stride 8	history stride 8	11.03M	8.9%	3.058
Window 1	dense window 1	14.17M	11.4%	3.050
Window 2	dense window 2	18.89M	15.2%	3.049
Window 3	dense window 3	23.61M	19.0%	3.048
Swap-guided	layer-swap rule	17.32M	14.0%	**3.045**
	Gate placement			
Query gate	gate query projection	14.17M	11.4%	3.109
Key gate	gate key projection	14.17M	11.4%	3.062
Value gate	gate value projection	14.17M	11.4%	3.070
Residual gate	gate residual update	14.17M	11.4%	3.069
	Branch scope			
Attention-only	scope = attention only	7.09M	5.7%	3.058
MLP-only	scope = MLP only	7.09M	5.7%	3.074
	Gate form and branch output			
Additive + SiLU	additive form, SiLU	14.17M	11.4%	3.055
Branch-output	RMSNorm on branch output	14.17M	11.4%	3.074

optimization after insertion, and the gains become smaller when the intervention is applied close to the training endpoint. Adaptive insertion and better use of limited remaining training budgets are natural directions for future work.

7 Conclusion

We introduced ReHiGate, a history-conditioned residual controller for augmenting partially trained Transformers without expanding the backbone. ReHiGate exactly preserves the inherited computation at insertion while retaining active first-order controller directions, and its functional updates are less aligned with those of the trained backbone than parameter-matched width and depth growth. Across 124M- and 1.5B-parameter Transformers, ReHiGate consistently improves continued pre-training over plain continuation and matched model-growth baselines across a broad range of insertion points. These results show that how new capacity is introduced matters for reusing prior training, and establish history-conditioned residual modulation as a complementary alternative to conventional model growth.

AI use statement

Generative AI tools were used for language editing and LaTeX formatting assistance. All AI-assisted suggestions were reviewed by the authors, who take responsibility for the final content of the work.

Reproducibility Statement

All model, training, baseline, and evaluation details required to reproduce the experiments are provided in the main text and appendices.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 7319–7328, 2021.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in artificial intelligence*, pp. 1352–1361. PMLR, 2021.

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

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

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient llm pre-training. volume 37, pp. 10491–10540, 2024.

Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Max Vladymyrov, and Fabian Pedregosa. Gradmax: Growing neural networks using gradient information. *arXiv preprint arXiv:2201.05125*, 2022.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. Efficient training of bert by progressively stacking. In *International conference on machine learning*, pp. 2337–2346. PMLR, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Mike Heddes, Adel Javanmard, Kyriakos Axiotis, Gang Fu, MohammadHossein Bateni, and Vahab Mirrokni. Deepcrossattention: Supercharging transformer residual connections. *arXiv preprint arXiv:2502.06785*, 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 3651–3657, 2019.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. 2025.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Gaurav Menghani, Ravi Kumar, and Sanjiv Kumar. Laurel: Learned augmented residual layer. *arXiv preprint arXiv:2411.07501*, 2024.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Guilherme Penedo, Hynek Kydlicek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

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

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the association for computational linguistics*, 8:842–866, 2020.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *Proceedings of the European Signal Processing Conference*, pp. 606–610, 2007.

Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. Staged training for transformer language models. In *International Conference on Machine Learning (ICML)*, 2022.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4593–4601, 2019.

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

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. 2023.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Muddformer: Breaking residual bottlenecks in transformers via multiway dynamic dense connections. 2025.

Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. Masked structural growth for 2x faster language model pre-training. *arXiv preprint arXiv:2305.02869*, 2023.

Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, and Zhenzhong Lan. Value residual learning for alleviating attention concentration in transformers. 2024.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. *arXiv preprint arXiv:2409.19606*, 2024.

A Insertion Properties of ReHiGate

This appendix provides the derivation underlying Proposition 1. The main experiments use exact identity initialization, so we first show that ReHiGate can preserve the inherited backbone function while retaining active first-order controller directions. We then consider a more general near-identity initialization and show that its perturbation scale can be controlled independently of model width.

A.1 SETUP

For a branch $b \in \{a, m\}$ at layer l, let $r_l^b(x_l) \in \mathbb{R}^d$ denote its ungated residual contribution at one token position. The original residual update is

$$x_{l+1} = x_l + r_l^b(x_l), \tag{11}$$

whereas ReHiGate introduces a coordinate-wise gate,

$$x_{l+1} = x_l + g_l^b(h_l^{(s)}) \odot r_l^b(x_l). \tag{12}$$

For the low-rank controller used in the main experiments,

$$g_l^b = \frac{1}{2} + \sigma(a_l^b), \qquad a_l^b = h_l^{(s)} U_l^b V_l^b + b_{g,l}^b, \tag{13}$$

with $U_l^b \in \mathbb{R}^{k_l d \times r}$ and $V_l^b \in \mathbb{R}^{r \times d}$. For clarity, the derivations below consider a single token position and treat $h_l^{(s)} \in \mathbb{R}^{k_l d}$ as a row vector, consistent with the matrix convention in the main text.

