STRAIGHT TO ZERO: WHY LINEARLY DECAYING THE LEARNING RATE TO ZERO WORKS BEST FOR LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLMs are commonly trained with a learning rate (LR) warmup, followed by cosine decay to 10% of the maximum (10× decay). In a large-scale empirical study, we show that under an optimal max LR, a simple linear decay-to-zero (D2Z) schedule consistently outperforms other schedules when training at compute-optimal dataset sizes. Benefits increase further with more training tokens; e.g., a 617M-parameter model trained for 80 tokens-per-parameter (TPP) using D2Z achieves *lower* loss than when trained for 200 TPP using 10× decay, corresponding to an astonishing 60% FLOPs savings. This implies models like Llama2-7B, trained for 286 TPP with 10× decay, were severely under-decayed. We demonstrate the benefits of D2Z across a range of model sizes, batch sizes, and other training configurations. We explain the success of linear D2Z via a novel interpretation of AdamW as a convex combination of weight updates, with coefficients governed by the LR schedule. This interpretation demonstrates how linear D2Z balances the demands of early training (moving away quickly from initial conditions) and late training (smoothing over more updates to mitigate gradient noise).

1 INTRODUCTION

Learning rate schedules play an important role in training large language models. The original Transformers paper (Vaswani et al., 2017) proposed a brief LR warmup followed by decay proportional to the inverse square root of the step number. This schedule has the advantage of not requiring prior specification of the total training steps. However, cooling down to a specific minimum LR is acknowledged to be "preferable when one knows the training duration in advance" (Zhai et al., 2022) as it produces "slightly better results" (Raffel et al., 2020). In this paper, our primary focus is finding LR schedules that achieve the minimum loss given a fixed number of training tokens.

The "predominant choice" (Hu et al., 2024) in such training — the "de-facto standard" (Hägele et al., 2024) — is warmup followed by cosine decay to 10% of the max LR, an approach used in GPT3 (Brown et al., 2020), Gopher (Rae et al., 2022), Chinchilla (Hoffmann et al., 2022), BLOOM (Scao et al., 2023), Llama (Touvron et al., 2023a), Llama2 (Touvron et al., 2023b), Falcon (Almazrouei et al., 2023), Pythia (Biderman et al., 2023), etc. It is used "following Hoffmann et al." (Muennighoff et al., 2023), and is the default schedule in LLM codebases (Karpathy, 2024).

We present a large-scale, hypothesis-driven study to determine which schedules work best in which situations, and why. We focus on *compute-efficient* models. According to Chinchilla scaling laws (Hoffmann et al., 2022), the fewest FLOPs to achieve a given loss is obtained when models are trained for around 20 tokens-per-parameter (TPP). For inference, we often train for more than 20 TPP because smaller, over-trained models are cheaper to serve (Touvron et al., 2023a). We hypothesized that the optimal LR schedule may depend on the max LR, and validated this empirically. However, our experiments in various settings revealed a consistent outcome: when all schedules are at their optimal max LR, simple linear decay-to-zero (D2Z) works best at compute-optimal TPP. Moreover, the relative benefit of D2Z over 10× (in terms of training, validation and even downstream loss) *increases* with TPP (Figure 1). Models

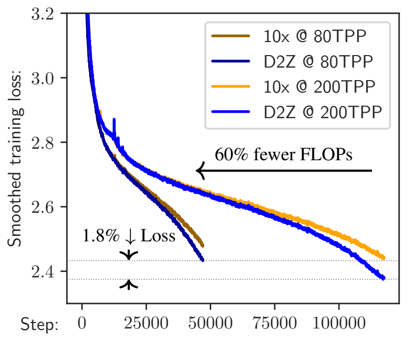


Figure 1: A 617M model trained for 80 TPP with *Linear*-D2Z has better train (and *validation*) loss than when trained for 200 TPP with *Linear*-10×.

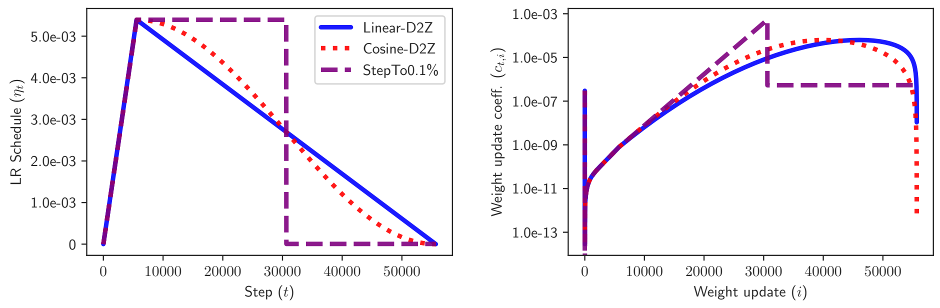


Figure 2: Each LR schedule, η_t (left) and setting of weight decay, λ, implies a convex combination of weight updates, with combination coefficients $c_{t,i}$ (right, log-scale) giving the contribution of the ith update to model weights θ_t at step t. Coefficients here for the final training step (t=55680), corresponding to settings for 111M-param μP models: $\tilde{\eta}$=1.6e-02, ρ=1/3, λ=0.1. The more sudden the drop from the high-LR period to the low-LR period, the more earlier updates are emphasized; this could explain the slight advantage of *Linear* over *Cosine* decay, and of *Cosine* over *Step* decay.

such as Llama2-7B, trained for 286 TPP at 10× decay, could likely have saved the majority of their total compute by switching to D2Z. We also confirm the advantage of linear D2Z over cosine D2Z, and various other approaches (Hu et al., 2024; Bi et al., 2024; Hägele et al., 2024).

