RIVER-HILL BUDGETS: UNDERSTANDING THE EFFECTS OF ANNEALING IN MODEL GROWTH

Anonymous authors
Paper under double-blind review

ABSTRACT

Model growth initializes a larger model from a smaller source model that has already been trained, but saves compute only if the grown model properly reuses that source's training. Work on growth has focused on the growth operator, which is in charge of preserving the source model's function across the transfer while ensuring the newly added parameters are properly made use of. However, when parameters are transferred and the learning-rate schedule is reset, the sudden learning-rate increase creates a training offset on top of the parameter transfer. We separate the impact of the growth operator from the learning-rate increase and find that the newly expanded parameter space does not necessarily help recover from this schedule offset. Using the river-valley view of the loss landscape, we explain and quantify this offset. For a standard BERT configuration, the river-valley view predicts that 45% of progress made during the decay to 10% peak learning rate is lost upon reset, and up to 50% for a decay to zero. The same model trained with 45% fewer anneal steps, making only reset-surviving progress, closely matches the annealed run's loss trajectory. These results show that the source model's learning-rate schedule severely limits how much training survives model growth.

1 INTRODUCTION

Model growth trains a large model by starting from a smaller model that has already been trained instead of from a random initialization. We call the smaller model the *source* and the larger one the *grown model*. The growth procedure applies a *growth operator* that maps the source's parameters onto the larger architecture by widening its layers or adding new ones, and then continues training. The aim is to reach the performance of a large model trained from scratch with less total compute by exploiting the source's pretraining. Growth procedures are accordingly evaluated by their compute savings relative to training the larger model from scratch. Much of the prior work focuses on *function-preserving* growth operators (Chen et al., 2016; 2022; Wang et al., 2024). These produce a grown model that computes the same function as the source and hence starts training at the source's loss.

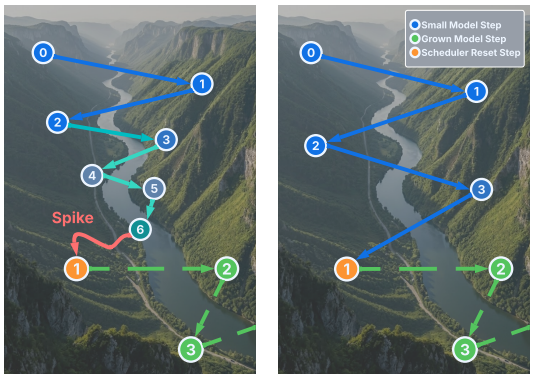

Figure 1: River-valley view of training (Wen et al., 2025): an annealed source (left) and a flat source (right). The anneal lowers the hill level but slows progress along the river, and a learning-rate reset discards the lowered hill. Both sources reach the same river progress; the annealed source takes nearly twice as many steps to get there.

Starting at the source's loss does not mean the grown model keeps it. The source is typically fully trained, so its learning rate has been annealed to a small value. Training the grown model begins again at the peak learning rate. This reset raises the loss whatever the operator does, and the grown model must first recover the source's loss before improving on it. The compute spent on the source is then partly wasted even when the final performance and compute savings look good. A growth procedure thus makes two simultaneous

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.

changes: one to the model's parameters through the growth operator and another to the optimizer through the reset of the learning rate.

The savings attributed to the operator therefore reflect both the operator and the learning-rate reset. This raises a question that has received little attention: *what share of the source's training steps still contributes to the grown model's loss once the learning rate is reset?* If the growth operator compensated for the loss increase caused by the reset, the source's schedule would not matter. If it does not, part of the source's training may be lost to the schedule regardless of how well the operator transfers the source's function.

By separating the contributions of the growth operator and the learning-rate reset, we measure each as an *offset*: the excess loss of a run relative to an otherwise identical run without that change. Measured per step, the schedule offset closes quickly in the grown model. This suggests that the additional parameters introduced by the operator help the model absorb the reset. However, measured at equal compute, the operator we study does not close the schedule offset any faster (Section 2): the reset leaves the same offset whether or not the model is grown. A good operator is therefore not by itself enough to exploit the source's pretraining.

If the operator does not remove the schedule offset, the question becomes what determines its size and whether it can be anticipated. We apply the river-valley view of the loss landscape (Wen et al., 2025) to relate the schedule offset to the source's learning-rate schedule. While this model of the landscape rests on strong assumptions, it has proven a useful approximation of language-model training (Song et al., 2025; Liu et al., 2025). In this view, part of the loss decrease obtained by annealing is maintained only while the learning rate stays low, and resetting the learning rate gives it back. The share of the anneal lost this way follows from the schedule alone (Section 3), and reaches half of the anneal for a cosine anneal to zero.

We test this prediction on BERT (Section 4). For our cosine anneal to $0.1\,\eta_{\max}$, the model predicts that the reset discards 45% of the anneal's steps. A source trained at a constant learning rate for 45% fewer steps than the anneal should therefore behave like the annealed source after the reset. The two sources indeed follow nearly the same loss trajectory, with or without growth, so the model accounts for the schedule offset and lets its cost be estimated from the schedule before training.

2 Separating the Growth and Schedule Offsets

When a model is grown from an annealed checkpoint, two perturbations occur at the same step, which we call the *event*. The growth operator changes the model, and the learning rate jumps from the bottom of the source's anneal back to $\eta_{\max}$. The loss spike and recovery that follow reflect both. This section separates them and asks whether growth helps the model recover from the schedule's perturbation.

Offsets. A *spike* is the visible jump in the loss at the event; what matters is the part of it the training curve retains afterwards. We measure this as an *offset*: the difference in loss between a perturbed run and the equivalent run without the perturbation, compared at equal compute c counted from the event, since a grown model spends more compute per step. We consider four runs, which cross growth with the source's schedule: *flat* sources are trained at a fixed learning rate $\eta_{\max}$, while *annealed* sources follow a cosine decay to $0.1\,\eta_{\max}$ and have their learning rate reset to $\eta_{\max}$ at the event.

Definition 1 (Growth offset)**.** *The* growth offset *is the offset caused by the growth operator alone,*

$$\Delta_{\textit{grow}}(c) \;=\; \mathcal{L}_{\textit{flat+grow}}(c) - \mathcal{L}_{\textit{flat}}(c). \tag{1}$$

Definition 2 (Schedule offset)**.** *The* schedule offset *is the offset caused by the source's schedule alone,*

$$\Delta_{\textit{reset}}(c) \;=\; \mathcal{L}_{\textit{reset}}(c) - \mathcal{L}_{\textit{flat}}(c). \tag{2}$$

Here $\mathcal{L}_{\text{reset}}$ and $\mathcal{L}_{\text{reset+grow}}$ denote the annealed source after the reset, without and with growth. A positive offset is excess loss a run still has relative to its reference at the same compute; until it closes, the run spends compute recovering ground rather than improving on its reference.