We write

$$z_l^b := h_l^{(s)} U_l^b \in \mathbb{R}^r \tag{14}$$

for the projected history representation, so that $a_l^b = z_l^b V_l^b + b_{g,l}^b$.

A.2 EXACT IDENTITY PRESERVATION

The main experiments initialize U_l^b with an orthogonal basis and set $V_l^b = 0$ and $b_{g,l}^b = 0$. The resulting gate logits are therefore exactly zero:

$$a_l^b = h_l^{(s)} U_l^b V_l^b + b_{g,l}^b = 0. \tag{15}$$

Because the shifted sigmoid is centered at one,

$$g_l^b = \frac{1}{2} + \sigma(0) = \mathbf{1}. \tag{16}$$

The gated residual update consequently reduces to

$$x_{l+1} = x_l + r_l^b(x_l), \tag{17}$$

which is exactly the original ungated computation.

Applying the same initialization to every gated attention and MLP branch therefore makes the complete augmented network identical to the inherited backbone at the moment of insertion. In particular, ReHiGate introduces no forward-function perturbation before continued optimization begins.

A.3 FIRST-ORDER ACTIVITY AT EXACT IDENTITY

Exact preservation of the forward function does not make the entire controller first-order inactive. Since $a_l^b = z_l^b V_l^b + b_{g,l}^b$, the gradient with respect to the output factor is

$$\nabla_{V_l^b} \mathcal{L} = (z_l^b)^\top \nabla_{a_l^b} \mathcal{L}. \tag{18}$$

By the chain rule,

$$\nabla_{a_l^b} \mathcal{L} = \frac{\partial g_l^b}{\partial a_l^b} \odot \nabla_{g_l^b} \mathcal{L}. \tag{19}$$

At the exact identity point, $a_l^b = 0$ and $\sigma'(0) = \frac{1}{4}$. Hence

$$\nabla_{V_l^b} \mathcal{L} = \frac{1}{4} (z_l^b)^\top \nabla_{g_l^b} \mathcal{L} \tag{20}$$

$$= \frac{1}{4} \left(h_l^{(s)} U_l^b \right)^\top \nabla_{g_l^b} \mathcal{L}. \tag{21}$$

Equation (21) is nonzero whenever both the projected history $h_l^{(s)} U_l^b$ and the upstream gate gradient $\nabla_{g_l^b} \mathcal{L}$ are nonzero. Thus the exact identity initialization retains active first-order optimization directions through V_l^b.

The history factor U_l^b behaves differently at the exact identity point. Its gradient is

$$\nabla_{U_l^b} \mathcal{L} = \left(h_l^{(s)}\right)^\top \left(\nabla_{a_l^b} \mathcal{L} (V_l^b)^\top\right). \tag{22}$$

Since $V_l^b = 0$ at insertion,

$$\nabla_{U_l^b} \mathcal{L} = 0. \tag{23}$$

The first optimization step therefore acts through the output factor V_l^b. Once V_l^b moves away from zero, gradients can also propagate to U_l^b, allowing the history subspace itself to adapt.

Together, the results above establish the two properties stated in Proposition 1: ReHiGate exactly preserves the inherited backbone function at insertion while retaining nonzero first-order adaptation directions in the newly introduced controller.

A.4 Controlled Near-Identity Initialization

The exact identity initialization used in the main experiments is the special case $V_l^b = 0$. For completeness, we consider a more general near-identity family in which U_l^b remains orthogonally initialized while

$$(V_l^b)_{ki} \sim \mathcal{N}\left(0, \frac{\tau_g^2}{r}\right), \qquad b_{g,l}^b = 0, \tag{24}$$

where $\tau_g \geq 0$ controls the initialization scale.

Assume that the projected history has bounded average second moment,

$$\frac{1}{r} \mathbb{E} \left\| z_l^b \right\|_2^2 \leq C_z, \tag{25}$$

with C_z independent of the hidden width d. Conditioned on z_l^b, each gate logit satisfies

$$\mathbb{E}\left[\left((a_l^b)_i\right)^2 \mid z_l^b\right] = \frac{\tau_g^2}{r} \left\| z_l^b \right\|_2^2. \tag{26}$$

Taking expectation therefore gives

$$\mathbb{E}\left[\left((a_l^b)_i\right)^2\right] \leq C_z \tau_g^2 = \mathcal{O}(\tau_g^2), \tag{27}$$

so the scale of the gate logits does not grow with the hidden width.