To explain the success of *Linear*-D2Z, we build on recent work on the related topic of *weight decay* (Andriushchenko et al., 2023; Wang & Aitchison, 2024). First, we find decaying the LR *to zero* works because compute-efficient training includes a long phase in which gradient noise is the predominant factor slowing loss reduction — with more training tokens or higher noise through smaller batch sizes, the advantages of D2Z increase. Secondly, we demonstrate that approaching zero *linearly* is beneficial via a novel interpretation of AdamW (Loshchilov & Hutter, 2017) — the predominant optimizer in LLM training. With AdamW, the weights generated at each step are implicitly a weighted average over all weight updates (including the initial, random weights). The shape of the vector of combination coefficients depends on the learning rate schedule and weight decay settings (Section 3.1). Analyzing this *dual* of the LR schedule, we see that linear decay produces a favorable combination of prior weight updates (Figure 2). When LR drops abruptly, e.g., via step-decay or, to a lesser extent, cosine decay, weight updates after the drop receive less emphasis; this results in worse model quality (Section 4). The dual view also reveals the implicit schedule-awareness of recent "schedule-free" approaches (such as Warmup-Stable-Decay). However, it also suggests a method for truly schedule-free training (Section 3.3).

Our findings also expose the LR decay ratio as a powerful confounder in prior work studying optimal hyperparameter transfer across dataset and batch size, in particular with the maximal update parameterization (Yang & Hu, 2020; Yang et al., 2021); optimal max LRs are much more stable when using D2Z than when using 10× or no decay. Moreover, we explain and demonstrate that the benefits of *weight decay* are observed primarily when using LR D2Z, where raising weight decay can fine-tune the dual coefficients without affecting initial training stability.

2 Background and Related Work

2.1 Learning rate schedules

LR schedules have a long history in stochastic optimization, and are motivated by convergence bounds for stochastic gradient methods (Moulines & Bach, 2011; Bottou et al., 2018). For example, following Andriushchenko et al. (2023), consider SGD for a convex loss parameterized by θ: with a constant LR η, the gap between the optimum and current loss at step t can be bounded by:

$$\mathbb{E}[\mathcal{L}(\theta_t) - \mathcal{L}(\theta_*)] \leq (1 - \eta\mu)^t ||\theta_0 - \theta_*||^2 + \eta\sigma^2 \tag{1}$$

where θ_0 are initial parameters, θ_* is the loss minimizer, σ^2 is a bound on the variance of gradient noise, and μ is a measure of curvature of the objective around its minimum (see also derivation of Theorem 6 in Bottou et al. (2018)). A larger LR can decrease dependence on initial conditions (the *bias* term), but also increase the effect of gradient noise (the *variance* term). As training progresses, bias decreases exponentially, and the relative importance of variance increases (noise has also been found to increase in absolute terms over training (McCandlish et al., 2018)). This motivates a schedule where the LR is high initially (to mitigate bias) and lowered later (to minimize variance).

In practice, when and how to reduce the LR is rarely informed by ML theory. Many LLMs simply follow the 10x cosine schedule, which is noted to work slightly better than cosine D2Z in Hoffmann et al. (2022). It is also well established that using a portion of a longer (or extending a shorter) schedule is suboptimal compared to using a schedule that reaches its minimum only at the final training step (Hoffmann et al., 2022; Hu et al., 2024; Hägele et al., 2024). Linear decay after warmup (Howard & Ruder, 2018) has been used in LLMs, typically also to 10% of the max (Henighan et al., 2020; Dey et al., 2023; Sengupta et al., 2023). Beyond LLMs, dropping LR at specific milestones (*step decay*) is popular in vision models (He et al., 2016; Zagoruyko & Komodakis, 2016; Li et al., 2020), but has also been used in LLMs (Bi et al., 2024).

Kaplan et al. (2020) compared various decay functions and concluded the specific schedule was unimportant given a high enough average LR, although decaying to zero "gives a fixed improvement close to the end of training." Few papers explicitly compare different LR schedules for large-scale training, and when comparisons are made (Shallue et al., 2019; Kaplan et al., 2020; Schmidt et al., 2021; Hoffmann et al., 2022; Yang et al., 2021), they are not the primary focus. So, while some insights are gained, "comprehensive study" is usually regarded as "out of scope" (Aleph Alpha, 2024). One exception is Defazio et al. (2023), who found linear equals or outperforms other common schedules, including cosine, across a range of problems, including LLM training. They develop convergence bounds that theoretically motivate linear as the optimal schedule. Unlike our work, they do not evaluate LLMs with different max LRs or decay ratios.

Seeking to measure model quality at different training durations without having to re-train separate models from scratch (Zhai et al., 2022; Hägele et al., 2024), researchers have adopted various *infinite* schedules, such as constant, cyclic (Smith, 2017), etc. Following optimization theory (Moulines & Bach, 2011; Defazio et al., 2024) weight averaging provides an alternative to decay (Sandler et al., 2023; Sanyal et al., 2023; Busbridge et al., 2024), although it is typically not as effective (Hägele et al., 2024), and moreover may have hyperparameters that implicitly depend on training duration (Defazio et al., 2024). *Warmup-Stable-Decay (WSD)* approaches have also been used in LLM training (Hu et al., 2024; Shen et al., 2024a; Ibrahim et al., 2024; Hägele et al., 2024). These methods train at a constant LR, but decay from a checkpoint in a separate process when an intermediate model is needed. While the goal is to perform *as well as* fixed schedules, we show the optimal constant LR implicitly depends on training duration, making these approaches not truly *schedule-free*.

2.2 Maximal update parameterization (μP)

Conventionally, initial weights are scaled to ensure activations have unit variance (Glorot & Bengio, 2010; He et al., 2015), but such methods do not ensure stability after multiple steps of training, due to imbalances in layer-wise LRs (Yang et al., 2021). In contrast, μP (Yang & Hu, 2020) prescribes a *re-parameterization* of initial weight variances and LRs – essentially, rules for how to change these values as model width (i.e., d_{model}) changes – such that activations and updates remain on the same scale. μP also stabilizes embeddings, layer norms, and self-attention in Transformers. μP is seeing growing application in LLMs (Dey et al., 2023; Shen et al., 2024b; Hu et al., 2024), where it acts to stabilize training and to enable transfer of optimal hyperparameters (HPs) across model scales.

