Analysis

Align the Effective Learning Rate Before Crediting the Recipe: Misattributed Gains in LLM Pretraining

HX Research

HX Research · LLM Technical Brief · November 5, 2026

Gains in large-model pretraining recipes are usually booked to a named component — a new normalization, optimizer or schedule. Yet in normalized networks most weights are scale-invariant, and the step actually applied is the ratio of the nominal learning rate to the weight norm: the effective learning rate (ELR). Recent studies show that once ELR trajectories are matched, weight-normalized and weight-decayed training largely coincide and runs with different learning rates and norms collapse onto one curve, while systematic benchmarks find that claimed optimizer speedups shrink markedly under equal tuning budgets and end-of-training comparison. We argue that until ELR is aligned and tuning budgets are equal, most gains in the normalization, optimizer and schedule debates cannot be attributed; recipes should be designed in ELR rather than nominal learning rate, and compared by aligning first and scoring second. We formalize scale invariance, ELR, norm equilibrium, cumulative ELR, gain decomposition and apparent speedup, and contribute a five-channel framework, a five-route comparison, an ELR-matched attribution algorithm, three seeded illustrative simulations and five falsifiable forecasts.

Keywords: effective learning rate; weight decay; weight normalization; learning-rate schedules; optimizer benchmarking; gain attribution; pretraining recipes

Introduction

Over the past two years, pretraining "recipes" have become a busy research market. Normalization has changed, from LayerNorm to nGPT, which constrains every vector to the unit hypersphere[1,2]; optimizers have changed, from Adam and AdamW to Muon, built around matrix orthogonalization[3–5]; and learning-rate schedules have changed, from cosine annealing to warmup–stable–decay (WSD), linear decay to zero and schedule-free methods[6–9]. Each change comes with a sizeable claim: nGPT reports a 4- to 20-fold reduction in the training steps needed to reach the same accuracy, Muon's scaling experiments report roughly 2× computational efficiency, and ScheduleFree+ reports a 31% improvement over current schedules in long runs at 1,000 tokens per parameter (all per the papers)[2,5,10].

The question is which component those gains belong to. A paper released on 29 October 2026 challenges one class of attribution directly: the pretraining-loss gains of weight-normalized training over weight decay are driven mainly by the effective-learning-rate trajectory it induces rather than by normalization itself, and matching ELR trajectories largely collapses the loss dynamics of the two (per the paper)[11]. The authors conclude that schedules should target ELR rather than the nominal learning rate and propose AIR decay, which, in 128-billion-token mixture-of-experts pretraining and at matched cumulative learning rate, lowers terminal training loss by about 0.01 relative to cosine (per the paper)[11]. A study two months earlier reported a more general "ELR collapse": across optimizers, architectures, datasets and scales, runs with different learning rates and parameter norms collapse throughout training once their ELRs are matched, with mean collapse errors typically a few $\times\ 10^{-3}$, below seed-to-seed variation in a representative configuration (per the paper)[12].

These results did not come from nowhere. In 2017 van Laarhoven showed that, combined with normalization, L2 regularization does not regularize but instead changes the weight scale and hence the effective learning rate[13]; Zhang and colleagues listed "increasing the effective learning rate" as one of three mechanisms of weight decay[14]; Li and Arora proved that with batch normalization, weight decay and momentum one can even train with an exponentially increasing learning rate[15], and follow-up work defined the product of learning rate and weight decay as an "intrinsic learning rate"[16]. Work on spherical motion dynamics and "rotational equilibrium" further showed that weight decay drives the average angular update of each layer to a steady state[17,18]. In other words, "the step is set jointly by the nominal learning rate and the norm" is a repeatedly demonstrated mechanism; what is new is that it is now being verified quantitatively at LLM pretraining scale and used to explain recent recipe gains.

Existing discussions leave three gaps. First, recipe papers usually sweep and compare in nominal learning rate and seldom report per-group norms or angular updates, so one cannot tell whether a gain merely reflects a different ELR trajectory[11,12]. Second, optimizer comparisons often use unequal tuning budgets, a single scale or batch size, or mid-training checkpoints, and systematic benchmarks show that each of these can inflate or flip the conclusion[19–21]. Third,

enterprise continued pretraining and domain adaptation face the same questions — will re-warming the learning rate cause forgetting, which schedule to use[22,23] — but these are rarely discussed in ELR coordinates.

Our central thesis is that in modern normalized LLMs, ELR is the common coordinate linking learning-rate schedules, norm control and loss dynamics; until ELR is aligned, tuning budgets are equal and comparisons are made at the end of training, most gains in the normalization, optimizer and schedule debates cannot be attributed. Two engineering claims follow: recipes should be designed in terms of ELR trajectories rather than nominal learning rates, and recipe comparisons should align first and score second. We make three contributions:

1. A unified formalization of scale invariance, ELR, norm equilibrium under decoupled weight decay, cumulative ELR and loss curves, a three-way gain decomposition and apparent speedup (equations (1)–(6)), with a notation table, showing why the nominal learning rate is not comparable across recipes and why a loss gap of order 0.01 can be reported as a double-digit "speedup".
2. A five-class framework that sorts recipe changes by the channel through which they act on ELR, and a mechanism-level comparison of five routes — nominal schedule engineering, norm control, new optimizers, schedule-free training with weight averaging, and hyperparameter scaling laws — with evidence for and against.
3. An ELR-matched recipe-attribution algorithm (Algorithm 1), three seeded illustrative simulations and five forecasts, each with a time scale, evidence, a falsification condition and implications for enterprises.

Problem formalization and notation

Scale invariance

Let W be a parameter group (a layer's weight matrix or a neuron's weight vector) and f the training loss. If the group is followed by normalization (batch, layer or RMS normalization, or weight normalization[1,24,25]), then for any c > 0

$$f(cW) = f(W) \quad \Rightarrow \quad \nabla f(cW) = \nabla f(W)/c, \quad \langle W, \nabla f(W) \rangle = 0 \tag{1}$$

Equation (1) has two direct consequences: the gradient is inversely proportional to the weight norm, and it is orthogonal to the weights, so a gradient step can only increase the norm. Weight normalization was originally proposed precisely as a reparameterization that decouples a weight vector's length from its direction[25]. In modern Transformers, much of the freedom in the linear layers surrounded by RMSNorm, QK normalization and the like approximately satisfies equation (1) in the forward pass[26,27]; nGPT pushes the property to its limit by constraining all vectors to the unit hypersphere[2].

Effective learning rate

For a group satisfying equation (1), what changes the function is the change in direction $\hat{W} = W/\|W\|$. With nominal learning rate η_t, define the effective learning rate

$$\tilde{\eta_t} = \eta_t / \|W_t\|^p, \quad p = 2 \text{ (SGD-type); } p = 1 \text{ (Adam, Lion, Muon, etc.)} \tag{2}$$

The exponent follows from how the update depends on the norm: under equation (1) the SGD gradient shrinks as $1/\|W\|$, and converting to a change in direction divides by $\|W\|$ once more; adaptive and orthogonalizing optimizers produce updates of roughly scale-free magnitude, so the division happens once[12,13,18]. For p = 1, $\eta_t\|u_t\|$ is approximately the per-step angular update, where u_t is the update direction produced by the optimizer. The spherical-motion-dynamics paper proposes the "angular update" as a direct substitute for ELR as a measure of training progress[17]; the MuonGS paper uses $\eta_t/\|W_t\|_F$ as the ELR and analyses its accumulation[28].