The shifted sigmoid is globally $1/4$-Lipschitz around its centered value, so

$$\left|(g_l^b)_i - 1\right| \leq \frac{1}{4} \left|(a_l^b)_i\right|. \tag{28}$$

Assume additionally that the projected history and residual branch satisfy the bounded mixed-moment condition

$$\frac{1}{rd} \mathbb{E}\left[\left\| z_l^b \right\|_2^2 \left\| r_l^b(x_l) \right\|_2^2\right] \leq C_{zr} R^2, \tag{29}$$

where C_{zr} is independent of d. Averaging over the initialization of V_l^b then yields

$$\frac{1}{d} \mathbb{E} \left\| (g_l^b - \mathbf{1}) \odot r_l^b(x_l) \right\|_2^2 = \mathcal{O}(\tau_g^2 R^2). \tag{30}$$

Hence τ_g controls a width-stable departure from the inherited residual computation. The main initialization is recovered at $\tau_g = 0$, where the perturbation vanishes exactly.

A.5 Dense-Gate Extension

The same near-identity argument applies to the dense controller

$$a_l^b = h_l^{(s)} W_{g,l}^b + b_{g,l}^b, \qquad W_{g,l}^b \in \mathbb{R}^{k_l d \times d}. \tag{31}$$

Using the fan-in scaling

$$(W_{g,l}^b)_{ji} \sim \mathcal{N}\left(0, \frac{\tau_g^2}{k_l d}\right), \qquad b_{g,l}^b = 0, \tag{32}$$

and assuming $\mathbb{E}\|h_l^{(s)}\|_2^2/(k_l d)$ remains bounded independently of d, each gate logit again has second moment $\mathcal{O}(\tau_g^2)$.

Under the analogous bounded mixed-moment condition for $h_l^{(s)}$ and $r_l^b(x_l)$, the expected squared residual perturbation also scales as $\mathcal{O}(\tau_g^2 R^2)$. At $\tau_g = 0$, $W_{g,l}^b = 0$ and $b_{g,l}^b = 0$, giving exact identity initialization.

A.6 Summary

The exact-identity initialization used in our main experiments has two key properties. First, setting $V_l^b = 0$ and $b_{g,l}^b = 0$ makes every gate exactly one, so ReHiGate preserves the inherited backbone function at insertion. Second, the output factor V_l^b retains nonzero first-order gradients at this same point, so exact function preservation does not eliminate all adaptation directions of the newly introduced controller. After V_l^b moves away from zero, gradients also reach U_l^b, allowing the history subspace itself to adapt.

More generally, a nonzero perturbation scale τ_g defines a near-identity initialization whose gate-logit scale and expected squared residual deviation remain controlled independently of the hidden width. The dense controller admits the same corresponding extension.

B Experimental Details

This appendix provides the full training, continuation, baseline, and ablation configurations used in the experiments.

B.1 Training and Evaluation Protocol

We use GPT-style Transformer backbones at two scales. The 124M experiments are trained on a 10B-token subset of FineWeb, while the 1.5B experiments are trained on a 100B-token subset of FineWeb (Penedo et al., 2024). Both settings use the GPT-2 BPE tokenizer (Radford et al., 2019), sequence length 1024, AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1, and gradient clipping at 1.0. Table 5 summarizes the remaining scale-specific settings.

For continuation experiments, all methods start from the same trained state at insertion budget τ and process the same remaining token budget to the shared endpoint. For the 124M model, we sweep $\tau \in \{1\text{B}, \ldots, 8\text{B}\}$ and evaluate all runs at 10B tokens. For the 1.5B model, we use $\tau \in \{50\text{B}, 60\text{B}, 70\text{B}, 80\text{B}, 90\text{B}\}$ and evaluate all runs at 100B tokens.

Inherited model parameters, optimizer moments, data-loader state, and learning-rate schedule are restored from the trained state at τ. For methods that introduce new parameters, only those parameters receive fresh AdamW state.

B.2 ReHiGate Configuration

Table 6 reports the ReHiGate configuration used at each model scale. Both settings use the same structural design—stride-4 history, post-attention modulation, gates on both attention and MLP residual branches, and a rank-256 controller—while the controller learning rate and warmup are adjusted by scale.

Table 5: **Training configuration by model scale.**

Setting	124M	1.5B
Layers	12	52
Hidden size	768	1536
Attention heads	12	12
Vocabulary	50,257	50,257
Sequence length	1024	1024
Dataset	FineWeb10B	FineWeb100B
Training endpoint	10B tokens	100B tokens
Global batch	256 sequences	512 sequences
Tokens / step	262K	524K
Optimizer	AdamW	AdamW
AdamW betas	0.9, 0.95	0.9, 0.95
Weight decay	0.1	0.1
Gradient clipping	1.0	1.0
Base LR	4×10^{-4}	1×10^{-4}
Schedule	cosine decay	warmdown over final 50K steps

Table 6: **ReHiGate configuration by model scale.**

Setting	124M	1.5B
History pattern	stride-4	stride-4
Gate placement	post-attention	post-attention
Gate scope	attention + MLP	attention + MLP
Controller rank	256	256
Initialization	exact identity	exact identity
Added parameters	14.17M	145.6M
Added %	11.4%	9.4%
Controller LR	1×10^{-4}	5×10^{-5}
Controller warmup	400 steps	3000 steps

At insertion, U_l^b is orthogonally initialized while $V_l^b = 0$ and $b_{g,l}^b = 0$, giving the exact identity initialization described in Section 3.3. Backbone parameters and optimizer state are inherited unchanged, whereas the newly introduced controller parameters start with fresh AdamW state.