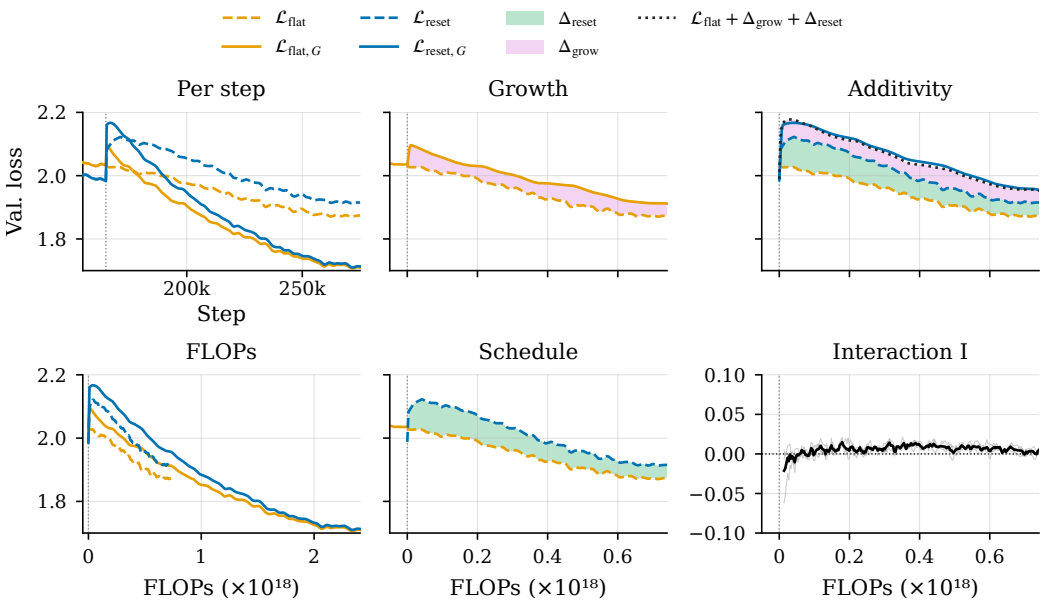

Figure 2: Separating the growth and schedule offsets at a BERT growth event. Left: the four runs per step (top) and at equal compute (bottom); the subscript G in the legend denotes growth. Middle: the growth offset Δ_{grow} (top) and the schedule offset Δ_{reset} (bottom), shaded against their references at equal compute. Right: $\mathcal{L}_{\text{flat}} + \Delta_{\text{grow}} + \Delta_{\text{reset}}$ against $\mathcal{L}_{\text{reset+grow}}$ (top), and the interaction I (bottom); thin lines are individual seeds, the thick line their mean.

How the offsets combine. A model grown from an annealed checkpoint is subject to both offsets at once. We measure how they combine through the *interaction*

$$I(c) \;=\; \mathcal{L}_{\text{reset+grow}}(c) - \mathcal{L}_{\text{flat}}(c) - \Delta_{\text{grow}}(c) - \Delta_{\text{reset}}(c), \tag{3}$$

the part of the combined offset that the two individual offsets do not explain. Three outcomes are possible. If $I < 0$, the offsets *absorb* each other: the new parameters accelerate recovery from the schedule offset, and a good enough growth operator would make the source's schedule irrelevant. If $I \approx 0$, the offsets are *independent*: the new parameters introduce their own offset, the learning-rate reset introduces a distinct one, and the operator can address only its own share. If $I > 0$, the offsets *compound*: annealing the source actively harms the growth.

Setup. The flat and annealed sources are trained for 165k steps, then grown or continued for 110k steps, giving four runs with three seeds each. Growth uses LEMON (Wang et al., 2024), so that any offset it produces cannot be attributed to a broken function; an operator that does not preserve the function could add its own disturbance and make the offsets harder to separate. After the event, all four runs follow the same cosine schedule from $\eta_{\max}$, so the flat runs see no change in learning rate. A grown step requires 3.25 times the compute of a source step (Appendix B), which sets the compute axis in Figure 2. Controls for the optimizer state and for a re-warmup at the event are in Appendix E.

Per step, growth appears to recover quickly. Compared per step (Figure 2, top left), both grown runs recover from their spikes within a small fraction of total steps and soon fall below the runs that were not grown, whether or not the source was annealed. The comparison is misleading: each grown step performs 3.25 times the compute of a source step, and some faster improvement per step is expected from that compute alone.

At equal compute, growth does not reduce the schedule offset. Compared at equal compute (Figure 2, bottom left), the picture changes. The grown runs stay above their non-grown counterparts for the whole window. The schedule offset is larger than the growth offset and persists longer, and neither is erased by the other. The sum $\mathcal{L}_{\text{flat}} + \Delta_{\text{grow}} + \Delta_{\text{reset}}$ closely predicts the combined run

(top right), and the interaction stays small throughout (bottom right), with a slight positive excess. Most importantly, the interaction is never negative: the offsets do not absorb each other, and at equal compute the operator does not visibly speed up recovery from the schedule offset. The new parameters recover from their own perturbation, and the learning-rate reset imposes a separate offset that growth does not address. Part of the source's training is thus lost to the schedule before the operator has any say in how much of it is reused. While this experiment is limited to a single operator and three seeds, it suggests that the operator alone is not always enough to preserve the source's training. This motivates examining the schedule offset on its own: Section 3 describes how much of a source's training a reset leaves reusable, and Section 4 tests this in the context of growth.

3 River–Hill Budget

River-valley picture. We adopt the river-valley view of the pretraining loss landscape (Wen et al., 2025): the loss rises steeply along most directions (the *hillsides*) and varies slowly along a low-curvature valley floor (the *river*). Under a large learning rate, stochastic gradient noise keeps the iterate oscillating across the hillsides while it advances quickly along the river; decaying the learning rate damps the oscillation and reveals progress already made. Wen et al. (2025) analyze SGD on such a landscape, with $d-1$ hillside directions and gradient noise of variance σ^2 confined to them. Throughout, $\eta_{\max}$ denotes the peak (stable-phase) learning rate. Their central result (Theorem 3.4) splits the expected loss of SGD at a constant learning rate η into two terms,

$$\mathbb{E}[L(w_k)] \;\approx\; L\big(x(T)\big) + \tfrac{(d-1)\eta\sigma^2}{2}, \qquad T \approx \eta k, \tag{4}$$

where $x(\cdot)$ is the river, the path followed by the gradient flow along the valley floor, and $x(T)$ is the point it reaches after time T. The second term is the excess loss due to oscillation across the hillsides. The first term depends on the learning rate only through the accumulated time T; the second depends only on the current η. We name the two terms along a general schedule $(\eta_k)_{k\geq 1}$, and state the properties we use as assumptions.

Definition 3 (River progress)**.** *The* river progress *after step k is the river time $T_k = \sum_{i\leq k}\eta_i$ of Wen et al. (2025), expressed in steps at the peak learning rate:*

$$B_k \;=\; \frac{T_k}{\eta_{\max}} \;=\; \sum_{i\leq k}\frac{\eta_i}{\eta_{\max}}. \tag{5}$$

Normalizing by $\eta_{\max}$ lets river progress be compared directly with step counts: one unit is the progress of one step at $\eta_{\max}$. River progress only accumulates. A change in learning rate alters how fast it grows, never the value already reached. The loss it produces is the river loss $L_R(B_k)$*, a decreasing function of B_k.*

Definition 4 (Hill level and hill reduction)**.** *The* hill level *is the excess loss due to oscillation across the valley, $H_k = \mathbb{E}[L_k] - L_R(B_k)$. At equilibrium under a learning rate η, the hill level is $H(\eta) \approx c\,\eta$ for a constant $c > 0$; after a change in learning rate it relaxes to this value within $O\big(1/(\eta\gamma)\big)$ steps, where γ is the smallest hillside curvature, and retains no memory of earlier learning rates.*

The hill reduction *is the loss removed relative to a flat run at the same river progress,*

$$\Delta H_k \;=\; H(\eta_{\max}) - H_k \;\approx\; c\,(\eta_{\max} - \eta_k) \quad \textit{(at equilibrium)}. \tag{6}$$

Hill reduction is set by the current learning rate alone. Whenever the learning rate increases, the hill refills to the level of the new rate and the corresponding part of the hill reduction disappears; all of it disappears once the rate returns to $\eta_{\max}$.