Norm equilibrium under decoupled weight decay

AdamW decouples weight decay from the gradient[4]. Ignoring correlation between update and weights, the norm after one step satisfies

$$\|W_{t+1}\|^2 \approx (1 - \eta_t\lambda)^2\|W_t\|^2 + \eta_t^2\|u_t\|^2 \quad \Rightarrow \quad \|W\|^2 \approx \eta\|u\|^2/(2\lambda), \quad \theta = \eta\|u\|/\|W\|_* \approx \sqrt{(2\eta\lambda)} \tag{3}$$

Equation (3) says that with constant η and λ the norm approaches an equilibrium at which the angular update θ* depends only on ηλ and not on the update magnitude. This is the core result of "rotational equilibrium": weight decay equalizes the average rotation across layers and neurons[18]. Wang and Aitchison reach the same quantity from another direction: the weights learned by AdamW can be read as an exponential moving average of recent updates with a time constant of 1/(ηλ) steps, and the optimal time constant measured in epochs is roughly constant across model and dataset sizes[29]. Power Lines writes it as τ = B/(ηλD) and finds that the optimal τ follows a power law in tokens per parameter D/N (per the paper)[30]. Some work questions the convention that λ be proportional to η, arguing instead for a scaling with η^2 at steady state[31]. These disagreements make the point: λ and η are not two independent knobs but a pair that jointly sets the ELR.

Equation (3) also has a counter-intuitive corollary: the equilibrium is not always stable. A recent study derives an exact discrete-time law for how learning-rate schedules and weight decay interact in scale-invariant optimization, shows that a constant learning rate with weight decay cannot stably maintain an interior equilibrium, and separates contraction-dominated from expansion-dominated ELR regimes[32]; another calls the mechanism by which excessive decay drives scale-invariant norms towards zero, sharpness up and the loss into spikes "weight-norm criticality"[33].

Cumulative ELR and the loss curve

The effect of schedules on final loss has recently been captured by several empirical laws: Tissue and colleagues describe the full loss curve with the area under the learning-rate curve, S_1, and an annealing area, S_2[34]; the multi-power law adds extra decay-induced reductions to a power law in the sum of learning rates[35]. The ELR view replaces the nominal learning rate in these laws with the ELR:

$$\tilde{S}_T = \sum_{t<T} \tilde{\eta}_t, \quad L(T) \approx L_0 + A\cdot\tilde{S}_T^{-\alpha} - C\cdot\tilde{S}_T^{(2)} \tag{4}$$

where $\tilde{S}^{(2)}$ is the annealing area measured in ELR. Per the paper, replacing the learning rate with ELR lets a fitted functional scaling law transfer across norm-control methods[12]; the MuonGS paper proves that Muon without norm control accumulates ELR only logarithmically, whereas variants with weight decay or a hyperball constraint accumulate at least linearly, which helps explain the former's slower empirical progress[28]. The engineering meaning of equation (4) is that two recipes with different cumulative ELR and annealing area will produce different loss curves even if their "components" are identical.

A three-way decomposition of gains

Let baseline recipe A and candidate recipe B reach terminal losses L_A and L_B at equal compute, with observed gain $\Delta_{obs} = L_A - L_B$. Split it into three parts:

$$\Delta_{obs} = \Delta_{intr} + \Delta_{ELR} + \Delta_{tune} + \varepsilon, \quad \Delta_{intr} = L_A(\tilde{\eta}_{ref}, k) - L_B(\tilde{\eta}_{ref}, k) \tag{5}$$

Δ_{intr} is the gain of the component itself, measured with both recipes on the same reference ELR trajectory $\tilde{\eta}_{ref}$ and the same tuning budget k; Δ_{ELR} is the difference caused by different ELR trajectories; Δ_{tune} is the difference caused by unequal tuning — if the candidate tried k_B configurations and the baseline only $k_A < k_B$, then, writing m(k) for the expected best terminal loss over k trials, pure noise already gives $E[\Delta_{tune}] = m(k_A) - m(k_B) > 0$; ε is seed noise. The gain of weight normalization over weight decay lies mainly in Δ_{ELR}[11]; in optimizer benchmarking, the findings that "optimal hyperparameters for one optimizer may be suboptimal for another" and that "mid-training comparisons can flip because of learning-rate decay" correspond to Δ_{tune} and to comparison-time bias respectively[19].

From loss gaps to "apparent speedups"

Recipe papers often convert loss gaps into "the multiple of compute or tokens needed to reach the same loss". If, over the comparison range, loss follows $L(C) = E + A\cdot C^{-\alpha}$[36,37], then

$$s(\Delta) = \exp\{\ln[(L - E)/(L - E - \Delta)]/\alpha\} \tag{6}$$

When α is small and L − E is small, s is very sensitive to Δ. With α = 0.15 and L − E = 0.5, a loss gap of 0.01 corresponds to an apparent speedup of about 1.14× (illustrative parameters). A loss gap inside the range of seed noise and tuning asymmetry can therefore be reported as a "speedup of a dozen percent or more". Equation (6) is not wrong; the error is to plug in an undecomposed Δ_{obs}.

Notation

Table 1 summarizes the notation.

Table 1 | Main notation

Symbol	Meaning	Where used
W, $\hat{W}$, c	parameter group, its direction, any positive rescaling	Eq. (1)
η_t, $\tilde{\eta}_t$, p	nominal learning rate, effective learning rate, norm exponent	Eq. (2)
λ, u_t, θ*	decoupled weight decay, update direction vector, equilibrium angular update	Eq. (3)
$\tilde{S}_T$, $\tilde{S}^{(2)}$, L_0, A, C, α	cumulative ELR, ELR annealing area, loss-law parameters	Eq. (4)
Δ_{obs}, Δ_{intr}, Δ_{ELR}, Δ_{tune}, k	observed gain, component gain, ELR difference, tuning difference, tuning budget	Eq. (5)
E, s(Δ)	irreducible loss, apparent speedup multiple	Eq. (6)

Framework: five channels through which recipe changes act on ELR

If ELR is the common coordinate, every recipe change invites one question first: through which channel does it change the ELR, and what else does it change? On that basis we sort common pretraining recipe changes into five classes (Table 2). The purpose is not to deny components any value of their own, but to state, for each class, which control is needed to align ELR.

Table 2 | Five channels through which recipe changes act on ELR: representative practices, effects beyond ELR, common misattributions and the controls needed for alignment

Channel	Representative practices	Possible effects beyond ELR	Common misattribution	Control needed for alignment
C1 Nominal schedule	cosine, WSD, linear decay to zero, AIR, logarithmic	bias–variance trade-off in the annealing phase	crediting a lower or longer late-stage ELR to the shape	compare at equal cumulative ELR and

	decay		of the schedule	annealing area
C2 Norm control	weight decay, weight normalization, hyperball constraints, norm caps	numerical stability, plasticity, representational separability	crediting the ELR difference induced by norm trajectories to normalization itself	ELR transplant: make one recipe reproduce the other's ELR trajectory
C3 Update geometry	Adam→Muon, Shampoo/SOAP and other matrix preconditioners	choice of direction, sensitivity to batch size	reading results from unequal tuning and a single batch size as a universal speedup	equal tuning budgets, several batch sizes, several scales, end-of-training comparison
C4 Parameterization and scale	μP, layer-wise weight-decay scaling, QK normalization	relative allocation of ELR across layers, stability region	reading "can use a larger learning rate" as "learns faster"	report per-layer ELR and angular updates, compare under fixed stability constraints
C5 Estimator	weight averaging, checkpoint merging, terminal shrinkage averaging	replaces decay with averaging, reduces terminal variance	crediting estimator gains to the schedule	fix the trajectory, compare only how the returned model is estimated

C1 is the most direct channel: a nominal schedule is written straight into η_t. Work on linear decay to zero found that, at the optimal peak learning rate, D2Z consistently beats other schedules at compute-optimal dataset sizes; per the paper, a 610-million-parameter model trained for 80 tokens per parameter with D2Z reaches lower loss than the same model trained for 200 tokens per parameter with decay to one tenth of the peak, which the authors describe as about 60% compute savings[8]. Their explanation is exactly the trade-off, in the exponential-moving-average view, between "moving away from initial conditions" early and "averaging over noise" late[8,29]. The conclusion holds in nominal learning rate; but when the norm changes during training, the nominal schedule and the ELR schedule diverge, which is where AIR starts[11].