With μP, base HPs can be tuned on a small *proxy/base* model and then transferred to larger models. Given the width of the proxy model, d_p, and width of the target, d_t, μP prescribes scaling factors to apply to HPs. The base LR $\tilde{\eta}$ is scaled down to $\eta = \rho\tilde{\eta}$, where $\rho = {}^{d_p}/_{d_t}$. In terms of LR schedules, the base LR $\tilde{\eta}_t$ is scaled at every step to provide η_t. μP is convenient in our study as we can sweep the same *base* maximum LRs, $\tilde{\eta}$, at each model size, and observe trends that are scale-invariant.

2.3 AdamW weights as exponentially-weighted moving average (EMA)

An AdamW update at a single training step, t, can be expressed as:

$$\theta_t = (1 - \eta\lambda)\theta_{t-1} - \eta\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2}$$

where η is the learning rate, $\hat{m}_t$ and $\hat{v}_t$ are (bias-corrected) running averages of the gradient and the squared gradient, respectively, and ϵ is a small constant added to prevent division by zero. The weight decay value, λ, is typically set to 0.1 in LLM training (Brown et al., 2020; Hoffmann et al., 2022; Almazrouei et al., 2023; Aleph Alpha, 2024).

While the running averages of m and v in Adam (Kingma & Ba, 2014) and AdamW are of course exponentially-weighted moving averages (EMAs), Wang & Aitchison (2024) recently showed how the *weights* generated by AdamW can also be understood as an EMA of the weight *updates*. A generic EMA, y_t, for a time-varying quantity, x_t, can be written as:

$$y_t = (1-\alpha)y_{t-1} + \alpha x_t \tag{3}$$

where α is the *smoothing parameter*. AdamW from Eq. 2 can be interpreted as an EMA by letting:

$$y_t = \theta_t,\ \alpha = \eta\lambda, \text{ and } x_t = -\frac{1}{\lambda}\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \tag{4}$$

Wang & Aitchison note the quantity $1/\alpha$, i.e., $1/(\eta\lambda)$, provides a rough *timescale* over which updates are averaged. The weight decay λ can therefore be used to control the effective window over which weight updates are combined (smaller values of λ increase the timescale, increasing the contribution of earlier updates to the EMA). This view also motivates dynamic LR schedules. Initial timescales should be small (high η_t), to ensure early contributions to the EMA are forgotten, "while the final timescale is around the total number of epochs [low η_t], to ensure averaging over all datapoints." In fact, we will show that the contribution of weight updates, $x_0, x_1, \ldots x_t$, to the model θ_t at a particular training step, t, cannot be determined by the *instantaneous* value of $\eta_t\lambda$. Rather, the contribution of any x_i to the weights θ_t requires looking at the full LR schedule *holistically* (Section 3).

Wang & Aitchison also motivate scaling rules for the optimal λ as model and dataset size vary. If the dataset size increases by a factor of M, the EMA view recommends scaling λ by $1/M$ in order to expand the timescale proportional to the expansion of the dataset. Moreover, with μP, if model size increases and the LR is scaled by ρ (Section 2.2), the EMA view motivates scaling λ by $1/\rho$ to keep the timescale constant. Although not explored by Wang & Aitchison, analagous rules can be derived for batch size changes. Of course, in practice we typically scale the batch size and dataset size together with model size, so real application requires jointly accounting for multiple factors.

3 Methods

3.1 AdamW as convex combination of weight updates, driven by LR schedule

To properly account for time-varying LRs, we now consider a moving average with time-varying smoothing, α_t. If we let $\alpha_1 = 1$ (so that $y_1 = x_1$), we can express y_t in terms of all inputs x_t:

$$\begin{aligned} y_1 &= \alpha_1 x_1, \\ y_2 &= (1-\alpha_2)\alpha_1 x_1 + \alpha_2 x_2, \\ y_3 &= (1-\alpha_3)(1-\alpha_2)\alpha_1 x_1 + (1-\alpha_3)\alpha_2 x_2 + \alpha_3 x_3, \cdots \\ y_t &= \sum_{i=1}^{t}\left(\prod_{j=i+1}^{t}(1-\alpha_j)\right)\alpha_i x_i \end{aligned} \tag{5}$$

Let $c_{t,i} = \left(\prod_{j=i+1}^{t}(1-\alpha_j)\right)\alpha_i$ be the contribution of input x_i to output y_t at time t, such that $y_t = \sum_{i=1}^{t} c_{t,i}x_i$. It can be shown $\forall t, \sum_i c_{t,i} = 1$, and hence, as in a standard EMA, each y_t is a *convex combination* of the inputs $x_1 \ldots x_t$ (i.e., all coefficients are non-negative and sum to 1). However, with time-varying smoothing, we are not restricted to exponentially-decreasing coefficients as i decreases. Indeed, for *any* convex combination of inputs, there is a corresponding smoothing schedule that generates that combination. In terms of LR schedules for AdamW training, $\alpha_t = \eta_t\lambda$ becomes the smoothing parameter at step t (cf. Eq. 3 and 4). Note, when using a μP LR scaling factor, ρ (Section 2.2), the smoothing parameter is $\alpha = \eta\lambda = \rho\tilde{\eta}\lambda$.

This formulation enables proactive analysis of existing LR schedules — by calculating coefficients and judging how updates are integrated over training — without needing to actually train. For example, Figure 2 (Section 1) compares the effect of *Linear*, *Cosine*, and *Step* decay on the output at step t of a 111M-parameter model trained for 200 TPP (2D plots of $c_{t,i}$ over all t and i are in appendix Figure 25; see also Figures 26 and 27). Moreover, we can also design novel LR schedules to have desirable blends of weight updates (Section 3.3). Optimizing these coefficients thus provides a *dual* to optimizing the LR decay function, and we therefore refer to these coefficients as the *dual coefficients* of the LR schedule.

3.2 Bias and variance in the LR schedule dual

The dual view provides an interpretation of bias and variance that aligns with Eq. 1. The coefficient on the initial random weights, $c_{t,1} = \prod_{j=2}^{t}(1 - \alpha_j)$, monotonically decreases with t. The higher the max LR, the higher the α_j values, and the more rapidly $c_{t,1}$ diminishes. In other words, higher LRs reduce *bias* more quickly, rapidly moving the solution away from initial weights. Only if the bias is sufficiently low can we prioritize noise (*variance*) reduction by lowering the learning rate.