The budget split. Each step at learning rate η_k realizes a fraction $p_k = \eta_k/\eta_{\max}$ of the river progress available to a step at peak learning rate. The remaining fraction $1 - p_k$ is forgone: it adds nothing to river progress.

Definition 5 (Forgone progress)**.** *The* forgone progress *after step k is*

$$F_k \;=\; \sum_{i\leq k}\Big(1 - \frac{\eta_i}{\eta_{\max}}\Big). \tag{7}$$

Schedule segment	Reusable (B/N)	Discarded on reset (F/N)
Flat at $\eta_{\max}$	1.00	0.00
WSD-S (90% flat, 10% inverse decay to $0.1\,\eta_{\max}$)	0.93	0.07
Cosine or linear to $0.1\,\eta_{\max}$	0.55	0.45
Cosine or linear to 0	0.50	0.50

Table 1: Share of a schedule segment's N steps that remains reusable after a reset, as river progress, and share discarded, relative to a flat run. The inverse-proportional decay of Wen et al. (2025) to $0.1\,\eta_{\max}$ keeps $\ln 10/9 \approx 0.26$ of its steps (Appendix C).

Each step contributes p_k to river progress and $1 - p_k$ to forgone progress, so B_k and F_k always sum to the number of steps taken, and the loss decomposes as

$$\mathbb{E}[L_k] \;=\; L_R(B_k) \;+\; H(\eta_{\max}) \;-\; \Delta H_k. \tag{8}$$

The asymmetry between the two terms is central to what follows. River progress is kept indefinitely; hill reduction is held only while the learning rate stays low, and is realized only if the model is evaluated at that point. The forgone progress F_k is what an anneal costs in steps; the hill reduction, set by the final learning rate, is what it obtains in return.

When the learning rate is reset from a low value back to $\eta_{\max}$, the hill refills to its equilibrium $H(\eta_{\max})$, so $\Delta H \to 0$. Because the hill level retains no memory of the schedule, the hill reduction obtained during the anneal is discarded in full, and the forgone progress that produced it is not recovered. Only B survives.

Proposition 1 (Reset equivalence)**.** *Consider a run that anneals over N steps with schedule $(\eta_k)_{k=1}^{N}$, then resets to $\eta_{\max}$ and continues until the hill has refilled. Under the assumptions of Wen et al. (2025), its expected loss thereafter approximately equals that of a flat run at $\eta_{\max}$ advanced by B_N steps from the same starting point, rather than N steps.*

Proof sketch. By Definition 3, the river state after the anneal depends on the schedule only through B_N. After refilling, the hill level equals $H(\eta_{\max})$ for both runs, independently of their histories (Definition 4). Both terms of (8) therefore coincide. □

For growth, B_N is the part of an annealed source's training that a grown model can reuse; the forgone progress F_N is lost at the reset, before the operator is applied.

For a cosine anneal from $\eta_{\max}$ to $r\,\eta_{\max}$, the mean of $\eta_k/\eta_{\max}$ is $(1 + r)/2$ (Appendix C), so

$$B_N \;=\; N\,\frac{1+r}{2}. \tag{9}$$

At $r = 0.1$, a reset keeps 55% of the anneal's steps as river progress, and the remaining 45% is forgone. Table 1 gives the same fractions for other schedules.

4 River-Matched Sources

Section 2 shows that the source's learning-rate schedule contributes its own offset at growth, alongside the operator's. The budget of Section 3 explains this schedule offset and makes it predictable. An annealed source holds hill reduction that a reset discards, and it paid for that hill reduction with the forgone progress F of its anneal. A source that never obtained the hill reduction has none to lose. Proposition 1 turns this into a prediction that can be tested without fitting anything. Take a source trained with an anneal, and a second source trained flat at $\eta_{\max}$ only until its river progress equals that of the first. The second source takes F fewer steps. If the proposition holds, the two sources become indistinguishable once both are reset and their hills have refilled: the extra steps of the annealed source produced only hill reduction, and the reset discards it. We call the second source *river-matched*. Our aim is not an exact prediction of the loss, but an estimate of how much of the annealed source's training survives the reset: if river progress alone accounts for the bulk of the post-reset behaviour, the reset keeps B of the anneal's steps and loses F.

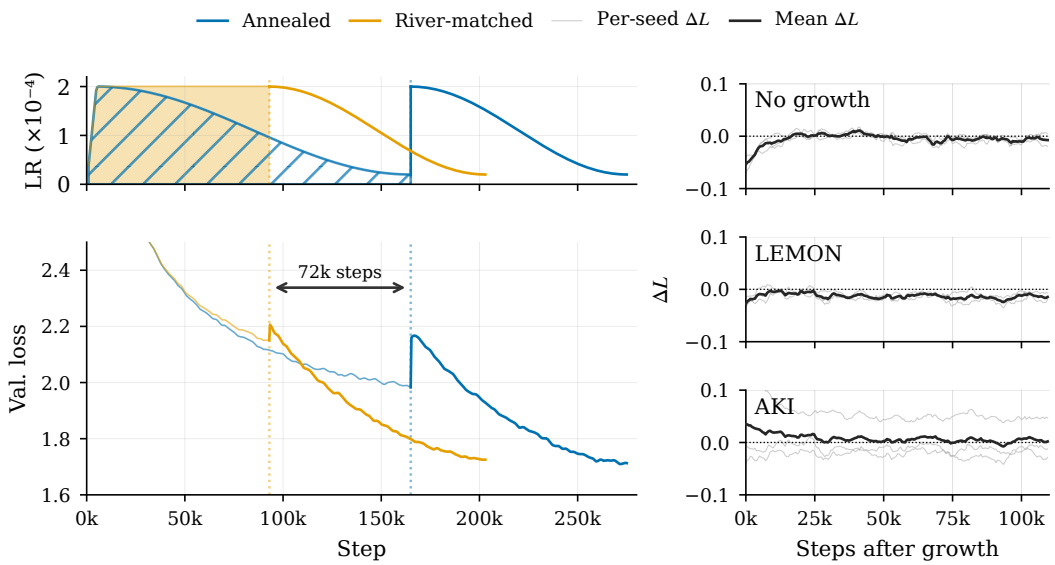

Figure 3: River matching. Left: learning rate and validation loss for the annealed source and its river-matched flat source, both grown with LEMON at their events. The areas under the two schedules up to the events are equal, so both sources have the same river progress B; the river-matched source reaches its event 72k steps sooner. Right: ΔL, annealed minus river-matched, aligned at each source's event, with growth withheld, under LEMON and under AKI; thin lines are individual seeds, the thick line their mean. $\Delta L < 0$ means the annealed source has the lower loss. Loss curves without growth and with AKI are in Figure 6.

The river valley as an intuitive approximation. The analysis of Section 3 is derived for SGD on an idealized landscape, and our runs train BERT with AdamW. We therefore use the river-valley picture as an intuitive approximation of the training dynamics, as prior work has done for language models trained with Adam-type optimizers (Section 5), and test whether predictions derived from it hold. River matching rests on the assumption our analysis cannot guarantee under AdamW: that river progress is proportional to the learning rate. If it failed, sources matched on B would not coincide after the reset. Small residuals, discussed at the end of this section, are expected even if it holds approximately.