C2 is about the norm. Weight decay, weight normalization and hyperball constraints all change the trajectory of $\|W\|$ and hence the denominator of equation (2). Controlled interventions in the ELR-collapse study show that weight decay and hyperball constraints shape loss dynamics mainly through the ELR schedules they induce (per the paper)[12]. Norm control still matters beyond ELR: larger weight decay has been found to increase the plasticity of pretrained models, so that some base models with worse pretraining loss end up better after fine-tuning (per the paper)[38]; in near-single-epoch LLM training, weight decay has also been explained as balancing the bias–variance trade-off of stochastic optimization and improving stability[39]. These are candidate sources of Δ_{intr}, but they can only be confirmed after ELR alignment.

C3 is update geometry. Matrix optimizers such as Muon change the direction and magnitude of u_t. Moonshot's scaling experiments report about 2× computational efficiency for Muon while noting that scaling Muon up requires adding weight decay and carefully adjusting the per-parameter update scale[5] — both of which are ELR knobs. Essential AI reports that Muon retains data efficiency better far beyond the critical batch size[40]; and a very recent study proves that gradient noise introduces a bias–variance trade-off into preconditioning that can reverse optimizer rankings across batch sizes even when learning rate and momentum are optimally tuned[20]. Theory has begun to analyse Muon's training dynamics for next-token prediction in Transformers[41], and other work shows that its large-step dynamics are not captured by the classical edge-of-stability picture[42]. All of this suggests that C3 gains may be partly real, but their size depends on batch size, scale and ELR settings.

C4 concerns how ELR is distributed across layers. μP lets the learning rate transfer zero-shot from small to large models[43]; large-scale experiments across parameterizations and optimizers found that the best learning-rate scaling prescription would often have been excluded by the assumptions of prior work[44]. In modern scale-invariant architectures, training soon enters an optimizer-governed steady state in which ELR becomes width-dependent and μP transfer degrades; one study therefore proposes scaling weight decay with width to keep sublayer gains invariant[45]. Stabilizers such as QK normalization change the ceiling on usable learning rates: small-scale proxy experiments show that attention-logit growth and output-logit divergence in large models also appear in small models trained at high learning rates, and that the corresponding mitigations are equally effective there[26,27]. "Can use a larger learning rate" is not a gain in itself; it only widens the feasible ELR range.

C5 is the estimator. Weight averaging, checkpoint merging and terminal shrinkage averaging all try to replace learning-rate decay with averaging. A systematic evaluation on AlgoPerf shows that weight averaging significantly accelerates training and yields considerable efficiency gains, and shows how best to combine averaging with learning-rate annealing[46]; WSM formally links decay and merging and finds the merge window to be the most critical factor[47]; terminal shrinkage averaging reveals an interaction between the schedule and "which model is returned": once the estimator is separated out, the preferred terminal schedule changes[48]. Weight averaging has also been found to affect post-training quantization[49]. Booking C5 gains to C1 leads to choosing the wrong schedule.

Mechanism analysis and route comparison

Why nominal learning rates are not comparable

Within a recipe, $\|W\|$ stands between the nominal learning rate η and the ELR. When two recipes have different norm trajectories — one with weight decay and one with the norm fixed, one starting from a large initialization and one from a small one — the same η corresponds to different $\tilde{\eta}$. "Both recipes were trained at $\eta = 3\times10^{-4}$" therefore does not mean they were compared under equal conditions. Equation (3) gives a finer corollary: at the equilibrium of decoupled weight decay the angular update depends only on $\eta\lambda$, so doubling η and halving λ leaves the equilibrium ELR unchanged and only alters how fast equilibrium is reached[18,29]. Read this way, Power Lines' findings that the optimal λ scales linearly with batch size and that the optimal time constant follows a power law in D/N say that what is really being optimized is the time constant and the ELR, not η or λ on their own[30].

Hyperparameter scaling laws point the same way. Step Law trained more than 3,700 models and found that, at fixed model and data size, the hyperparameter landscape is convex with a broad optimum and that the optimal learning rate follows a power law in model and data size (per the paper)[50]; another study found that the optimal learning rate of Adam-style optimizers first rises and then falls with batch size, a "surge" phenomenon[51]; under WSD schedules, the classical critical-batch-size theory needs revising[52]; in ultra-sparse mixtures of experts the optimal learning rate and batch size also vary with the activation ratio[53]; and the OpenEuroLLM derivation examines whether optimal learning rates and batch sizes transfer between the stable and decay phases[54]. All of these regularities are fitted in nominal coordinates; if the norm-control method changes, the fitted coefficients may fail — which is the significance of the ELR-collapse finding that scaling laws transfer across norm-control methods once the learning rate is replaced by ELR[12].

Reinterpreting normalization gains

nGPT's 4- to 20-fold reduction in steps is one of the most striking numbers on the normalization route in recent years[2]. The nGPT training recipe released in August 2026 gives a more conservative and more checkable result: on a 30-billion-total-parameter hybrid Mamba-2–Transformer mixture-of-experts model, reaching the same validation loss takes about half as many training tokens as an unnormalized model of the same architecture trained with AdamW (per the paper); notably, the recipe itself contains ELR-level design such as logarithmic learning-rate decay and "angular update control"[55]. The approximately normalized Transformer likewise reports that approximate normalization by scalar multiplication, together with norm constraints on parameters, removes the need for weight decay and learning-rate warmup[56].

The ELR view does not deny these gains; it changes their explanation. With a fixed norm the nominal learning rate is (proportionally) the ELR, so a schedule designed in nominal coordinates is a schedule in ELR coordinates; in a weight-decay recipe, by contrast, the ELR is set jointly by η, λ and the norm dynamics, and is often too small early on and too large later as the norm shrinks. This is the sense in which the AIR paper says that the gains of weight normalization come mainly from its ELR trajectory, and that applying the same ELR trajectory to weight-decay training reproduces the corresponding training dynamics (per the paper)[11]. It also explains why warmup benefits from controlling ELR: warmup has been found to work by limiting overly large early angular updates[57], or by moving the network into better-conditioned regions so that it tolerates a larger target learning rate[58].

Tuning asymmetry in the optimizer debate

Optimizer comparison is where Δ_{tune} most readily appears. The authors of the AlgoPerf benchmark noted in 2023 that the community cannot reliably identify improvements to training algorithms, citing how to decide when training is complete, the sensitivity of measurements to workload details and how to compare algorithms that require tuning fairly[21]. The analysis of the first competition shows that the winner under the external-tuning ruleset used Distributed Shampoo, while the winner under the self-tuning ruleset was Schedule-Free AdamW[9,59]; in the new AlgoPerf V1 results released in October 2026, Schedule-Free AdamW again ranked top among 13 submissions under the self-tuning ruleset, which requires a single configuration across all workloads, and learning rate and momentum were highly sensitive across domains and algorithms (per the paper)[60].