Note that weight updates have a coefficient of $1/\lambda$ in Eq. 4. So, while η_t and λ contribute equally to α_j, increasing λ to reduce bias is counterproductive as weight updates will be scaled down proportionally, reducing movement from initial conditions. This observation will be crucial for interpreting our experimental findings where we systematically vary LR and weight decay (e.g., Figure 6).

To what extent do the bias and variance terms play a role in modern LLM training? We hypothesize that the answer to this question is *scale-invariant* given a fixed training tokens-per-parameter (TPP). We know that, regardless of scale, models tend to reach the compute-efficient-training frontier at around 20 TPP (Hoffmann et al., 2022). It seems unlikely that at some scales, models train efficiently to 20 TPP by mostly minimizing bias, while at others they train efficiently to 20 TPP by mostly grappling with variance. It is more likely that 20 TPP is consistently efficient across scales precisely *because* it corresponds to a balanced combination of bias and variance.

3.3 Truly schedule-free LR schedules

Constant schedules, or schedules with a long constant phase such as *WSD*, are not truly "schedule-free" because their max LR setting affects the $(1 - \alpha_j)$ terms in the dual. Different LRs will correspond to different effective timescales over updates, and thus different LRs will be optimal for different training durations (see appendix Figure 26a for *Constant* and Figure 27d for *WSD*).

In contrast, we now derive a schedule such that coefficients are always weighted equally, at every training step. First, it can be shown that:

$$\frac{c_{t,i+1}}{c_{t,i}} = \frac{\alpha_{i+1}}{(1 - \alpha_{i+1})\alpha_i} \tag{6}$$

For the coefficients to be uniform at any step t, we require this ratio to be 1, which implies that smoothing evolves $\alpha_{i+1} = \frac{\alpha_i}{(1+\alpha_i)}$. Assuming a fixed weight decay (so $\alpha_i = \eta_i \lambda$), coefficients will be equal if the LR evolves:

$$\eta_{i+1} = \frac{\eta_i}{(1 + \eta_i \lambda)} \tag{7}$$

We can initialize η_0 to some value and iterate Eq. 7 to generate the full LR schedule. We call this the *Rational* schedule since it is both a rational expression of η_i and a very reasonable approach: regardless of how long we train, all weight updates contribute equally. At each step we effectively decrease all prior coefficients such that they now equal the coefficient of the current update, α_t.

With $\lambda = 1$ and no warmup, this schedule evolves like $1/n$. However, in order to distance ourselves from the initial conditions, we can warmup the LR in the usual manner to the desired max LR, then switch on rational decay, ensuring equal coefficients going forward (depicted in appendix Figure 26d). However, such a schedule will almost surely not work as effectively as *Linear* D2Z for fixed-duration training, since it lacks the cooldown phase where gradient noise is minimized. Indeed, $1/n$ has performed relatively poorly in prior studies (Ge et al., 2019; Defazio et al., 2023). However, we introduce it here as a promising schedule for *continuous* pretraining.[1]

[1]In fact, $1/n$ decay has long been regarded as optimal for strongly-convex loss (Robbins & Monro, 1951), and optimal in other contexts when combined with *averaging* (Defazio et al., 2023). Since averaging is an alternative to cooldown (Hägele et al., 2024), *Rational* plus averaging is a compelling schedule-free direction.

Table 1: Comparison of decay schedules: validation loss for 617M μP models, 20 TPP. 1.6e-02 is the μP proxy-tuned max LR. *Linear*-D2Z outperforms other schedules across most LRs.

$\tilde{\eta}$	*InvSqrt*	*Constant*	*Cosine*-10×	*Linear*-10×	*Cosine*-D2Z	*Linear*-D2Z
6.5e-02	2.789	3.035	NaN	2.667	2.611	**2.605**
3.2e-02	2.710	2.850	2.604	2.606	2.574	**2.571**
1.6e-02*	2.671	2.768	2.590	2.591	2.578	**2.573**
8.1e-03	2.665	2.722	2.598	2.600	2.595	**2.590**
4.0e-03	2.691	2.711	2.634	2.635	2.637	**2.633**
2.0e-03	2.762	2.739	**2.707**	2.710	2.717	2.714

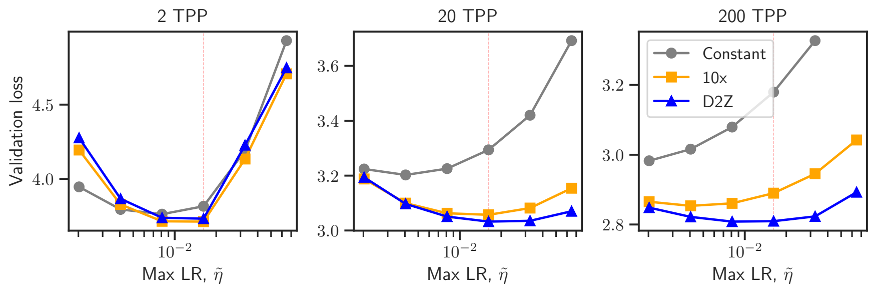


Figure 3: Validation loss for 111M models. As TPP increases, *Linear*-D2Z outperforms 10×, especially at the proxy-tuned max LR (red lines). 617M results are similar (appendix Figure 12).

4 Empirical Analysis of Decay-to-Zero

4.1 Experimental setup

Experiments use a GPT-like LLM (Radford et al., 2019), with ALiBi embeddings (Press et al., 2022) and SwiGLU (Shazeer, 2020). Models are trained on SlimPajama (Soboleva et al., 2023) and evaluated over 1.1B tokens (regardless of training TPP). Unless otherwise indicated, standard weight decay of $\lambda = 0.1$ is used. Training runs use the same random seed, so all decay functions (*Linear*, *Cosine*, etc.) and ratios (*Constant*, 10×, D2Z) have identical warmup phases, but note validation is very consistent across seeds at this scale (appendix Figure 22). By default we use μP (standard parameterization results are in appendix Section B.2). μP hyperparameters are derived from a smaller proxy model tuned using a *Linear*-10× schedule. Since we hypothesize the max LR, $\tilde{\eta}$, is a confounder when comparing LR schedules, we sweep $\tilde{\eta}$ by factors of 2× around the μP proxy-tuned $\tilde{\eta} = 1.6\text{e-}02$. Appendix A has full experimental details.