4.1 Setup

Matching condition. The annealed source is the one of Section 2, trained for $N = 165{,}000$ steps: a linear warmup over $W = 5{,}060$ steps, then a cosine decay over $N - W = 159{,}940$ steps from $\eta_{\max}$ to $r\,\eta_{\max}$ with $r = 0.1$. By (9), the anneal's river progress is $B = (N - W)(1 + r)/2 = 87{,}967$. A flat source with the same warmup gains one unit of river progress per step, so it is river-matched at

$$s \;=\; W + B \;=\; 93{,}027. \tag{10}$$

The annealed source's forgone progress is $F = N - s = 71{,}973$ steps, 45% of the anneal. The warmup contributes equally to both sources and cancels. Both s and F are fixed before training, from the schedule alone.

Protocol. At its event (step N for the annealed source, step s for the river-matched one), each source is continued at its original size, grown by LEMON (Wang et al., 2024), which preserves its function, or grown by AKI (Chen et al., 2022), which does not, before the first post-event step. Both sources of a pair are handled the same way, are reset to $\eta_{\max}$, and are trained with the same 110k-step cosine. Without growth, the comparison is the direct test of Proposition 1; LEMON adds a change of model that leaves the function intact, and AKI one that does not. We report ΔL, the annealed source's validation loss minus the river-matched source's, with each curve aligned at its own event. Because the two runs follow identical schedules after the event, any persistent difference

in ΔL must come from their states at the event. To vary the amount of annealing, we also apply the matching within scheduled LEMON growth, where a source whose cosine runs over 220k steps is grown partway through it, at $g \in \{50\text{k}, 100\text{k}, 150\text{k}\}$; the anneal at the event is then partial, and F is a smaller share of the steps trained since warmup. Optimizer, data and model details are given in Appendix A.

What the prediction looks like. At the event the two sources have equal river progress but different hill levels: the annealed source sits at the bottom of its anneal, with hill reduction $\Delta H \approx c\,(1-r)\,\eta_{\max}$ if its hill has equilibrated, while the river-matched source is at the equilibrium hill of $\eta_{\max}$. Without growth, and under LEMON, which preserves the source's function, the proposition therefore predicts that ΔL starts negative, since the annealed source begins with its hill still low, rises to zero as the reset refills its hill, and then stays at zero. The initial transient is the discard itself, and the level afterwards tests whether anything other than river progress survived. Under AKI, the grown model no longer computes the source's function: the operator perturbs it by an amount that varies between sources and seeds, and this perturbation sets the starting point of ΔL, so only the later level is predicted. The proposition concerns a single model, and we carry it across growth as a hypothesis. Section 2 supports it: LEMON did not absorb the schedule offset, so the schedule affects the grown model's loss much as it does without growth.

4.2 Results

Matched on river progress, the two sources follow the same trajectory. Before its event, the annealed source's loss is below its river-matched counterpart's (Figure 3); in the model's terms, this gap is its hill reduction. After the event, ΔL behaves the same way with or without growth and under either operator: it starts away from zero, returns close to zero within a small fraction of post-event training, and stays near it. Without growth and under LEMON it starts negative, as predicted, and under LEMON the annealed source's spike is correspondingly larger. Under AKI it starts positive, since the operator sets the starting point, and individual seeds are noisier, but their mean also returns close to zero. The annealed source trained 72k more steps, and, to within the small residual discussed below, none of them are visible after the event. Matching on steps instead leaves a persistent gap: the schedule offset of Section 2 compares an annealed and a flat source at the same step count and remains positive throughout the window. River matching removes this persistent part of the schedule offset. The reset thus keeps about 55% of the anneal's steps and discards the other 45%, as predicted.

The later a source is grown within its anneal, the less of its training is reused. Scheduled growth varies the depth of the anneal at the growth event (Figure 4). Computed from the source's schedule before training, the forgone progress is 3%, 13% and 27% of the steps trained since warmup for $g = 50$k, 100k and 150k (Table 4). The initial transient grows with the depth of the anneal. After the hill has refilled, ΔL stays at zero for $g = 50$k and close to it for $g = 100$k, while at $g = 150$k the annealed source ends slightly ahead. River progress thus predicts most, but not all, of what a reset leaves reusable, and the unpredicted remainder, which favours the annealed source, grows with the depth of the anneal. The same holds for complete anneals of different lengths and for a linear decay (Appendix G).

Residual drift. Late in training, ΔL settles slightly below zero, most for the deepest anneal ($g = 150$k). Some residual is expected, because the two sources differ in more than river progress. The annealed source sees $N/s \approx 1.8$ times as many batches for the same river progress, and AdamW does not follow the SGD dynamics the model is derived for, so river progress is only approximately proportional to the learning rate. Our runs do not separate these factors. Despite them, the residual is small next to the separation that matching removes: river progress accounts for the bulk of what survives the reset.

5 Related Work

Model growth. Growth operators initialize a larger model from a trained smaller one, most often by expanding width or depth. We consider width expansion only. Net2Net (Chen et al., 2016) introduced function-preserving width expansion by duplicating neurons and rescaling their outgoing

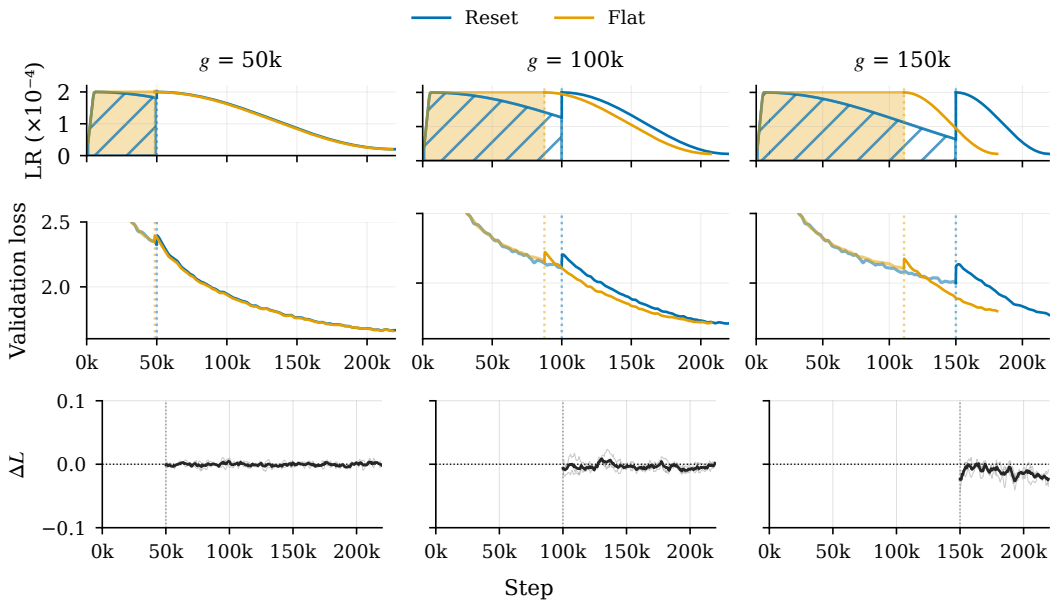

Figure 4: River matching within scheduled LEMON growth at $g = 50$k, 100k and 150k, annealed source (legend: Reset) against river-matched source (legend: Flat). Top: learning rates; the shaded and hatched areas up to each source's growth event are equal. Middle: validation loss. Bottom: ΔL, annealed minus river-matched, aligned at the growth events. The deeper the source's anneal at g, the sooner its river-matched counterpart reaches the event, and the larger the initial transient.