More direct evidence comes from a systematic study of ten optimizers: the authors identify two methodological flaws behind earlier comparisons claiming 1.4–2× speedups — unequal tuning and limited or misleading evaluation setups; with rigorous tuning and end-of-training comparison, the actual speedup of many optimizers over well-tuned baselines is lower than claimed and declines with scale to only about 1.1× at 1.2 billion parameters, and rankings at intermediate checkpoints can flip because of learning-rate decay (per the paper)[19]. This matches the decomposition in equation (5): if the candidate optimizer was tuned more, or compared before decay began, Δ_{obs} contains Δ_{tune} and timing bias.

Schedules: the late-stage ELR is the main source of gains

If the loss curve is mainly set by cumulative ELR and annealing area (equation (4)), then differences between schedules should show up mainly at the end. The evidence agrees. WSD schedules keep the loss elevated during the stable phase and drop it sharply during decay, which the river-valley hypothesis explains as progress along the river floor during the stable phase and descent to the valley floor

during decay[7,61]; work on the cooldown phase finds that cooldown shape reflects a bias–variance trade-off whose effect is comparable to tuning AdamW hyperparameters[62]; a constant learning rate with cooldown scales as predictably as cosine and allows training runs to be reused, cutting the cost of scaling experiments[63]. The AIR paper further finds that a shared late-stage ELR schedule attenuates the effect of earlier learning-rate choices, a key source of its gains (per the paper)[11]. Work on loss-curve "collapse" corroborates this from the other side: compute-optimally trained models of different sizes collapse onto one curve after normalization, more tightly with learning-rate decay, and the collapse breaks when hyperparameters are scaled suboptimally[64,65].

Schedule-free and anytime training are, at heart, attempts to approximate with averaging or a particular decay shape an ELR trajectory that "suits any stopping point": Schedule-Free unifies scheduling and iterate averaging[9] and has been explained as navigating the river-valley structure without a decay phase[66]; theoretical work on anytime pretraining stresses the role of weight averaging in reaching minimax convergence rates[67]; WSqD replaces WSD's constant phase with a shifted inverse-square-root base[68]; AIR's $1/\sqrt{t}$ decay belongs to the same family[11]. Some work tries to learn schedules online: SOLAR learns bounded, state-dependent residual corrections relative to a base schedule[69]; "reactive optimizers" go further, letting a language-model agent inspect training dynamics during a single run and rewrite optimizer code, and their decisions most often change learning-rate schedules, optimizer structure and numerical precision[70]. If the decision variable of such methods is the nominal learning rate, they too are exposed to norm drift.

Stability: the cost of too large an ELR

More ELR is not always better. In theory, the fixed-norm updates of normalized SGD induce an ELR that grows as gradients shrink: it accelerates learning early on, but at finite step size ultimately breaks the acceleration and leads to marginal stability[71]. In practice, loss spikes are often tied to logit growth: an October 2026 study finds that delayed "time-bomb" spikes appearing only after long training are preceded by rapidly growing quantization error in the language-model head, and that lower numerical precision accelerates the spikes while higher precision improves stability (per the paper)[72]; weight-norm criticality supplies a spike mechanism from the norm side[33]. Technical reports of leading open models all put stability at the centre of recipe design: OLMo 2 reports architecture and recipe changes focused on training stability and introduces a specialized data mixture during the annealing phase of pretraining[73]; Llama 3 reports annealing on small amounts of high-quality data at the end of pretraining[74]; DeepSeek-V3 reports a learning-rate plan that is constant first and then decays in stages[75]. All of these act directly on the late-stage ELR, yet the reports rarely give the ELR itself.

Counter-evidence: ELR is not everything

The ELR view has three limits. First, it applies mainly to scale-invariant parameter groups; embeddings, output heads, biases and normalization gains do not satisfy equation (1), their learning rate remains the nominal one, and they are often the source of instability[45,72]. Second, components may contribute beyond ELR: weight decay's effects on plasticity and representational separability[38], the data efficiency of matrix optimizers at large batch sizes[20,40] and nGPT's token efficiency at the 30-billion-parameter scale[55] are not yet fully explained by ELR. Third, the precision of ELR collapse depends on normalization design and the timescale of learning-rate–norm variation[12], and in repeated-data, multi-epoch settings the optimal batch size and regularization also change[76]. Our claim is therefore "align before attributing", not "every gain is ELR".

Table 3 | Mechanism-level comparison of five routes to better recipes (gains as reported by the papers)

Route	Main mechanism	Reported gain	Evidence remaining after ELR alignment	Main risk	Conditions of use
R1 Nominal schedule engineering (D2Z, WSD, AIR)	changes cumulative ELR and annealing area	D2Z ~60% compute savings; AIR terminal loss ~0.01 lower	AIR still beats cosine at matched cumulative learning rate	norm drift makes the nominal schedule deviate from ELR	known stopping point, or branchable cooldowns
R2 Norm control (WD, WN, hyperball, MuonGS)	changes the $\|W\|$ trajectory and thus ELR	nGPT 4–20× fewer steps; ~half the tokens with the training recipe; MuonGS ~15%	WN and WD dynamics largely coincide after ELR alignment	too small a norm causes spikes; plasticity changes	scale-invariant groups dominate
R3 New optimizers (Muon, SOAP, Shampoo)	changes update geometry and preconditioning	Muon ~2×; ~1.1× at 1.2B under rigorous comparison	partly retained, varies with scale and batch size	tuning asymmetry, batch-size dependence, wall-clock overhead	large batches, mostly matrix parameters
R4 Schedule-free plus weight averaging	replaces decay with averaging	ScheduleFree+ 31% better at 1,000 TPP; first in AlgoPerf self-tuning	averaging accelerates markedly; best combined with annealing	memory overhead; extrapolation to long runs	unknown stopping point, continual training
R5 Hyperparameter scaling laws and μP transfer	predicts optimal η, λ, B	Step Law predicted optimum within 0.094% of exhaustive search	coefficients fitted in nominal coordinates may fail when norm control changes	transfer degrades at steady state	fixed recipe, only scale changes

The numbers in Table 3 are self-reported by the papers[2,5,8,10,11,19,28,50,55,60]; their baselines and conditions differ and they cannot be added up across rows. What Table 3 conveys is that, for at least four of the five routes, part or all of the gain can be explained by the ELR trajectory; the remainder must be confirmed separately with Algorithm 1.

Method: ELR-matched recipe attribution

We now condense the analysis into an executable procedure. The goal of Algorithm 1 is, given a baseline recipe and a candidate change, to split the observed gain into the three parts of equation (5) within a limited training budget, and to decide on that basis whether to adopt the change.

Algorithm 1 | ELR-Matched Recipe Attribution (EMRA)

Input: baseline recipe A and candidate recipe B; a set of scales {N} (at least two) and of data ratios {D/N} (at least two); a set of batch sizes {B}; tuning budget k per recipe; number of seeds r; E and α for converting to apparent speedup

Output: estimates and intervals for Δ_{obs}, Δ_{intr}, Δ_{ELR}, Δ_{tune}; trends with scale; an adopt/reject decision; a recipe card

1: Instrument: for every scale-invariant parameter group log $\|W_t\|$, $\|u_t\|$ and the angular update, and compute $\tilde{\eta}_t$ by equation (2); for non-scale-invariant parameters such as embeddings, output heads and gains, log the nominal learning rate separately

2: Equal-budget tuning: A and B use the same k and log-learning-rate search ranges of the same width, with end-of-training loss as the objective; log every trial for estimating m(k)

3: ELR transplant: taking A's $\tilde{\eta}$ trajectory as reference, solve for B's nominal learning rate $\eta_{B,t} = \tilde{\eta}_{A,t} \cdot \|W_{B,t}\|^p$ (adjusted online by the current norm) to obtain B′; if needed transplant in the reverse direction to obtain A′ and average the two