B.3 G_{zero} Model-Growth Controls

We use the G_{zero} construction of Du et al. (2024) as the function-preserving model-growth control. Unlike learned growth operators designed for large changes in model size, G_{zero} allows the added parameter budget to be adjusted directly, making it suitable for comparison with ReHiGate at approximately matched parameter counts.

At 124M scale, we use G_{zero}-Depth (12→14 layers) and G_{zero}-Width ($d = 768 \to 816$), both adding approximately 14.17M parameters. At 1.5B scale, we use G_{zero}-Depth (52→57 layers) and G_{zero}-Width ($d = 1536 \to 1608$), adding 141.56M and 144.87M parameters, respectively, compared with ReHiGate's 145.6M.

Width growth. Let d denote the inherited hidden size and $D > d$ the expanded size. We use $(d, D) = (768, 816)$ at 124M scale and $(1536, 1608)$ at 1.5B scale. Token and position embeddings are extended with zero-valued coordinates, so the residual stream at insertion has the form $x' = [x, 0_{D-d}]$.

Using the row-vector convention, an inherited linear map $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ is expanded as

$$W' = \begin{bmatrix} W & 0 \\ B & C \end{bmatrix}. \tag{33}$$

The upper-right zero block prevents inherited coordinates from producing new output coordinates at insertion, giving $x'W' = [xW, 0]$. The remaining new blocks are initialized with the backbone's

Table 7: **Compute resources by model scale.**

Setting	124M	1.5B
GPUs per run	1	8
Training endpoint	10B tokens	100B tokens
Full baseline cost	16 GPU-hours	870 GPU-hours
Continuation cost	3–15 GPU-hours	90–440 GPU-hours

default parameter initialization. Biases and normalization parameters are extended consistently over the new coordinates. The resulting model exactly reproduces the inherited function at the growth point.

Depth growth. At 124M scale, depth growth inserts two blocks, giving a 12→14-layer model. At 1.5B scale, five blocks are inserted, giving a 52→57-layer model. In both settings, the newly added blocks are placed after the inherited backbone blocks and before the final normalization and LM head.

For each added block, the internal attention, MLP, and normalization parameters are warm-started from trained backbone blocks, while the attention output projection and MLP down projection are initialized to zero. Each new block therefore initially satisfies $x_{\ell+1} = x_\ell + F_\ell(x_\ell)$ with $F_\ell \equiv 0$, making the depth-expanded model exactly function-preserving at insertion.

B.4 Architecture-Ablation Details

Unless otherwise stated, all architectural ablations use the 124M model with insertion at 5B tokens and retain the same backbone optimizer state, controller optimization, and continuation budget as the reference configuration. Only the design axis named in each ablation is changed.

History patterns. We compare fixed strides $s \in \{1, 2, 4, 8\}$ with sliding windows of one, two, or three preceding states. We also evaluate a swap-guided history rule derived from the layer-swap sensitivity in Figure 1(b). For each layer l, candidate preceding layers with $1 \leq \Delta L \leq 6$ are ranked by their distance to a target $\Delta L = 5$, and the top-k candidates are selected as history sources. This rule avoids both nearly interchangeable and strongly incompatible layers while allowing the selected history to vary with depth.

Gate placement and scope. Attention-side placement ablations apply the gate to the query, key, value, post-attention output, or residual update. Branch-scope ablations apply ReHiGate to attention only, MLP only, or both branches.

Gate form. We compare the reference shifted-sigmoid multiplicative gate with an additive SiLU controller and with RMSNorm applied to the branch output.

B.5 Compute Resources

All experiments were conducted on NVIDIA H200 GPUs. Table 7 summarizes the approximate compute used by each experimental setting.

C Functional Direction Diagnostics

This appendix describes the insertion-time functional diagnostics used in Section 5.2. All measurements are taken immediately after augmentation and before any optimizer update, with all models in evaluation mode. Each augmentation is constructed from the same trained 124M backbone at the 5B-token insertion point. The goal is to compare the functional directions exposed by the newly introduced parameters with those already available to the original backbone.

C.1 Functional Sketch Construction

Parameter groups. For the backbone reference, we use all parameters of the original baseline model. Importantly, this reference is computed directly from the original backbone rather than from the inherited-parameter Jacobian inside each augmented model.