4.2 Results

***Finding 1**: Linear-D2Z is the optimal LR schedule across virtually all maximum learning rates.*

For 617M-parameter μP models trained to compute-optimal 20 TPP, the optimal *Linear*-D2Z setting achieves 0.77% lower loss than the optimal *Linear*-10× setting (Table 1). At smaller, suboptimal max LRs, 10× can be better than D2Z. In appendix Figure 10, we demonstrate very similar results using the standard parameterization. Regarding the decay function, gains from *Linear*-D2Z over *Cosine*-D2Z are small, but perfectly consistent across max LRs, exactly in line with recent work (Defazio et al., 2023; Lingle, 2024). Interestingly, *Cosine*-10× is consistently slightly better than *Linear*-10×, showing that *Linear* itself is not always best, rather *Linear plus* D2Z is needed. Lacking a cooldown phase, inverse square root (*InvSqrt*) and *Constant* do not perform as well.

For the remaining experiments, we use a *Linear* schedule unless otherwise indicated.

***Finding 2**: As TPP increases, the relative improvement of D2Z over 10× also increases.*

For 111M models at only 2 TPP, D2Z performs *worse* than 10× across all max LRs (Figure 3; 617M plot is similar, see appendix Figure 12). However, as TPP increase, D2Z begins to outperform 10×,

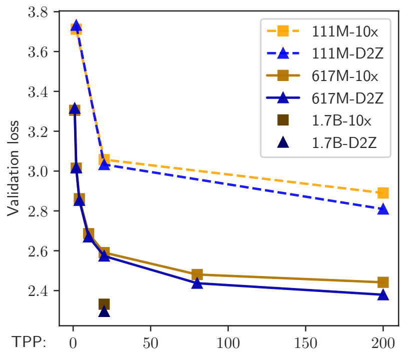


Figure 4: As TPP increases, the validation loss gap between *Linear*-D2Z vs. *Linear*-10× grows, across model sizes.

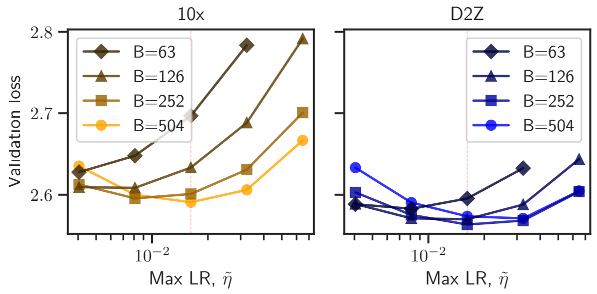
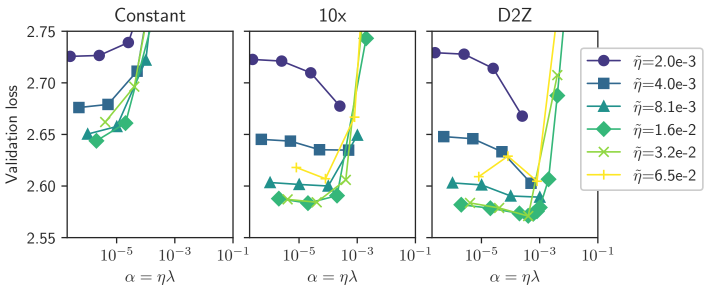


Figure 5: Validation loss for 617M models as batch size varies, comparing 10× (left) to D2Z (right). All models trained to 20 TPP (smaller batches have more steps). As batch size (B) decreases, the optimal LR ($\tilde{\eta}$) drifts significantly lower for 10× decay, less so for D2Z.



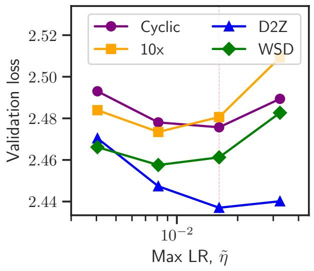


Figure 6: Validation loss for 617M models (20 TPP) as α=$\eta\lambda$ varies. Optimal loss corresponds to high (but not too high) $\tilde{\eta}$ (indicated by color). Only D2Z models further improve as λ increases.

Figure 7: Validation loss for 617M models (80 TPP): D2Z surpasses *Cyclic* and *WSD* schedules.

exceeding the best setting of 10× by 1.6% at 200 TPP, and performing 2.8% better at the proxy-tuned $\tilde{\eta}$ = 1.6e-02 (marked in plots with a red vertical line). Figure 4 plots the validation loss of 10× and D2Z at different TPP settings for different model sizes; in this plot, all models are trained using the proxy-tuned max LR. For the 617M model, D2Z is also initially worse than 10×, but begins to surpass it around 4 TPP, and by 200 TPP is 2.6% better. As with training loss (Figure 1), an 80 TPP D2Z model can surpass a 200 TPP 10× model in validation loss.

Importantly, these trends also hold for *downstream* evaluation of the models (appendix Table 5).

***Finding 3**: Compared to* Constant *and 10×, optimal max LRs are much more stable with D2Z.*

In Figure 3 we see different levels of hyperparameter sensitivity when increasing TPP: the optimal LR shifts substantially lower for *Constant*, somewhat lower for 10×, and hardly at all for D2Z. The same trend holds at 617M scale (appendix Figure 12). Moreover, with D2Z, loss is less sensitive to a sub-optimal LR (bowls are flatter). Similar trends can be observed when we vary the batch size (Figure 5). Note D2Z is superior to 10× across all batch sizes (this is observed more clearly in appendix Figure 13). In terms of LR stability, 10× models already see significant shift at a batch size B=126; D2Z models begin to shift at B=63. For models with varying batch sizes, but where we fix the number of *steps*, rather than TPP, results are broadly similar (appendix Figure 14). D2Z is also superior, and loss is more stable, as we vary *weight decay* (appendix Figure 15 and 17).