4: Score: at the end of training, over at least $r \geq 2$ seeds, compute $\Delta_{obs} = L_A - L_B$ and $\Delta_{intr} = L_A - L_{B'}$; $\Delta_{ELR} = \Delta_{obs} - \Delta_{intr} - \Delta_{tune}$, where Δ_{tune} is estimated from the trial logs of step 2 via m(k)

5: Repeat steps 2–4 across scales and batch sizes, testing whether Δ_{intr} shrinks with N or flips with B

6: Convert and decide: convert Δ_{intr} (not Δ_{obs}) to apparent speedup with equation (6); adopt B only if, at the largest scale, Δ_{intr} exceeds twice the seed standard deviation and downstream fine-tuning does not degrade

7: Write the recipe card: record the ELR trajectory, cumulative ELR and annealing area (equation (4)), tuning budget and number of trials, comparison time, and how parameter groups that violate equation (1) were handled

Complexity and scope: With n_N, n_D and n_B levels of scale, data ratio and batch size, the number of training runs is about $2 \cdot n_N \cdot n_D \cdot n_B \cdot k \cdot r$, plus $n_N \cdot n_D \cdot n_B \cdot r$ for ELR transplants; instrumentation costs O(number of parameter groups) per step. Applies to normalization-dominated Transformer-type models; for models with a large share of non-scale-invariant parameters, step 3 aligns only partially, which must be noted on the recipe card

Algorithm 1 embodies three design intentions. First, the "ELR transplant" of step 3 corresponds directly to the controlled experiments in the AIR and ELR-collapse studies: make one recipe reproduce another's ELR trajectory, and if the loss dynamics then coincide, the gain lies mainly in Δ_{ELR}[11,12]. Second, steps 2 and 5 encode the lessons of the optimizer-benchmarking literature: equal tuning budgets, end-of-training comparison and checks across scales and batch sizes[19,20,60]. Third, step 6 brings downstream fine-tuning into the decision, because pretraining loss is not the only objective and components such as weight decay change plasticity[38].

Illustrative simulations

All numbers below are illustrative and show how equations (1)–(6) interact. The model is a 64-dimensional scale-invariant regression (prediction $x \cdot w/\|w\|$, with label noise, batch size 8, 3,000 training steps); the script and random seed (20261105) are in src/figs.py. They do not represent measurements of any real large-model training.

ELR collapse of weight normalization and weight decay. We compare three runs: weight decay (nominal learning rate 0.6, $\lambda = 0.02$, initial norm 3); weight normalization with the per-step learning rate set to the $\tilde{\eta}_t$ measured in the weight-decay run ("matched ELR"); and weight normalization with the same nominal learning rate of 0.6 ("same nominal LR"). The smoothed loss curve of the matched-ELR run differs from the weight-decay run by 0.0023 on average, less than the 0.0092 obtained by rerunning weight decay with a different seed; the same-nominal-LR run differs by 1.26 on average, with a terminal excess loss of 1.398, an order of magnitude above weight decay's 0.139 (Fig. 1). The ELR of the weight-decay run moves from 0.067 at the start to about 0.042 at the end. In this toy model, then, whether normalization looks better or worse depends entirely on the ELR it is given, not on normalization itself.

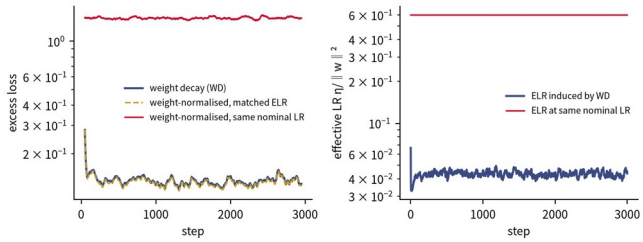


Fig. 1 | After ELR alignment, the loss curves of weight normalization and weight decay coincide. Left: excess loss of the three runs (100-step moving average); right: the ELR induced by weight decay and the ELR at the same nominal learning rate. Illustrative model, fixed random seed 20261105, computed by the HX Research Team.

"Gains" manufactured by tuning asymmetry. Suppose two optimizers are of exactly equal quality: terminal loss is a quadratic function of the log learning rate (curvature 0.08), the location of the optimum is unknown (standard deviation 0.35), and each run carries seed noise with standard

deviation 0.006. Over 20,000 repeated comparisons: when the candidate tries 12 configurations and the baseline 3, the candidate "wins" by 0.0075 on average, and 63.4% of comparisons show a win above 0.005; when the candidate tries 12 and the baseline uses only defaults, the average win is 0.0145 and 73.6% exceed 0.005; when each side tries 6, the average win is zero, yet 23.8% of comparisons still show a win above 0.005 because of noise (Fig. 2). By equation (6) with $\alpha = 0.15$ and $L - E = 0.5$, loss gaps of 0.0075 and 0.0145 correspond to apparent speedups of about 1.11× and 1.22×.

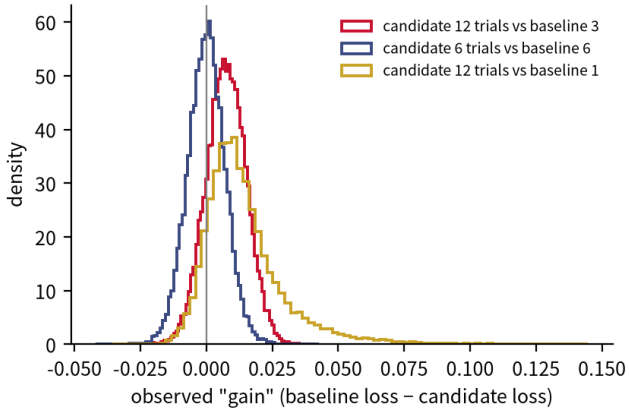


Fig. 2 | With unequal tuning budgets, two optimizers of identical quality show systematic "gains". The three curves are distributions of observed gains over 20,000 simulated comparisons; with equal budgets the distribution is centred on zero, but its tail still crosses common reporting thresholds. Illustrative parameters, fixed random seed 20261105, computed by the HX Research Team.

A shared late-stage ELR weakens early choices. On weight-normalized runs we compare two schedule families: cosine decay to 10% of peak, and a shared tail that holds the peak for the first 60% of training and then decays linearly from 0.3 to 0.003 over the last 40%. Early peak ELRs are 0.15, 0.3, 0.6 and 1.2. Under cosine the terminal excess losses are 0.032, 0.077, 0.220 and 0.676, a spread of about 21× between worst and best; under the shared tail the terminal excess loss is 0.031 at all four peaks, slightly better than the best cosine result (Fig. 3). The toy model's "memory" is shorter than the decay phase, so the attenuation is nearly complete; for real large models the paper reports attenuation, not elimination[11].

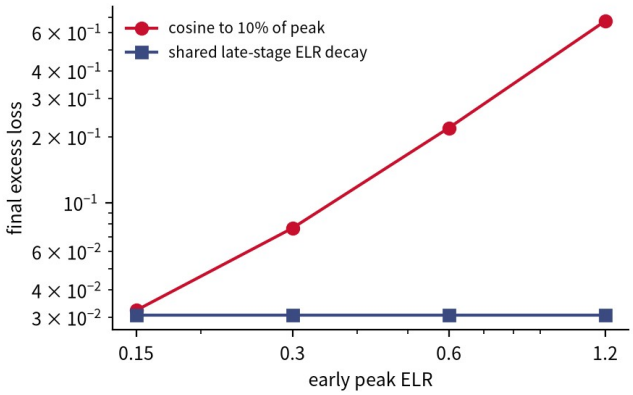


Fig. 3 | A shared late-stage ELR decay makes terminal loss insensitive to the early peak. Terminal loss under cosine decay varies strongly with the peak; the four points under the shared tail coincide. Illustrative model, fixed random seed 20261105, computed by the HX Research Team.