For each augmentation, the new-parameter group contains the parameters associated with the newly introduced first-order directions. For ReHiGate, this group consists of the output gate factor V_l^b and gate bias $b_{g,l}^b$. The history factor U_l^b is excluded because its Jacobian is zero at the exact-identity initialization, as derived in Appendix A.3. For G_{zero}-Depth, we include the newly added attention and MLP output projections and their biases; the remaining parameters of the inserted blocks are first-order inactive at insertion because the output projections are zero-initialized. For G_{zero}-Width, the new-parameter group is defined by a mask over all newly introduced coordinates, including coordinates whose Jacobian is zero at the insertion point.

Probe outputs. For a parameter group g, let $f_g \in \mathbb{R}^n$ denote the functional output used by the diagnostic. For each token in the probe batch, we select the logit corresponding to the true next token. Thus f_g contains one scalar logit per probe token rather than the full vocabulary-logit vector or the training loss. The corresponding Jacobian is

$$J_g = \frac{\partial f_g}{\partial \theta_g} \in \mathbb{R}^{n \times p_g}, \tag{34}$$

where θ_g denotes the parameters in group g and p_g is the number of parameters in that group.

Materializing the full Jacobian is unnecessary. Instead, we sample m isotropic parameter-space directions,

$$\omega_{g,j} \sim \mathcal{N}\left(0, \frac{1}{p_g} I\right), \qquad j = 1, \ldots, m, \tag{35}$$

and compute their Jacobian-vector products. The resulting functional sketch is

$$S_g = [J_g \omega_{g,1}, \ldots, J_g \omega_{g,m}] \in \mathbb{R}^{n \times m}. \tag{36}$$

Each column of S_g therefore represents the first-order output change induced by one random direction in the corresponding parameter group.

In the reported experiments, the probe batch contains two validation sequences of 128 tokens each, giving $n = 256$, and we use $m = 32$ random tangent directions. All Jacobian-vector products are computed directly without constructing the full Jacobian.

C.2 Alignment Metrics

We compare each new-parameter sketch S_g with the backbone sketch S_b using three complementary alignment measures. For metrics based on the sketch geometry, we first center across probe tokens:

$$X_g = H S_g, \qquad H = I - \frac{1}{n} \mathbf{1}\mathbf{1}^\top. \tag{37}$$

Kernel CKA. We use linear centered kernel alignment:

$$\mathrm{CKA}(g, b) = \frac{\left\| X_g^\top X_b \right\|_F^2}{\left\| X_g^\top X_g \right\|_F \left\| X_b^\top X_b \right\|_F}. \tag{38}$$

Higher CKA indicates greater overall similarity between the functional sketches, so lower values indicate less alignment with the inherited backbone.

Top-subspace overlap. Let Q_g and Q_b contain the leading left singular vectors of X_g and X_b, respectively. We use a target subspace rank $k = 16$, truncated by the numerical rank of each sketch when necessary. The overlap is

$$\mathrm{Overlap}(g, b) = \frac{\left\| Q_g^\top Q_b \right\|_F^2}{\min(k_g, k_b)}, \tag{39}$$

where k_g and k_b denote the retained ranks for the two sketches. Lower overlap indicates that the dominant functional subspaces share fewer directions.

Table 8: **Functional diagnostic configuration.**

Setting	Value
Model / insertion point	124M / 5B
Gradient batch	2 training sequences $\times$ 128 tokens
Probe batch	2 validation sequences $\times$ 128 tokens
Probe outputs n	256
Random tangent directions m	32
Top-subspace rank k	16
ReHiGate controller rank	256
ReHiGate history stride	4
Forward precision	CUDA bfloat16 autocast
Independent diagnostic runs R	3

Functional cosine. Functional cosine measures the alignment of the first-order output changes induced by the training-loss gradients. For parameter group g, we first compute the normalized gradient on a separate training batch,

$$v_g = \frac{\nabla_{\theta_g} \mathcal{L}_{\text{train}}}{\left\| \nabla_{\theta_g} \mathcal{L}_{\text{train}} \right\|_2}. \tag{40}$$

We then apply this direction to the probe outputs through

$$u_g = J_g^{\text{probe}} v_g, \tag{41}$$

and define

$$\text{Cos}(g, b) = \frac{u_g^\top u_b}{\|u_g\|_2 \|u_b\|_2}. \tag{42}$$

The reported cosine uses these uncentered first-order probe-output changes. This diagnostic measures the functional agreement of normalized loss-gradient directions; it is not the cosine between AdamW parameter updates.