weights. bert2BERT (Chen et al., 2022) adapted this to Transformers with function-preserving initialization (FPI), and proposed advanced knowledge initialization (AKI), which also draws on the parameters of the next layer to break the symmetry of the duplicated neurons, at the cost of exact function preservation. LiGO (Wang et al., 2023) learns the expansion as a linear map, MSG (Yao et al., 2024) preserves the function by masking the new parameters and removing the mask gradually, and LEMON (Wang et al., 2024) expands Transformers without any loss in the source's function. These operators are typically evaluated by the compute they save relative to training the larger model from scratch, to a target loss or downstream performance (Chen et al., 2022; Wang et al., 2024), and LEMON also tunes the grown model's learning-rate schedule. Such savings measure the outcome of growth, not how much of the source's training the grown model reuses, and the source is usually taken at the end of its training. The effect of the source's own schedule on that reuse has not, to our knowledge, been separated from that of the operator.

The learning rate at growth. Some growth methods already manage the learning rate at the growth step. Staged training (Shen et al., 2022) treats the optimizer state and the learning-rate schedule as part of the training state that a growth operator must pass on, so that training continues at the same rate of loss decrease, and grows partway through training. SPARKLING (Yu et al., 2026), which expands width mid-training, lets the existing parameters keep their optimizer state and schedule, and resets the optimizer state of the new parameters and briefly raises their learning rate to break the symmetry of the copied neurons. Outside growth, the same reset is a known problem in continual pretraining, where re-warming the learning rate first raises the loss (Gupta et al., 2023; Ibrahim et al., 2024), and the warmup-stable-decay (WSD) schedule avoids it by annealing only branches taken from a constant-rate main branch (Hu et al., 2024; Hägele et al., 2024). Wen et al. (2025) explain WSD through the river-valley picture, and report that models trained with a cosine schedule are implicitly hurt for continued training. These studies acknowledge the schedule offset and work to mitigate it. Our work focuses on understanding and evaluating the impacts of this offset.

The river-valley picture. The river-valley picture (Wen et al., 2025) has been used to study language models trained with Adam-type optimizers (Song et al., 2025; Liu et al., 2025), although the same loss curves also admit explanations from convex optimization (Schaipp et al., 2025). Other

work predicts loss curves from the learning-rate schedule by fitting terms for the accumulated learning rate and for annealing (Tissue et al., 2024; Luo et al., 2025); the accumulated learning rate there corresponds to our river progress. We look at the reset and its implications for growth.

6 DISCUSSION

The river valley across growth. The river-valley picture describes a model trained under a changing learning rate, but does not consider a change of architecture during training. Since Section 2 found the same schedule offset with and without growth, we expected its predictions to carry across growth, and the results of Section 4 are consistent with this under both operators. How the source's river and hill states transfer to the larger model remains open, however. A description of how the river-valley structure transforms under growth would explain when and why the prediction holds across growth.

Cyclic schedules and the case for WSD sources. Wen et al. (2025) report that a model trained with a cosine schedule is implicitly hurt for continued training, and that cyclic cosine falls clearly behind WSD-S when training resumes after each cycle. Our results are consistent with this, and suggest why and by how much: each cosine cycle keeps 55% of its steps as river progress at $r = 0.1$, and half at $r = 0$, and a reset discards the rest. For growth, the consequence is direct. Once a checkpoint has been trained, its schedule cannot be changed, and neither can the share of its training that the reset will discard. The loss can only be addressed by how sources are trained in the first place. Our results therefore reinforce the case for WSD-style training in progressive settings (Hu et al., 2024; Hägele et al., 2024): a source kept on a flat branch discards little when grown or continued, and so remains almost fully reusable, unlike one taken from the end of an anneal.

Growth operators and the schedule. The operator we studied did not accelerate recovery from the schedule offset, but nothing rules out one that does. This possibility may also help explain why rankings between operators can differ across settings. When sources are annealed, an operator that speeds recovery from the reset gains an advantage unrelated to how well it transfers the source's function or uses the new parameters. Without a reset, other properties, such as how well it breaks the symmetry of the added parameters, may decide the ranking instead. An operator could therefore save substantial compute while reusing little of its source, and in that case the value of growth itself would be in question. The schedule offset would thus become a new factor to consider when designing and comparing growth operators.

Limitations and future work. This paper is intended as a guide to an issue that growth evaluations can overlook, rather than a complete analysis of it. Our experiments use one pair of BERT models, a single operator, LEMON, for the decomposition, and one anneal floor, and they measure reuse relative to a flat source rather than against a larger model trained from scratch, which would anchor these relative numbers. The natural next step is to measure the schedule's effect across operators, tasks and scales: whether other operators interact with the schedule offset, how the share of training a reset discards varies with the model and the data, and whether the river-valley picture remains as useful a guide when the grown model differs more from its source.

7 CONCLUSION

How much of a source's training survives growth is decided by the schedule before any growth operator is applied. The raw number of steps the source was trained for is therefore not a direct measure of how much of it the larger model reuses. Part of the problem of growth thus moves away from the operator and toward the training that precedes it. Operators, and comparisons between them, are shaped by the state of their sources, and training that anticipates growth can keep more of what it has already paid for. Treating the source's schedule as part of the growth procedure, rather than as a fixed starting point, opens this part of the problem to study.

AI USE STATEMENT

We used a large language model to edit the paper's text and suggest wording, and to help locate related work. We reviewed all AI-assisted text, and checked every reference it suggested against the original source. We take responsibility for the final content of this work, including text and claims produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix A lists the models, data, optimizer, schedules, seeds and evaluation protocol of every experiment. Appendix B gives the FLOP accounting and how runs are compared at equal compute, and Appendix C derives the river-matched steps of every run (Table 4), which follow from the schedule alone.

REFERENCES

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 (ACL)*, 2022.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations (ICLR)*, 2016.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to (re)warm your model? *arXiv preprint arXiv:2308.04014*, 2023.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling (COLM)*, 2024.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research*, 2024.

Ziming Liu, Yizhou Liu, Jeff Gore, and Max Tegmark. Neural thermodynamic laws for large language model training. *arXiv preprint arXiv:2505.10559*, 2025.

Kairong Luo, Haodong Wen, Shengding Hu, Zhenbo Sun, Zhiyuan Liu, Maosong Sun, Kaifeng Lyu, and Wenguang Chen. A multi-power law for loss curve prediction across learning rate schedules. In *International Conference on Learning Representations (ICLR)*, 2025.

Fabian Schaipp, Alexander Hägele, Adrien Taylor, Umut Simsekli, and Francis Bach. The surprising agreement between convex optimization theory and learning-rate scheduling for large model training. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. Staged training for transformer language models. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Minhak Song, Beomhan Baek, Kwangjun Ahn, and Chulhee Yun. Through the river: Understanding the benefit of schedule-free methods for language model training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Howe Tissue, Venus Wang, and Lu Wang. Scaling law with learning rate annealing. *arXiv preprint arXiv:2408.11029*, 2024.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David D. Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. In *International Conference on Learning Representations (ICLR)*, 2023.