Table 4 | Summary of illustrative simulations (fixed random seed 20261105)

Simulation	Key setting	Result	Equations
ELR collapse	64-dimensional scale-invariant regression, WD ($\eta = 0.6$, $\lambda = 0.02$) versus WN	matched-ELR gap 0.0023 < seed gap 0.0092; same-nominal-LR gap 1.26	Eqs. (1)–(3)
Tuning asymmetry	identical true quality; 12 vs 3, 12 vs 1, 6 vs 6 trials	mean "gain" 0.0075, 0.0145, 0; apparent speedup 1.11×, 1.22×, 1.00×	Eqs. (5)–(6)
Late-stage ELR	peaks 0.15–1.2; cosine to 10% versus shared tail	cosine terminal losses differ ~21×; shared tail 0.031 at all four peaks	Eq. (4)

Table 4 makes no claim about any real system: the same component can look better, worse or no different under different ELR trajectories and tuning budgets, so comparison is meaningful only after alignment.

Forecasts and research agenda

The following are views of the HX Research Team, not statements of fact.

Table 5 | Five forecasts (opinion)

No.	Forecast	Time scale	Main basis	Falsification condition	Implication for enterprises
1	Technical reports of major open models will begin to report per-group weight norms or ELR/angular-update trajectories, not only nominal learning-rate schedules	12–24 months	ELR collapse[12]; AIR[11]; angular-update control[55]	by end-2028, no technical report of a major open-weight model gives norm or ELR curves	when training or continuing pretraining in-house, make ELR logs a mandatory field of the training card
2	Under equal-budget,	6–18 months	systematic benchmark	an optimizer keeps a	when assessing vendor

	end-of-training, multi-scale comparison protocols, optimizer claims of "more than 1.5× speedup" above a billion parameters will generally shrink to within 1.2×		1.1×[19]; batch-size flips[20]; AlgoPerf[60]	≥1.5× advantage in strict equal-budget comparisons at several scales above a billion parameters and is independently reproduced	"speedup" claims, ask for tuning budget and comparison time
3	Schedules will increasingly be designed directly in ELR or angular-update space, and anytime schedules plus weight averaging will become the default for continued pretraining	12–24 months	AIR $1/\sqrt{t}$[11]; WSqD[68]; anytime pretraining[67]; ScheduleFree+[10]	within two years the default schedules of mainstream open-source training frameworks remain mainly nominal cosine, with no ELR option	in continued pretraining, design the re-warming amplitude in ELR terms to reduce the risk of forgetting
4	Claims that "normalization itself brings large speedups" will be weakened by more ELR-aligned experiments, and norm-control methods will be seen as different ways of implementing an ELR schedule	12–24 months	WN and WD dynamics coincide[11,12]; MuonGS cumulative ELR[28]	at ≥7B parameters and under strict ELR alignment, the normalization route keeps an advantage several times seed noise and is independently reproduced	before adopting an architectural change, run an ELR-transplant control
5	Acceptance of enterprise domain continued pretraining and fine-tuning will include "downstream plasticity" together with ELR logs, not only pretraining loss	6–12 months	plasticity[38]; re-warming and replay in continued pretraining[22]; infinite schedules[23]	within a year, acceptance templates of major enterprise training platforms still contain only pretraining loss and benchmark scores	on train-while-serving platforms, write ELR and downstream fine-tuning results into the record of every model update

The research agenda has four open questions. First, how should the "effective learning rate" of non-scale-invariant parameter groups (embeddings, output heads, gains) be defined, and how does it interact with the ELR of the scale-invariant part? Work on loss spikes suggests the output head is often the bottleneck[72]. Second, can ELR alignment be extended to matrix optimizers? Muon's updates are orthogonalized, so its "angular update" needs to be defined under the spectral rather than the Frobenius norm, and analysis has only just begun[28,41]. Third, are hyperparameter scaling laws in ELR coordinates more robust than in nominal coordinates, particularly for mixtures of experts and repeated data[53,76]? Fourth, can the decomposition of equation (5) be turned into an auditable public protocol, so that recipe papers report tuning budgets and ELR trajectories the way they report seed counts?

Key points:

1. Equation (2): the step of scale-invariant parameters is nominal learning rate over norm.
2. Equation (3): the equilibrium angular update is set by $\eta\lambda$ alone.
3. Equation (4): cumulative ELR and annealing area set the loss curve; the tail matters most.
4. Equation (5): separate ELR difference and tuning asymmetry before crediting a component.
5. Equation (6): a loss gap of order 0.01 can become a double-digit "speedup".

Where common industry views may be wrong

"Normalization makes training faster." More precisely, normalization changes the mapping between nominal learning rate and ELR, so that the same nominal schedule corresponds to a more suitable ELR trajectory. After ELR alignment, the training dynamics of weight normalization and weight decay largely coincide[11].

"The new optimizer gives a 2× speedup." With rigorous tuning and end-of-training comparison, many optimizers' speedups fall below the claimed values and shrink with scale[19]; the best optimizer may also change with batch size[20]. The 2× may be real, but it is a conditional number.

"You can judge a recipe from the mid-training loss curve." WSD-type schedules keep the loss elevated in the stable phase and reveal real progress only in the decay phase[61]; rankings at intermediate checkpoints may flip because of learning-rate decay[19]. Comparisons should be made at the end of training or at aligned cooldown points.

"Hyperparameters transfer directly from small models." μP only guarantees consistent update sizes near initialization; once training enters the optimizer-governed steady state, ELR varies with width and weight-decay scaling is needed[43,45].

"Lower pretraining loss is always better." A base model with larger weight decay may have slightly worse pretraining loss yet do better after fine-tuning[38]; weight averaging also affects post-training quantization[49]. Recipe acceptance criteria should include downstream adaptation.

Limitations

First, the simulations in Section 6 use a low-dimensional scale-invariant regression; they illustrate mechanisms but cannot give numbers for large models, and the "complete attenuation" by the late-stage ELR is a consequence of the toy model's short memory. Second, key evidence such as ELR collapse and AIR comes from 2026 preprints; conclusions are as self-reported by the papers and await further independent replication[11,12]. Third, the equilibrium analysis of equation (3) ignores update–weight correlation and momentum, so actual equilibrium values will deviate; other work also argues for different scalings of λ[31]. Fourth, Algorithm 1 is not cheap: ELR transplantation needs at least one additional set of runs, and executing every step at the largest scale is unrealistic for most teams; in practice one can run it at two smaller scales and extrapolate, but the extrapolation itself is risky.

Conclusion

ELR is not a new concept, but it is turning from a theoretical footnote into the common coordinate of LLM pretraining recipes. After ELR alignment, most of the difference between weight normalization and weight decay disappears; after rigorous tuning and end-of-training comparison, the speedups of new optimizers shrink markedly; after sharing a late-stage ELR, early learning-rate choices matter much less. These three lines of evidence point to the same conclusion: until ELR is aligned and tuning budgets are equal, most gains in the recipe debates cannot be attributed.

For enterprises the implications are concrete. When training or continuing pretraining in-house, log ELR, not only the nominal learning rate; when assessing external "speedups" and "new recipes", ask about tuning budgets, comparison time and whether ELR was aligned; in domain continued pretraining, design the re-warming amplitude in ELR rather than nominal learning-rate terms[22,23]. According to the company website, Deep Hope S1 uses a train-while-serving architecture in which statistics, machine learning, deep learning and large models can be deployed and trained on one on-premises platform, with professional capability built through industry LLM distillation[77]. On such platforms we recommend writing the ELR trajectory, tuning budget and downstream fine-tuning results into the training card of every model update, so that recipe changes can be reviewed and rolled back.