Sketch effective rank. We additionally measure the dimensionality represented by the centered new-parameter sketch X_g. Let

$$\lambda_i = \frac{\sigma_i^2(X_g)}{m}, \qquad q_i = \frac{\lambda_i}{\sum_j \lambda_j}. \tag{43}$$

We define the effective rank as

$$r_{\text{eff}} = \exp\left(-\sum_i q_i \log q_i \right). \tag{44}$$

Unlike CKA, subspace overlap, and functional cosine, effective rank does not compare the new parameters with the backbone. It measures the dimensionality of the functional sketch and serves as a control for whether lower backbone alignment is simply explained by a larger functional space.

C.3 Diagnostic Configuration

Table 8 summarizes the diagnostic configuration. The gradient and probe batches are sampled separately. All forward computations use CUDA bfloat16 autocast.

Each diagnostic run varies four sources of randomness: `construction_seed`, which controls augmentation construction and new-parameter initialization; `train_seed`, which controls the sampled training batch; `probe_seed`, which controls the sampled probe batch; and `tangent_seed`, which controls the random parameter-space directions used to construct the functional sketches.

Table 9: **Detailed insertion-time functional diagnostics.** Values are mean ± standard deviation across three independent runs. Lower values indicate less backbone alignment for CKA, top-subspace overlap, and functional cosine.

Method	Kernel CKA ↓	Subspace ↓	Cosine ↓	Eff. rank
G_{zero}-Depth	0.1564 ± 0.0104	0.1347 ± 0.0048	0.2455 ± 0.1089	27.33 ± 0.69
G_{zero}-Width	0.1605 ± 0.0020	0.1229 ± 0.0150	0.2277 ± 0.1579	24.54 ± 0.85
ReHiGate	$\mathbf{0.1300 \pm 0.0064}$	$\mathbf{0.1094 \pm 0.0031}$	$\mathbf{0.1534 \pm 0.0578}$	23.44 ± 0.88

C.4 Detailed Results

Each complete diagnostic run produces one value for each metric. We report the mean and sample standard deviation across the three independent runs:

$$\bar{a} = \frac{1}{R}\sum_{r=1}^{R} a_r, \qquad s = \sqrt{\frac{1}{R-1}\sum_{r=1}^{R}(a_r - \bar{a})^2}. \tag{45}$$

The reported standard deviation is therefore computed across complete diagnostic runs rather than across probe tokens or tangent directions within a run.

ReHiGate has the lowest alignment with the inherited backbone under all three measures. Its effective rank is also slightly lower than those of the two model-growth baselines, showing that the reduced alignment is not explained by a larger functional sketch.

D Analysis of Cross-Layer History

Section 5.3 showed that a factorized ReHiGate controller performs similarly to the unconstrained history-to-gate projection over a broad range of intermediate ranks. This appendix examines the structure of the cross-layer history that supports this behavior. We study three complementary properties: (i) redundancy across Transformer depth, measured through layer swaps; (ii) overlap between the subspaces contributed by different history lags; and (iii) the stability of the history spectrum across training stages. We also report the full per-layer spectral statistics underlying Figure 3(a).

D.1 Layer-Swap Structure Across Model Scales

We first characterize how much the residual computation changes when Transformer blocks at different depths are exchanged. For a pair of layers (i, j), we swap the corresponding blocks and report the validation-loss change $\Delta L_{i,j} = L_{i,j} - L_{\text{base}}$. Low swap sensitivity indicates that the two layers can be exchanged with only a small change in model behavior.

124M model. Figure 1(b) shows a broad low-sensitivity region across the middle layers. Adjacent swaps in this region incur only small loss changes, while sensitivity generally increases for more distant pairs and for swaps involving boundary layers. For example, middle-layer pairs such as layers 5–10 remain substantially less sensitive than long-range swaps involving layer 0. At the same time, swap sensitivity varies gradually across depth rather than remaining uniformly small.

This pattern supports the history design used by ReHiGate: neighboring and middle-layer states contain substantial redundancy, but representations from different depths are not interchangeable everywhere. A sparse history can therefore cover multiple depths without repeatedly reading highly similar neighboring states.

1.5B model. We repeat the same analysis on the 1.5B-parameter backbone. Figure 4 shows the same qualitative organization: swaps involving boundary regions are more costly, while a broad middle portion of the network remains comparatively low-sensitivity. The persistence of this pattern at the larger scale provides additional evidence that cross-layer redundancy is not specific to the 124M model.