Yite Wang, Jiahao Su, Hanlin Lu, Cong Xie, Tianyi Liu, Jianbo Yuan, Haibin Lin, Ruoyu Sun, and Hongxia Yang. LEMON: Lossless model expansion. In *International Conference on Learning Representations (ICLR)*, 2024.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. In *International Conference on Learning Representations (ICLR)*, 2025.

Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. Masked structural growth for 2x faster language model pre-training. In *International Conference on Learning Representations (ICLR)*, 2024.

Qifan Yu, Xinyu Ma, Zhijian Zhuo, Minrui Wang, Deyi Liu, Shiyi Zhan, Yiyuan Ma, Liang Xiang, Xingyan Bin, and Di He. SPARKLING: Balancing signal preservation and symmetry breaking for width-progressive learning. *arXiv preprint arXiv:2602.02472*, 2026.

A Experimental setup

All experiments pretrain BERT on English Wikipedia with masked language modelling and grow the same source model to twice its width. Table 2 lists the shared configuration and Table 3 the schedules.

Table 2: Configuration shared by all experiments. "Full anneal" refers to the runs of Section 2 and the 165k-step matching of Section 4; "scheduled" to the scheduled growth of Section 4.

Models	
Source: layers, width, heads, feed-forward width	12, 384, 6, 1,536
Grown: layers, width, heads, feed-forward width	12, 768, 12, 3,072
Parameters, source / grown	33.2M / 108.9M
Per-step FLOP ratio, grown / source	3.25 (Appendix B)
Growth operators	LEMON; AKI in Section 4 only
Data	
Corpus, tokenizer	English Wikipedia, `bert-base-uncased`
Objective	MLM only, 15% masking
Sequence length, batch size	128; 256 sequences (32,768 tokens)
Tokens, full anneal / scheduled	9.01B / 7.21B
Epochs over the corpus	3.5 / 2.8
Optimization	
Optimizer	AdamW
β_1, β_2, ϵ, weight decay	0.9, 0.999, 10^{-8}, 0.01
Peak rate $\eta_{\max}$ (source and grown)	2×10^{-4}
Warmup W (sources only)	5,060 steps, linear from 0
Floor of every decay	$r\,\eta_{\max}$, $r = 0.1$
Gradient clipping, precision, dropout	1.0, bf16, 0.1
Seeds per run	3

Models and optimizer. The grown model doubles the width of the source and keeps its depth; the head dimension stays at 64, so the number of heads doubles. Source and grown model share the same peak rate and weight decay λ, so $\eta\lambda$ is the same across all runs at equal learning rate.

Schedules. Every source warms up linearly over $W = 5{,}060$ steps. At every event the learning rate returns to $\eta_{\max}$ in a single step, without a re-warmup, and both sources of a pair then follow the same decay to $0.1\,\eta_{\max}$.

Table 3: Learning-rate schedules after warmup. Every decay ends at $0.1\,\eta_{\max}$.

Experiment	Source	Before the event	Event	After the event
Sections 2, 4	annealed	cosine, ends at 165k	165k	110k cosine
Section 2	flat	constant $\eta_{\max}$	165k	110k cosine
Section 4	river-matched	constant $\eta_{\max}$	93,027	110k cosine
Section 4, scheduled	annealed	cosine, ends at 220k	g	cosine, 220k $- g$
Section 4, scheduled	river-matched	constant $\eta_{\max}$	s_g	cosine, 220k $- g$
Appendix G	annealed	cosine or linear, ends at g	g	20k, same shape
Appendix G	river-matched	constant $\eta_{\max}$	s_g	20k, same shape

Growth operators. LEMON (Wang et al., 2024) doubles the width while preserving the source's function. AKI (Chen et al., 2022) initializes the new units from the next layer's weights and does not preserve the function. LEMON runs carry the AdamW moments across the event with the scaling of Shen et al. (2022) ($m/2$, $v/4$); AKI runs reinitialize them; runs with growth withheld keep them unchanged. Both sources of a matched pair are handled the same way. Appendix E shows that resetting the moments instead does not change the conclusions.

Seeds and evaluation. Each run uses three seeds. Validation loss is evaluated every 500 steps on a held-out Wikipedia split. Plotted curves are smoothed with a centered moving average; thin lines are individual seeds, the thick line their mean.

B Compute accounting and equal-compute comparison

FLOPs per step. FLOPs per step are counted analytically, over the forward and backward passes through the attention, feed-forward and MLM-head matrix products. Doubling the width multiplies the attention projections and the feed-forward cost by four, but the attention scores and the output projection onto the vocabulary only by two, so a grown step costs 3.25 times a source step. The ratio would reach 4 only if every term scaled quadratically with the width.

Equal-compute comparison. Compute c is counted from the event, in units of source-step FLOPs, so a run that is not grown advances by one unit per step and a grown run by 3.25. Each seed's validation curve is linearly interpolated onto a common compute grid before offsets are taken, and the offsets and the interaction I are computed per seed and then averaged. The comparison window ends where the shorter run ends: 110k units, the length of the runs that are not grown.

At equal compute c, a run that is not grown is at step c of its 110k-step cosine, while a grown run is at step $c/3.25$ of the same cosine, so the two see different learning rates. The growth offset Δ_{grow} therefore also contains this difference in schedule phase. The interaction does not: both grown runs share the same phase, as do both runs that are not grown, so the phase difference cancels in $I = (\mathcal{L}_{\text{reset+grow}} - \mathcal{L}_{\text{flat+grow}}) - (\mathcal{L}_{\text{reset}} - \mathcal{L}_{\text{flat}})$.

Sensitivity to the FLOP ratio. A larger ratio compresses both grown runs along the compute axis, so at a given c the difference $\mathcal{L}_{\text{reset+grow}} - \mathcal{L}_{\text{flat+grow}}$ is read at an earlier grown step. As long as this difference decreases over grown steps, as it does throughout our runs (Figure 2), a larger ratio can only make I more positive, never negative. The conclusion of Section 2, that I is not negative, therefore holds for any ratio between 3.25 and 4.

C River progress of common schedules

Writing $u \in [0, 1]$ for the fraction of a decay phase elapsed, the reusable share of the phase is the mean of $\eta/\eta_{\max}$ over it. We compute it for the continuous schedule; the sum over discrete

steps differs by $O(1/N)$. The linear warmup contributes $W/2$ to every source and cancels within a matched pair.

Cosine and linear. A cosine from $\eta_{\max}$ to $r\,\eta_{\max}$ has $\eta(u)/\eta_{\max} = r + \frac{1-r}{2}\,(1+\cos\pi u)$, whose mean over $u \in [0,1]$ is $(1+r)/2$: 0.55 at $r = 0.1$ and 0.50 at $r = 0$ (9). A linear decay to the same floor has the same mean. This gives $s = W + (N-W)(1+r)/2 = 93{,}027$ for $N = 165{,}000$.

Inverse-proportional decay (WSD-S). The decay of Wen et al. (2025) interpolates $1/\eta$ linearly from $1/\eta_{\max}$ to $1/\eta_{\min}$. Writing $\rho = \eta_{\max}/\eta_{\min}$ for the ratio of the two rates, $\eta(u)/\eta_{\max} = 1/\big(1+(\rho-1)\,u\big)$, whose mean is $\ln\rho/(\rho-1) \approx 0.26$ at $\rho = 10$ (a floor of $0.1\,\eta_{\max}$). A segment that is 90% flat and 10% inverse decay therefore keeps $0.9 + 0.1 \times 0.26 \approx 0.93$.