Data availability

No new experimental data were generated. Figures quoted come from the listed references and the company website; the simulations were generated by script under the stated assumptions.

Code availability

The simulation and plotting script for Figs. 1–3 is src/figs.py (Python, seed 20261105), provided with this article's package.

Competing interests

The authors are employees of Great Unicorn Intelligence Technology (Shenzhen) Co., Ltd. Deep Hope S1, mentioned in this article, is a public product of the company.

References

1. Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer Normalization. Preprint at https://arxiv.org/abs/1607.06450 (2016).
2. Loshchilov, I., Hsieh, C.-P., Sun, S. & Ginsburg, B. nGPT: Normalized Transformer with Representation Learning on the Hypersphere. Preprint at https://arxiv.org/abs/2410.01131 (2024).
3. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. In *Proc. International Conference on Learning Representations* (2015); https://arxiv.org/abs/1412.6980.
4. Loshchilov, I. & Hutter, F. Decoupled Weight Decay Regularization. In *Proc. International Conference on Learning Representations* (2019); https://arxiv.org/abs/1711.05101.
5. Liu, J. et al. Muon is Scalable for LLM Training. Preprint at https://arxiv.org/abs/2502.16982 (2025).
6. Loshchilov, I. & Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proc. International Conference on Learning Representations* (2017); https://arxiv.org/abs/1608.03983.
7. Hu, S. et al. MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies. Preprint at https://arxiv.org/abs/2404.06395 (2024).
8. Bergsma, S. et al. Straight to Zero: Why Linearly Decaying the Learning Rate to Zero Works Best for LLMs. Preprint at https://arxiv.org/abs/2502.15938 (2025).
9. Defazio, A. et al. The Road Less Scheduled. In *Advances in Neural Information Processing Systems* **37** (2024); https://arxiv.org/abs/2405.15682.
10. Defazio, A. ScheduleFree+: Scaling Learning-Rate-Free & Schedule-Free Learning to Large Language Models. Preprint at https://arxiv.org/abs/2605.19095 (2026).
11. Yu, Q., Ma, X., Bin, X. & He, D. Your LR Schedule May Not Be Principled: Effective Learning Rate Scheduling in Weight-Normalized Training. Preprint at https://arxiv.org/abs/2610.40362 (2026).
12. Liu, Z. et al. Effective Learning Rate Governs Loss Dynamics in Language Model Pretraining. Preprint at https://arxiv.org/abs/2608.24814 (2026).
13. van Laarhoven, T. L2 Regularization versus Batch and Weight Normalization. Preprint at https://arxiv.org/abs/1706.05350 (2017).
14. Zhang, G., Wang, C., Xu, B. & Grosse, R. Three Mechanisms of Weight Decay Regularization. In *Proc. International Conference on Learning Representations* (2019); https://arxiv.org/abs/1810.12281.
15. Li, Z. & Arora, S. An Exponential Learning Rate Schedule for Deep Learning. In *Proc. International Conference on Learning Representations* (2020); https://arxiv.org/abs/1910.07454.
16. Li, Z., Lyu, K. & Arora, S. Reconciling Modern Deep Learning with Traditional Optimization Analyses: The Intrinsic Learning Rate. In *Advances in Neural Information Processing Systems* **33** (2020); https://arxiv.org/abs/2010.02916.
17. Wan, R., Zhu, Z., Zhang, X. & Sun, J. Spherical Motion Dynamics: Learning Dynamics of Neural Network with Normalization, Weight Decay, and SGD. In *Advances in Neural Information Processing Systems* **34** (2021); https://arxiv.org/abs/2006.08419.
18. Kosson, A., Messmer, B. & Jaggi, M. Rotational Equilibrium: How Weight Decay Balances Learning Across Neural Networks. In *Proc. 41th International Conference on Machine Learning* (PMLR, 2024); https://arxiv.org/abs/2305.17212.
19. Wen, K., Hall, D., Ma, T. & Liang, P. Fantastic Pretraining Optimizers and Where to Find Them. Preprint at https://arxiv.org/abs/2509.02046 (2025).