***Finding 4**: Benefits of* weight decay *are observed primarily when using LR D2Z.*

We explore the interaction between *weight decay* (WD) and LR in Figure 6 (for 617M models) and appendix Figure 16 (for 111M). Here we plot with the x-axis set to the (max) smoothing parameter, $\alpha = \eta\lambda$; note for a given decay ratio and α, dual coefficients (Section 3) are identical. Thus it supports the EMA/dual view to note that regardless of max LR, $\tilde{\eta}$, models tend to reach their lowest loss at around the same α. However, to reach optimal loss, models require $\tilde{\eta}$ to be high, but not too high (at $\tilde{\eta} = 6.5\text{e-}02$ and above, we consistently encounter instabilities in training). When increasing λ with a given maximum LR (moving left-to-right along curves), note the behavior differs depending on LR schedule: *Constant* models perform worse, *Linear*-10× models see marginal gains, and *Linear*-D2Z models benefit significantly. Results are similar for 111M models (appendix Figure 16).

These results align with Andriushchenko et al. (2023) who also observed no benefit from WD when using constant LRs. We also confirm WD does not act like a traditional regularizer here; beneficial settings of WD always improve both validation *and* training loss. Aleph Alpha (2024) also recently showed WD=0.1 improves over WD=0.0 in LLM training; notably, they also train with D2Z.

***Finding 5**: Linear-D2Z works better than WSD and Cyclic (continuous) LR schedules.*

In Figure 7, we compare D2Z to 10×, and to two approaches designed for continuous pretraining: *Cyclic*, which cycles LR up and down, and *WSD* (Section 2.1). For *WSD*, we simulate a model being retrieved after 80 TPP, and cool the LR for the final 22.5% of steps (around the proportion recommended by Hägele et al. (2024), and equal to the cooldown duration in our 20 TPP models). Appendix Figure 27 provides full LR curves and dual coefficients for all models in Figure 7.

At its optimal LR, *WSD* works better than 10×, confirming results in Hägele et al. (2024). However, note the optimal max LR shifts lower for both 10× and *WSD* at this TPP. *Linear*-D2Z remains best here, around 0.84% better than the optimal *WSD*. Given the diminishing returns of high-TPP training (Figure 4), *WSD* would require significantly more training FLOPs to reach the level of D2Z.

***Finding 6**: Constant and 10×, but not D2Z, strongly overfit to the end of the training data.*

We also performed an experiment where we evaluated trained models on the same data, in the same order, as used during training (Figure 8). As predicted by the dual view (Section 3), the higher the LR, the more the models fit to later training data. It is striking both *Constant* and 10×, but not D2Z, *overfit* to the very final portion. Extra adaptation of generative models to recent training sequences has long been observed (Graves, 2013), but to our knowledge this is the first evidence that D2Z may help mitigate these effects. Since D2Z performs best on data *slightly before* the final training phase, placing the highest-quality and most-recent data in the *very* final phase, while using D2Z (e.g., as in Dubey et al. (2024)), may be suboptimal.

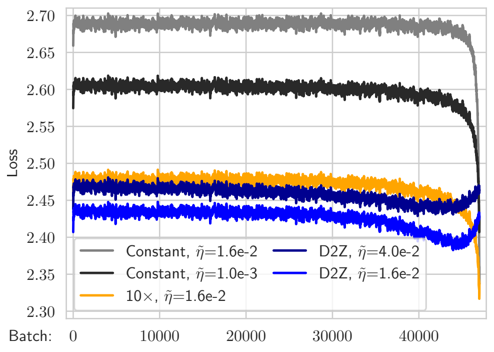


Figure 8: Loss on *training set* after models have been trained: 617M models, 80 TPP.

5 Discussion

5.1 Interpretation of our results

LR decay mitigates gradient variance, which grows with TPP Recall Eq. 1: learning can benefit from a high LR early (to escape initial conditions: *bias*), and a low LR later (to minimize increasing gradient noise: *variance*). Of course, there is no hard transition between these phases. Moreover, as TPP increases, the balance between these two criteria shifts (Figure 9). With *Constant*, the shifting emphasis as TPP increases can only be satisfied by lowering the fixed LR (explaining the shift in optimal LR, Figures 3 and 12). At 2 TPP, some LR decay is beneficial, but too much hampers bias reduction; here, D2Z underperforms 10×. However, a major finding of our work is that, when training LLMs at *compute-efficient* TPP (20+ TPP), decaying to *zero* is optimal. At these TPP, optimal LRs for 10× also shift lower as noise reduction gains importance. But lower LRs move parameters less from initial conditions, and 10× loss lags D2Z.

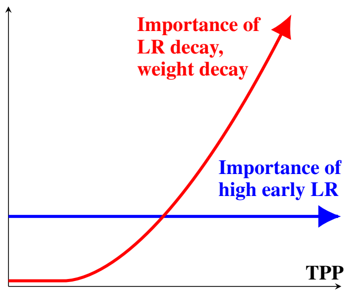

Importance of LR decay, weight decay
Importance of high early LR
TPP

Figure 9: LR and weight decay settings increase in importance with training duration.

Our results clarify that reducing $\alpha = \eta\lambda$ is primarily beneficial as a noise-reduction mechanism. Recall that in the view of Wang & Aitchison (2024), the benefit of α is to optimize the timescale of *data*; if we double the amount of data, we should double the steps that we smooth over, e.g., *decrease* η or λ by a factor of 2. But consider appendix Figure 14: as we double the amount of data (via doubling of the batch size and keeping the same number of steps), the optimal LR actually *increases* rather than decreases. This is because increasing batch size reduces gradient noise; with less noise, we can afford a larger LR throughout training. Indeed, since raising α decreases the number of AdamW weight updates that we combine, it has a similar effect to decreasing the batch size; optimal batch sizes of course also deeply depend on gradient noise (McCandlish et al., 2018).