Partial cosine (scheduled growth). The source's cosine spans $N' = 214{,}940$ steps after warmup, ending at step 220k. A source grown at step g has completed a fraction $u_g = (g-W)/N'$ of it, over which the mean of $\eta/\eta_{\max}$ is

$$\bar{p}(u_g) \;=\; r + \frac{1-r}{2}\left(1 + \frac{\sin\pi u_g}{\pi u_g}\right), \tag{11}$$

so the river-matched source reaches the event at $s_g = W + (g-W)\,\bar{p}(u_g)$. The forgone progress is $F = g - s_g$, and its share of the steps trained since warmup is $F/(g-W) = 1 - \bar{p}(u_g)$. Table 4 gives the resulting steps. For every complete anneal to $0.1\,\eta_{\max}$, the forgone share is 45% regardless of its length.

Table 4: River-matched event steps, with $W = 5{,}060$ and $r = 0.1$. The forgone share is $F/(g-W)$, the share of the steps trained since warmup that the reset discards.

Setting	End of decay	g	s_g	$F = g - s_g$	Forgone share
Scheduled	220k	50k	48,577	1,423	3.2%
Scheduled	220k	100k	87,550	12,450	13.1%
Scheduled	220k	150k	111,062	38,938	26.9%
Full anneal	165k	165k	93,027	71,973	45.0%
Complete anneal	50k	50k	29,777	20,223	45.0%
Complete anneal	100k	100k	57,277	42,723	45.0%
Complete anneal	150k	150k	84,777	65,223	45.0%

D THE HILL LEVEL UNDER GENERAL NOISE

In Wen et al. (2025), the hill level is $c\,\eta$ with $c = (d-1)\sigma^2/2$, for isotropic noise of variance σ^2 in each of the $d-1$ hillside directions. More generally, let A be the Hessian restricted to the hillside directions and Σ the covariance of the gradient noise there. For SGD near the valley floor, the iterate's deviation z across the valley follows $z_{k+1} = z_k - \eta(Az_k + \xi_k)$ with $\mathrm{Cov}(\xi_k) = \Sigma$. To first order in η, its stationary covariance S satisfies the Lyapunov equation

$$AS + SA \;=\; \eta\,\Sigma. \tag{12}$$

Taking the trace, and using $\mathrm{Tr}(SA) = \mathrm{Tr}(AS)$, gives $2\,\mathrm{Tr}(AS) = \eta\,\mathrm{Tr}(\Sigma)$. The hill level, the expected quadratic excess loss, is therefore

$$\tfrac{1}{2}\,\mathbb{E}\big[z^\top A z\big] \;=\; \tfrac{1}{2}\,\mathrm{Tr}(AS) \;=\; \tfrac{\eta}{4}\,\mathrm{Tr}(\Sigma), \tag{13}$$

proportional to η and independent of the hillside curvatures. For isotropic noise, this constant is half that of Wen et al. (2025). The curvatures set only how fast the hill relaxes: the slowest direction relaxes at rate $\eta\gamma$ per step, with γ the smallest eigenvalue of A, which gives the $O\big(1/(\eta\gamma)\big)$ relaxation time used in Definition 4. The main text uses only the proportionality to η, so the constant does not affect any result.

E Optimizer state and re-warmup at the reset

Two implementation choices at the event could contribute to the offsets we attribute to growth and to the schedule. Figure 5 compares each against the default, for the annealed source (left) and the flat source (right).

Optimizer state. LEMON runs carry the AdamW moments across the event with the scaling of Shen et al. (2022) (Appendix A). Resetting the moments instead changes the loss of the grown model only transiently after the event, for both the annealed and the flat source, and the curves then converge to the default, so the carried moments do not account for the growth offset.

Re-warmup. Our runs return to $\eta_{\max}$ in a single step, whereas continual pretraining often re-warms the rate over a short interval (Gupta et al., 2023). Under Section 3, a re-warmup changes how fast the hill reduction is lost, not how much of it is. Re-warming linearly over 5k or 10k steps (a single seed) delays the spike and worsens the offset rather than reducing it, so the abrupt reset is not what produces the schedule offset.

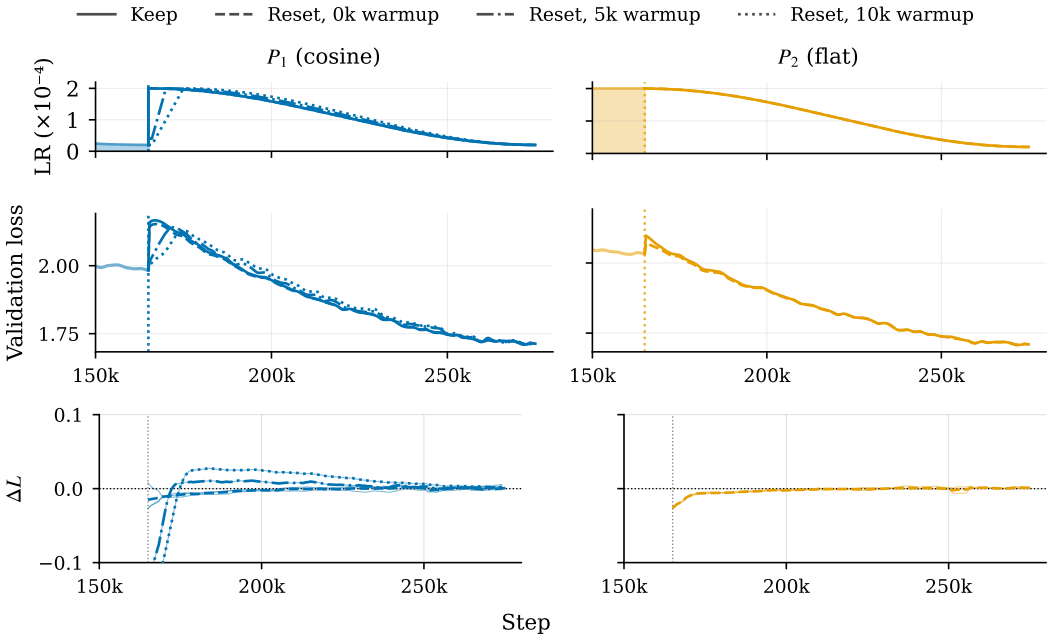

Figure 5: Optimizer-state and re-warmup controls around the growth event, for the annealed source (P_1) and the flat source (P_2) (columns). Rows: learning rate, validation loss, and the difference to the default run, which keeps the moments ("Keep"). "Reset" in the legend refers to resetting the optimizer moments, with a re-warmup of 0, 5k or 10k steps. The bottom row is a difference to the default run, not the river-matched ΔL of Section 4.

F River matching without growth and under AKI

Figure 6 shows the loss curves behind the growth-withheld and AKI panels of Figure 3.