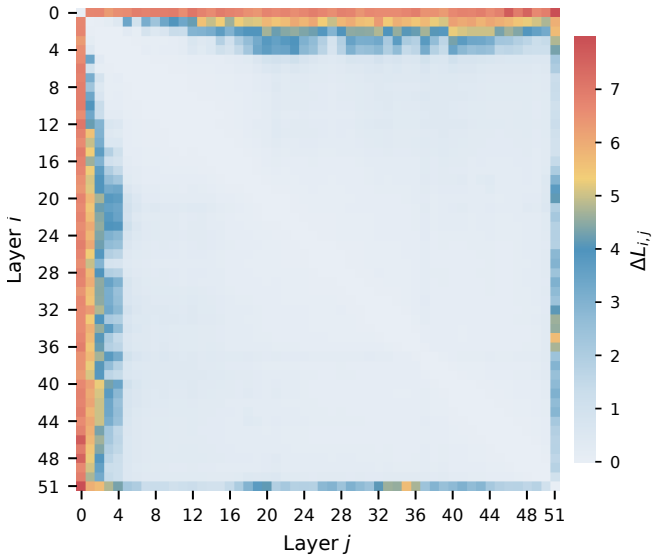


Figure 4: **Layer-swap sensitivity at 1.5B scale.** Each cell reports the validation-loss increase $\Delta L_{i,j}$ after exchanging layers i and j. As in the 124M model, boundary swaps are more sensitive while a broad middle region exhibits substantially lower sensitivity.

D.2 Cross-Lag Subspace Overlap

The spectral decay of the stacked history matrix could arise simply because each hidden state is individually low-dimensional, a phenomenon previously observed in Transformer representations (Aghajanyan et al., 2021). To distinguish this effect from redundancy *between* history states, we compare the spectral rank of individual lags with that of their concatenation.

Let $\mathcal{S}_l$ denote the set of valid stride-history offsets at layer l, with $k_l = |\mathcal{S}_l|$. For each $q \in \mathcal{S}_l$, let $H_{l,q} \in \mathbb{R}^{T\times d}$ denote the history matrix associated with that offset. The full history matrix is $H_l = [\, H_{l,q} \,]_{q\in\mathcal{S}_l}$. We write $r_{95}(\cdot)$ for the smallest rank capturing 95% of spectral energy and define the rank-95 redundancy ratio

$$\rho_l = \frac{\sum_{q\in\mathcal{S}_l} r_{95}(H_{l,q})}{r_{95}([\, H_{l,q} \,]_{q\in\mathcal{S}_l})}. \tag{46}$$

For a single history state, $\rho_l = 1$. When $\rho_l > 1$, the joint representation requires fewer spectral directions than the sum required by the individual states, indicating overlap between their dominant subspaces. The quantity $1 - 1/\rho_l$ gives the corresponding reduction relative to the additive per-lag rank.

Table 10 and Figure 5 show a clear increase in redundancy as additional lags are concatenated. For two-lag layers, ρ_l ranges from 1.38 to 1.59, corresponding to a 28–37% reduction relative to the sum of the individual 95%-energy ranks. For three-lag layers, ρ_l reaches 1.76–1.94, corresponding to a 43–48% reduction.

Importantly, the joint r_{95} in Table 10 already includes this cross-lag overlap. For example, layers 8–11 have additive per-lag rank sums of 1181–1373 but joint r_{95} values of only 672–742. This shows that stacking histories introduces substantial redundancy beyond the low-dimensionality of individual hidden states.

The controller rank that gives the best continuation performance is still smaller than these 95%-energy ranks. Thus the rank sweep in Section 5.3 should not be interpreted as directly recovering the full spectral rank of the history. Instead, the combined evidence suggests that residual modulation depends primarily on a compact subset of the variation present in the full cross-layer representation.

Table 10: **Per-layer rank-95 redundancy.** The joint r_{95} of stacked history states is consistently smaller than the sum of their individual r_{95} values once multiple lags are used.

Layer	Offsets $\mathcal{S}_l$	k_l	joint r_{95}	$\sum_q r_{95}(H_{l,q})$	ρ_l
0	$\{0\}$	1	444	444	1.00
1	$\{0\}$	1	224	224	1.00
2	$\{0\}$	1	137	137	1.00
3	$\{0\}$	1	295	295	1.00
4	$\{0, 4\}$	2	537	856	1.59
5	$\{0, 4\}$	2	507	714	1.41
6	$\{0, 4\}$	2	470	650	1.38
7	$\{0, 4\}$	2	556	810	1.46
8	$\{0, 4, 8\}$	3	708	1373	1.94
9	$\{0, 4, 8\}$	3	703	1243	1.77
10	$\{0, 4, 8\}$	3	672	1181	1.76
11	$\{0, 4, 8\}$	3	742	1331	1.79

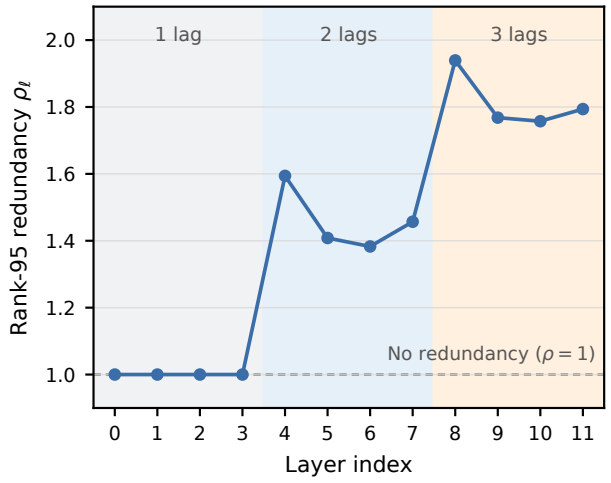


Figure 5: **Rank-95 redundancy ratio ρ_l across layers.** Single-lag layers have $\rho_l = 1$ by construction. Adding a second and third lag increases ρ_l, showing increasing overlap among the dominant subspaces of the stacked history states.