η plays a role early, $\eta\lambda$ plays a role later Section 3.2 makes the observation that while LR η and weight decay λ can both equally change the dual coefficients, only η is effective in moving the model away from initial conditions. This is confirmed by Figure 6 and 16: only when η is sufficiently high do models achieve optimal loss. Given such an η, the benefit of weight decay is evidently to adjust $\alpha = \eta\lambda$ to a setting that synergizes best with the decay schedule (for 617M models, around 4e-04). Weight decay is beneficial because training instabilities prevent reaching this optimal α purely through increasing η. We can further confirm that the α setting primarily plays a role *later* in training as follows: First, note again that *Constant* does not benefit from weight decay. *Constant* must make a training trade-off: it uses a LR that is sub-optimally *low* early and sub-optimally *high* later on. Increasing λ raises the already-too-high (late) α even higher, hurting the model. In contrast, if α primarily played its role early (e.g., via wider exploration of the loss surface), increasing λ would improve *Constant* loss, but this is not the case. In this way, $\alpha = \eta\lambda$ can be viewed as the *effective* or *instrinic* LR (Li et al., 2020; Wang & Aitchison, 2024), but only later in training.

The special benefit of D2Z A low α later in training can expand the timescale over which we combine weight updates, reducing noise in a manner similar to increasing batch size. However, there is apparently a separate, independent benefit from decaying LR to a very small value. Indeed, looking at appendix Figure 27 for the 80 TPP comparison to the continuous schedules, we see 10× coefficients are quite similar to the D2Z curve, apart from missing the final drop. Moreover, they are flatter than the *WSD* dual for the same max LR, suggesting better integration of prior updates. Yet *WSD* performs better than 10× at all LR settings (Figure 7). In contrast, at 2 TPP (Figure 3), 10× performs better than D2Z at every LR setting. Prior work has shown large LRs allow exploration of the loss surface at a height “above the valley floor” (Xing et al., 2018), while LR cooldown phases descend into a local minimum (Hu et al., 2024; Hägele et al., 2024). It appears that descending into these minima is beneficial only after sufficient exploration of the loss surface.

5.2 Interpretation of prior results

The confounding role of training duration LR schedules have been an unappreciated confounder of studies varying batch size and TPP. Analagously, TPP has been a confounder in studies evaluating LR schedules. Recall Kaplan et al. (2020) saw a benefit from D2Z. In contrast to Chinchilla scaling laws (Hoffmann et al., 2022), in the Kaplan et al. perspective, small models should be trained to very high TPP, while larger models should be trained less. It is therefore not surprising Kaplan et al. saw benefits testing D2Z with small models; as we have shown, D2Z is especially effective at high TPP. Similarly, in Figure 4 of Yang et al. (2021), *Linear* is worst of all schedules, and the gap between it and *Constant* and *InvSqrt* grows with model width. But here, since training data is fixed, TPP *decreases* as width increases. Thus training is in a phase where bias reduction is paramount. In contrast, in their LLM training experiments, Yang et al. do report linear D2Z to work best.

With this context, we can further speculate on why D2Z is not more widely used. First, it is common to evaluate hyperparameters on smaller training runs; unfortunately, with limited training, D2Z misleadingly underperforms. Secondly, coupling between LR schedule and the optimal max LR is problematic. That is, with 10× decay, we may find a lower max LR is optimal; if we then test D2Z with the same max LR, we may not see a benefit. Finally, poorly controlled training dynamics may prevent networks from being trained with LRs high enough to achieve optimal quality. Indeed, when we initially compared D2Z and 10× with NanoGPT (appendix Section B.9), 10× performed better at the default LR. Raising the LR resulted in training divergence. Only after switching from `float16` to `float32` could D2Z succeed — and the model reach optimal loss.

The confounding role of LR schedule Sensitivity of the optimal max LR to the LR schedule can explain a number of findings in prior work. For example, Shen et al. (2024b) were puzzled by their observation that "although the *WSD* scheduler could, in theory, continue in the stable phase forever . . . the optimal LRs are different for different amounts of training tokens." As noted in Section 3.3, LR schedules with long periods of constant LR only appear "schedule-free" in the primal; from the dual perspective, the higher the LR, the more emphasis is placed on recent updates (see appendix Figure 26a, for *Constant*, and Figure 27d, for *WSD*). Figure 8 provides empirical evidence for this perspective; the highest max LRs achieve lowest loss on the final (re-visited) training batches. Yang et al. (2021, Figure 19) also observed significant decreases in optimal max LR when TPP increases; this can also be explained by noting these tests were done with a constant LR.

Recent work has also explored the extent to which optimal HPs under μP transfer as we vary batch size. Some prior work (Yang et al., 2021; Noci et al., 2024; Shen et al., 2024b) has observed linear scaling of optimal LR with batch size, i.e., the so-called *linear scaling rule* (Krizhevsky, 2014; Chen et al., 2016; Smith et al., 2018). Others (Lingle, 2024) have observed square-root scaling, resonating with other prior studies (Hoffer et al., 2017; You et al., 2019; Malladi et al., 2022). This discrepancy can be explained by noting the linear scaling results were all found with a *Constant* or *WSD* LR decay, while square-root was observed with *Linear* D2Z, again underscoring the greater stability of D2Z. Other proposed scaling rules, e.g., square-root scaling for weight decay (Loshchilov & Hutter, 2017), should also be re-evaluated to account for the LR schedule.

5.3 Limitations and Further Experiments

While our findings provide strong evidence for linear D2Z being optimal in our specific context, there are several limitations to keep in mind. First, our focus in this paper was specifically LLM training at compute-optimal dataset sizes. For ML problems with limited access to training data, D2Z is likely not the best strategy. Second, our work focuses on AdamW (the preferred optimizer for LLM training). While the dual view of LR schedules will likely apply to other optimizers that use decoupled weight decay (as similarly noted by Wang & Aitchison (2024)), it may not apply to approximate second order methods, such as Shampoo (Gupta et al., 2018). Finally, for LLMs with unstable training dynamics that cannot tolerate high LRs, D2Z may not be beneficial. We experienced this first-hand when we initially trained NanoGPT (appendix Section B.9).