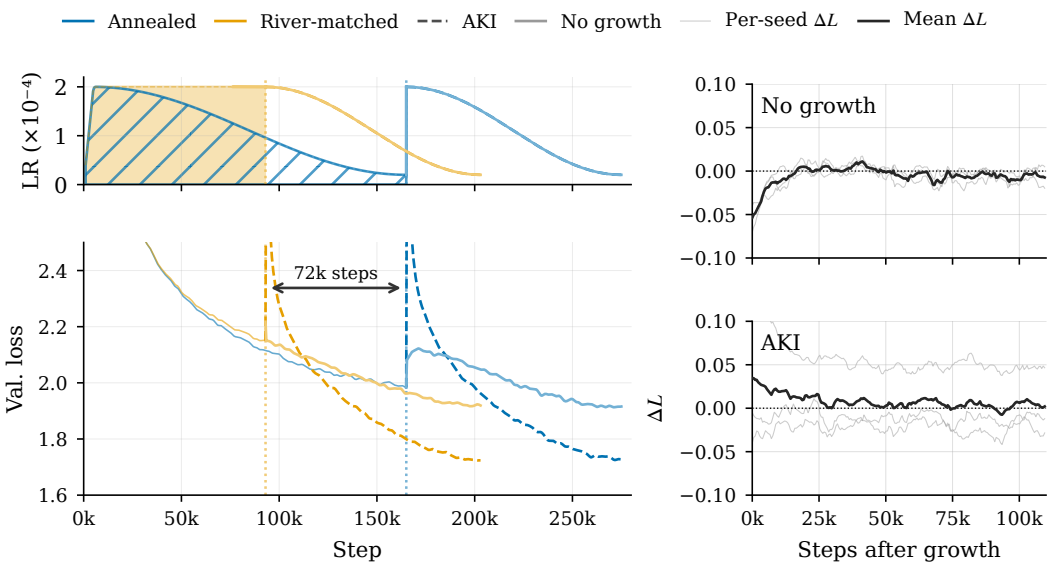

Figure 6: River matching without growth and under AKI, for the annealed and the river-matched source: after the event without growth (lighter lines) and under AKI (dashed). Left: learning rate and validation loss. Right: ΔL, annealed minus river-matched, aligned at each source's event; thin lines are individual seeds, the thick line their mean.

G Schedule shape and training length

Here each source is annealed over its whole training, ending at $g = 50$k, 100k or 150k, then grown with LEMON and trained for 20k more steps, and compared with its river-matched flat counterpart (Table 4). All three anneals forgo 45% of their steps. With a cosine decay (Figure 7), ΔL rises from its initial transient towards zero shortly after the event at all three lengths, and settles slightly below it, in favour of the annealed source.

This residual is larger than in Section 4. Here the post-event schedule decays to $0.1\,\eta_{\max}$ within 20k steps instead of 110k, so the learning rate falls again before the annealed source's hill has fully refilled at $\eta_{\max}$, in particular along the hillside directions of smallest curvature, which relax slowest (Definition 4), and the runs end before the difference has settled. What matters is that the residual has the same sign and a similar size at all three lengths, even though the anneals differ threefold in length. A linear decay to the same floor has the same mean learning rate, hence the same matched steps, and gives nearly the same curves (Figure 8). The prediction therefore behaves consistently across the length of the source's training and the shape of its decay.

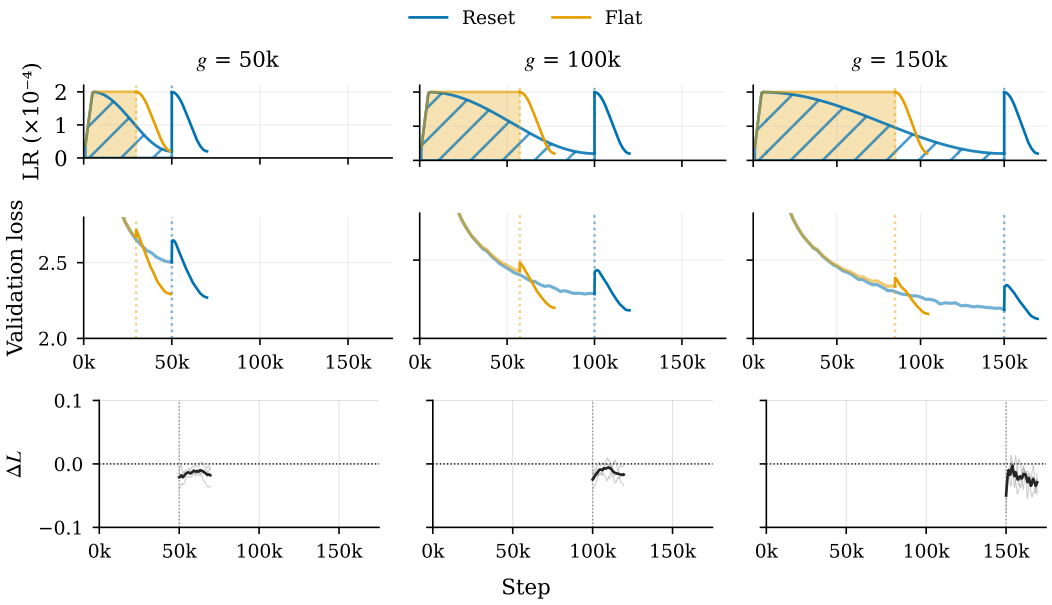

Figure 7: River matching for different training lengths before growth, with a cosine decay, at $g = 50$k, 100k and 150k (columns), annealed source (legend: Reset) against river-matched source (legend: Flat). Rows: learning rate, with equal areas up to each source's event; validation loss; ΔL, annealed minus river-matched, aligned at the events.

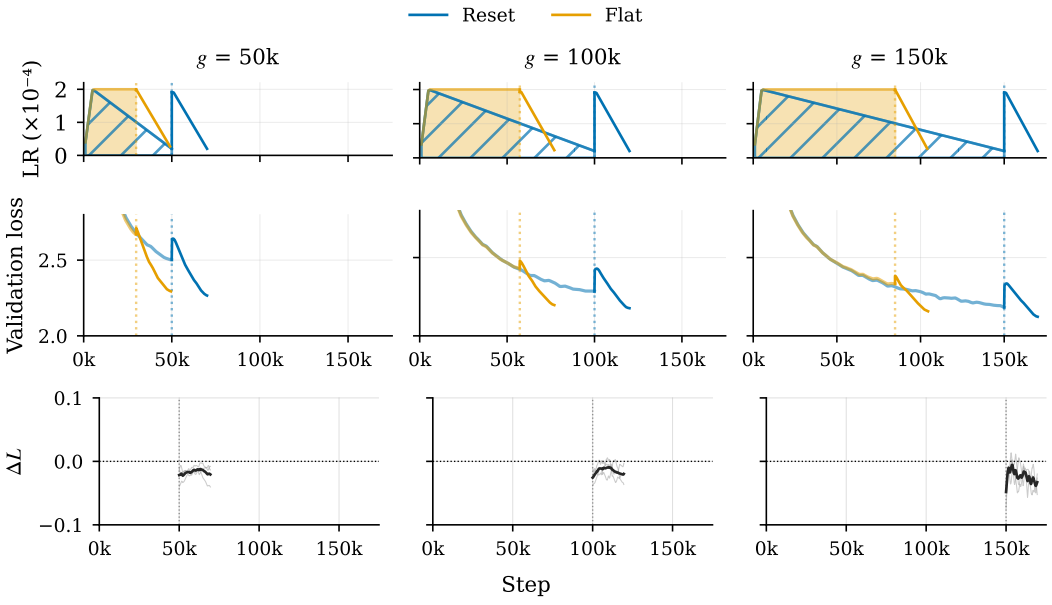

Figure 8: As Figure 7, with a linear decay.