D.3 Stability Across Training

We next test whether the observed spectral structure is specific to the 5B-token state used in the main analysis. We repeat the same SVD measurement at 10B tokens and compare the normalized rank $r_{95}/(k_l d)$ across layers.

As shown in Figure 6, the layerwise profiles remain similar after an additional 5B training tokens. The relative deviation is below 13% on every layer except layer 2. The cross-layer spectral structure is therefore stable across the two training stages measured here, rather than being specific to the 5B insertion point.

D.4 Detailed Spectral Measurements

For completeness, Table 11 reports the full per-layer spectral measurements summarized in Figure 3(a). In addition to r_{95}, we report stable rank $\|H_l\|_F^2/\|H_l\|_2^2$, entropy-based effective rank (Roy & Vetterli, 2007), and the alternative energy thresholds r_{90} and r_{99}.

All measures indicate substantial concentration relative to the ambient history dimension. Stable rank remains below 6 across all layers, while even the conservative r_{99} remains below approxi-

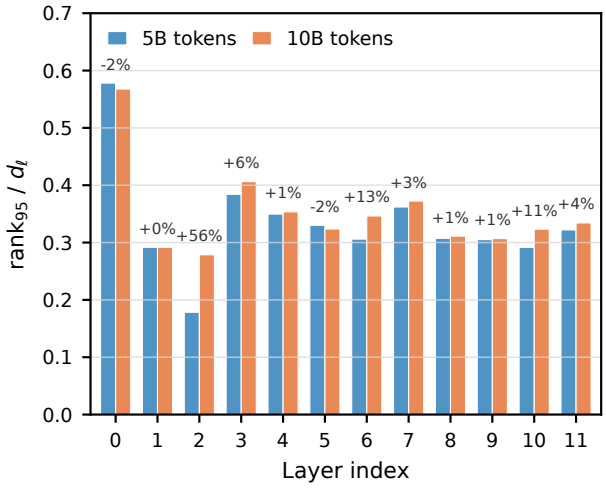


Figure 6: **Stability of cross-layer spectral structure across training.** Per-layer $r_{95}/(k_l d)$ at the 5B- and 10B-token states. The profiles remain similar across the two measured stages, with relative deviation below 13% for all layers except layer 2.

Table 11: **Per-layer spectral statistics of stacked residual history at the 5B-token state.** r_p denotes the smallest rank that captures $p\%$ of spectral energy.

Layer	Offsets	$k_l d$	Stable rank	Eff. rank	r_{90}	r_{95}	r_{99}
0	$\{0\}$	768	5.56	592.2	334	444	622
1	$\{0\}$	768	2.61	416.5	155	224	375
2	$\{0\}$	768	1.75	367.9	53	137	320
3	$\{0\}$	768	2.43	505.8	188	295	513
4	$\{0, 4\}$	1536	3.92	926.2	369	537	895
5	$\{0, 4\}$	1536	4.19	848.3	375	507	760
6	$\{0, 4\}$	1536	2.83	845.9	324	470	751
7	$\{0, 4\}$	1536	3.55	952.6	391	556	910
8	$\{0, 4, 8\}$	2304	4.67	1214.3	496	708	1174
9	$\{0, 4, 8\}$	2304	5.21	1179.1	511	703	1117
10	$\{0, 4, 8\}$	2304	3.78	1178.6	469	672	1113
11	$\{0, 4, 8\}$	2304	4.93	1248.7	522	742	1215

mately $0.60\, k_l d$ for the multi-state history layers. Together with the cross-lag analysis above, these measurements show that the concatenated history contains substantial spectral redundancy.

D.5 Summary

The analyses above characterize the structure exploited by the ReHiGate controller from several complementary views. Layer-swap experiments show substantial redundancy across middle Transformer layers at both 124M and 1.5B scales, while retaining variation across depth. The rank decomposition shows that stacking multiple history states creates additional redundancy through cross-lag subspace overlap. Finally, the spectral profile remains similar between the measured 5B and 10B training stages.

These results support using a sparse, factorized history controller: cross-layer history contains substantial shared structure, while the continuation results in Section 5.3 show that effective residual modulation can be achieved with a controller rank well below the dimension required to retain most of the raw history energy.