20. Dang, X., Wen, K. & Malladi, S. The Best Optimizer Depends on Batch Size. Preprint at https://arxiv.org/abs/2610.08975 (2026).
21. Dahl, G. E. et al. Benchmarking Neural Network Training Algorithms. Preprint at https://arxiv.org/abs/2306.07179 (2023).
22. Ibrahim, A. et al. Simple and Scalable Strategies to Continually Pre-train Large Language Models. Preprint at https://arxiv.org/abs/2403.08763 (2024).
23. Singh, V. et al. Beyond Cosine Decay: On the effectiveness of Infinite Learning Rate Schedule for Continual Pre-training. Preprint at https://arxiv.org/abs/2503.02844 (2025).
24. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. 32th International Conference on Machine Learning* (PMLR, 2015); https://arxiv.org/abs/1502.03167.
25. Salimans, T. & Kingma, D. P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems* **29** (2016); https://arxiv.org/abs/1602.07868.
26. Dehghani, M. et al. Scaling Vision Transformers to 22 Billion Parameters. In *Proc. 40th International Conference on Machine Learning* (PMLR, 2023); https://arxiv.org/abs/2302.05442.
27. Wortsman, M. et al. Small-scale proxies for large-scale Transformer training instabilities. In *Proc. International Conference on Learning Representations* (2024); https://arxiv.org/abs/2309.14322.
28. Glentis, A., Li, D. & Hong, M. MuonGS: Pretraining LLMs with Relaxed Weight-Norm Control. Preprint at https://arxiv.org/abs/2610.37647 (2026).
29. Wang, X. & Aitchison, L. How to set AdamW's weight decay as you scale model and dataset size. Preprint at https://arxiv.org/abs/2405.13698 (2024).
30. Bergsma, S. et al. Power Lines: Scaling Laws for Weight Decay and Batch Size in LLM Pre-training. Preprint at https://arxiv.org/abs/2505.13738 (2025).
31. Chou, J. C.-C. Correction of Decoupled Weight Decay. Preprint at https://arxiv.org/abs/2512.08217 (2025).
32. Amin, H., Chang, W.-K. & Khanna, R. When Does Scale-Invariant Optimization Become Unstable? An Exact Schedule Law with Weight Decay. Preprint at https://arxiv.org/abs/2609.09116 (2026).
33. Li, X., Zhou, Z. & Xu, Z.-Q. J. Weight-norm Criticality: A Mechanism for Loss Spikes Induced by the Normalization and Weight Decay. Preprint at https://arxiv.org/abs/2607.21005 (2026).
34. Tissue, H., Wang, V. & Wang, L. Scaling Law with Learning Rate Annealing. Preprint at https://arxiv.org/abs/2408.11029 (2024).
35. Luo, K. et al. A Multi-Power Law for Loss Curve Prediction Across Learning Rate Schedules. Preprint at https://arxiv.org/abs/2503.12811 (2025).
36. Kaplan, J. et al. Scaling Laws for Neural Language Models. Preprint at https://arxiv.org/abs/2001.08361 (2020).
37. Hoffmann, J. et al. Training Compute-Optimal Large Language Models. In *Advances in Neural Information Processing Systems* **35** (2022); https://arxiv.org/abs/2203.15556.
38. Han, T., Bordt, S., Zhang, H. & Kakade, S. Weight Decay Improves Language Model Plasticity. Preprint at https://arxiv.org/abs/2602.11137 (2026).
39. D'Angelo, F., Andriushchenko, M., Varre, A. & Flammarion, N. Why Do We Need Weight Decay in Modern Deep Learning? Preprint at https://arxiv.org/abs/2310.04415 (2023).
40. Essential AI et al. Practical Efficiency of Muon for Pretraining. Preprint at https://arxiv.org/abs/2505.02222 (2025).
41. Wu, S., Yang, Y., Huang, G. & Li, S. Theoretical Analysis of Muon on Transformers for Next-Token Prediction. Preprint at https://arxiv.org/abs/2611.04465 (2026).
42. Chen, Y., Zhao, Q., Xu, X. & Liu, F. Muon Sublates the Edge of Stability in LLM Pretraining. Preprint at https://arxiv.org/abs/2609.34915 (2026).
43. Yang, G. et al. Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer. In *Advances in Neural Information Processing Systems* **34** (2021); https://arxiv.org/abs/2203.03466.
44. Everett, K. et al. Scaling Exponents Across Parameterizations and Optimizers. In *Proc. 41th International Conference on Machine Learning* (PMLR, 2024); https://arxiv.org/abs/2407.05872.
45. Fan, Z., Liu, Y., Zhao, Q., Yuan, A. & Gu, Q. Robust Layerwise Scaling Rules by Proper Weight Decay Tuning. Preprint at https://arxiv.org/abs/2510.15262 (2025).
46. Ajroldi, N., Orvieto, A. & Geiping, J. When, Where and Why to Average Weights? Preprint at https://arxiv.org/abs/2502.06761 (2025).
47. Tian, C. et al. WSM: Decay-Free Learning Rate Schedule via Checkpoint Merging for LLM Pre-training. Preprint at https://arxiv.org/abs/2507.17634 (2025).
48. Ousherovitch, A. & Wang, Y. Terminal Shrinkage Averaging Reveals a Schedule-Estimator Interaction in LLM Pretraining. Preprint at https://arxiv.org/abs/2609.25482 (2026).
49. Wang, H. et al. Understanding the Weight Averaging Mechanism in LLM Training for Post-Training Quantization. Preprint at https://arxiv.org/abs/2610.05329 (2026).
50. Li, H. et al. Predictable Scale: Part I, Step Law -- Optimal Hyperparameter Scaling Law in Large Language Model Pretraining. Preprint at https://arxiv.org/abs/2503.04715 (2025).
51. Li, S. et al. Surge Phenomenon in Optimal Learning Rate and Batch Size Scaling. Preprint at https://arxiv.org/abs/2405.14578 (2024).
52. Zhou, Y. et al. How to Set the Batch Size for Large-Scale Pre-training? Preprint at https://arxiv.org/abs/2601.05034 (2026).
53. Tian, C. et al. Hyperparameter Scaling Laws Across MoE Sparsity. Preprint at https://arxiv.org/abs/2609.08690 (2026).
54. Ajroldi, N. et al. Deriving Scaling Laws for OpenEuroLLM Models: Learning Rate, Batch Size and Loss. Preprint at https://arxiv.org/abs/2608.28308 (2026).
55. Loshchilov, I. & Ginsburg, B. Training nGPT. Preprint at https://arxiv.org/abs/2608.01284 (2026).
56. Franke, J. K. H. et al. Learning in Compact Spaces with Approximately Normalized Transformer. Preprint at https://arxiv.org/abs/2505.22014 (2025).
57. Kosson, A., Messmer, B. & Jaggi, M. Analyzing & Reducing the Need for Learning Rate Warmup in GPT Training. Preprint at https://arxiv.org/abs/2410.23922 (2024).
58. Kalra, D. S. & Barkeshli, M. Why Warmup the Learning Rate? Underlying Mechanisms and Improvements. Preprint at https://arxiv.org/abs/2406.09405 (2024).
59. Kasimbeg, P. et al. Accelerating Neural Network Training: An Analysis of the AlgoPerf Competition. Preprint at https://arxiv.org/abs/2502.15015 (2025).
60. Kasimbeg, P. et al. Towards Easier-to-Tune Training Algorithms: New Results from AlgoPerf. Preprint at https://arxiv.org/abs/2610.13036 (2026).
61. Wen, K. et al. Understanding Warmup-Stable-Decay Learning Rates: A River Valley Loss Landscape Perspective. Preprint at https://arxiv.org/abs/2410.05192 (2024).
62. Dremov, A., Hägele, A., Kosson, A. & Jaggi, M. Training Dynamics of the Cooldown Stage in Warmup-Stable-Decay Learning Rate Scheduler. *Transactions on Machine Learning Research* (2025); https://arxiv.org/abs/2508.01483.
63. Hägele, A. et al. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations. In *Advances in Neural Information Processing Systems* **37** (2024); https://arxiv.org/abs/2405.18392.
64. Qiu, S., Xiao, L., Wilson, A. G., Pennington, J. & Agarwala, A. Scaling Collapse Reveals Universal Dynamics in Compute-Optimally Trained Neural Networks. Preprint at https://arxiv.org/abs/2507.02119 (2025).
65. Bergsma, S. et al. Scaling with Collapse: Efficient and Predictable Training of LLM Families. Preprint at https://arxiv.org/abs/2509.25087 (2025).
66. Song, M., Baek, B., Ahn, K. & Yun, C. Through the River: Understanding the Benefit of Schedule-Free Methods for Language Model Training. Preprint at https://arxiv.org/abs/2507.09846 (2025).
67. Meterez, A., Nair, P. A., Morwani, D., Pehlevan, C. & Kakade, S. Anytime Pretraining: Horizon-Free Learning-Rate Schedules with Weight Averaging. Preprint at https://arxiv.org/abs/2602.03702 (2026).
68. Ma, J. & Chen, Y. WSqD: A Horizon-Free Learning Rate Schedule for Large Model Training. Preprint at https://arxiv.org/abs/2607.10959 (2026).
69. Shang, Q. et al. SOLAR: A State-Driven Online Learning Rate Scheduler for LLM Pretraining. Preprint at https://arxiv.org/abs/2609.34681 (2026).
70. Bai, X., Liang, J.-H., Teoh, J. & Sitzmann, V. Reactive Optimizers. Preprint at https://arxiv.org/abs/2610.33826 (2026).
71. Lavie, I., Lauditi, C. & Pehlevan, C. A Solvable Model of Adaptive Learning Rate Rescaling: Acceleration, Stability & Scaling. Preprint at https://arxiv.org/abs/2610.06701 (2026).

72. Chen, Z. et al. The Role of Quantization in Loss Spikes. Preprint at https://arxiv.org/abs/2610.38277 (2026).
73. Team OLMo et al. 2 OLMo 2 Furious. Preprint at https://arxiv.org/abs/2501.00656 (2024).
74. Grattafiori, A. et al. The Llama 3 Herd of Models. Preprint at https://arxiv.org/abs/2407.21783 (2024).
75. DeepSeek-AI et al. DeepSeek-V3 Technical Report. Preprint at https://arxiv.org/abs/2412.19437 (2024).
76. Min, S. Scaling Repeated-Data Pretraining: Batch Size, Regularization, and GPU-Time Tradeoffs. Preprint at https://arxiv.org/abs/2610.15965 (2026).
77. 弘犀智能（Great Unicorn Intelligence）. Deep Hope S1 · 推训一体大模型平台（Deep Hope S1 · unified inference–training LLM platform）. https://www.hxallai.com/platform/deep-hope/ (2026).

Copyright Notice: © 2026 Great Unicorn Intelligence Technology (Shenzhen) Co., Ltd. All rights reserved. Any reposting, quotation in commentary, or adaptation must credit the authors (Great Unicorn Intelligence Research Team) and cite the source (Great Unicorn Intelligence, https://www.hxallai.com).