With these caveats in mind, we also highlight here the remarkable consistency of D2Z's success. Beyond the experiments in the main paper, we refer the reader to the appendices for further results with downstream evaluations (Section B.1), as well results with different parameterizations (Sections B.2, B.9), model scales (Section B.3, B.10), training durations (Section B.4), batch sizes (Section B.5), weight decay settings (Section B.6), datasets (Section B.9, B.10), model architectures (Section B.10), weight sparsity settings (Section B.7), and training frameworks (Section B.9).

6 Conclusion

The main takeaway from our work is that linear decay-to-zero is the optimal decay strategy for LLM training using AdamW. To be clear, less decay is beneficial at low tokens-per-parameter training, but there is no practical reason to perform such training with LLMs, since the same FLOPs could be used to train a smaller model, over more tokens, to a lower loss – using D2Z. The superiority of D2Z in this compute-optimal context was validated across a range of experimental conditions. Results suggest its relative benefit will increase as models increase in scale. Moreover, when using D2Z and μP, the optimal max LR is less sensitive to changes in weight decay, dataset size, and batch size, i.e., there is better hyperparameter *transfer*.

Varying the decay schedule has proven to be a useful tool for stress-testing LLMs and developing insights into training. Here, our analysis was aided by our interpretation of AdamW's output as a convex combination of prior weight updates. D2Z overfits less the final training sequences, and is especially beneficial when gradient noise dominates training. As we enter a phase of applied ML where inference efficiency becomes the primary concern, there is strong motivation to study high-TPP training, where gradient noise is the bottleneck. While our results indicate that D2Z is a key component of the solution here, further investigation is required, including into how and when to adjust hyperparameters such as weight decay, batch size, and learning rate, in the high-TPP context.

REFERENCES

Aleph Alpha. Introducing Pharia-1-LLM: transparent and compliant. https://aleph-alpha.com/introducing-pharia-1-llm-transparent-and-compliant/, 2024. Accessed: 2024-09-07.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The Falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

Maksym Andriushchenko, Francesco D'Angelo, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? *arXiv preprint arXiv:2310.04415*, 2023.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. DeepSeek LLM: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling, 2023.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Dan Busbridge, Jason Ramapuram, Pierre Ablin, Tatiana Likhomanenko, Eeshan Gunesh Dhekane, Xavier Suau Cuadros, and Russell Webb. How to scale your EMA. *Advances in Neural Information Processing Systems*, 36, 2024.

Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

Aaron Defazio, Ashok Cutkosky, Harsh Mehta, and Konstantin Mishchenko. When, why and how much? adaptive learning rate scheduling by refinement. *arXiv preprint arXiv:2310.07831*, 2023.

Aaron Defazio, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, Ashok Cutkosky, et al. The road less scheduled. *arXiv preprint arXiv:2405.15682*, 2024.

Nolan Dey, Gurpreet Gosal, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. Cerebras-GPT: Open compute-optimal language models trained on the Cerebras wafer-scale cluster. *arXiv preprint arXiv:2304.03208*, 2023.

Nolan Dey, Shane Bergsma, and Joel Hestness. Sparse maximal update parameterization: A holistic approach to sparse training dynamics. *arXiv preprint arXiv:2405.15743*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling, 2020.

Rong Ge, Sham M Kakade, Rahul Kidambi, and Praneeth Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. *Advances in neural information processing systems*, 32, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (PMLR)*, 2010.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, 2018.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

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. *arXiv preprint arXiv:2405.18392*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35, 2022.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 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. *arXiv preprint arXiv:2403.08763*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, 2020.

Andrej Karpathy. nanoGPT, 2024. URL https://github.com/karpathy/nanoGPT. GitHub repository.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Analyzing & eliminating learning rate warmup in GPT pre-training. In *High-dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*, 2014.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Zhiyuan Li, Kaifeng Lyu, and Sanjeev Arora. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. *Advances in Neural Information Processing Systems*, 33:14544–14555, 2020.

Lucas Lingle. A large-scale exploration of μ-transfer. *arXiv preprint arXiv:2404.05728*, 2024.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2017.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the SDEs and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35: 7697–7711, 2022.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An Empirical Model of Large-Batch Training, 2018.

Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in Neural Information Processing Systems*, 24, 2011.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36, 2023.

Lorenzo Noci, Alexandru Meterez, Thomas Hofmann, and Antonio Orvieto. Why do Learning Rates Transfer? Reconciling Optimization and Scaling Limits for Deep Learning. *arXiv preprint arXiv:2402.17457*, 2024.

Ofir Press, Noah Smith, and Mike Lewis. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *International Conference on Learning Representations*, 2022.

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

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling Language Models: Methods, Analysis & Insights from Training Gopher, 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 2020.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

Mark Sandler, Andrey Zhmoginov, Max Vladymyrov, and Nolan Miller. Training trajectories, mini-batch losses and the curious role of the learning rate. *arXiv preprint arXiv:2301.02312*, 2023.

Sunny Sanyal, Atula Neerkaje, Jean Kaddour, Abhishek Kumar, and Sujay Sanghavi. Early weight averaging meets high learning rates for LLM pre-training. *arXiv preprint arXiv:2306.03241*, 2023.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model, 2023.

Robin M Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley-benchmarking deep learning optimizers. In *International Conference on Machine Learning*, pp. 9367–9376. PMLR, 2021.

Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, Haonan Li, Fajri Koto, William Marshall, Gurpreet Gosal, Cynthia Liu, Zhiming Chen, et al. Jais and Jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models. *arXiv preprint arXiv:2308.16149*, 2023.

Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019.

Noam Shazeer. GLU Variants Improve Transformer, 2020.

Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. JetMoE: Reaching Llama2 performance with 0.1M dollars. *arXiv preprint arXiv:2404.07413*, 2024a.

Yikang Shen, Matthew Stallone, Mayank Mishra, Gaoyuan Zhang, Shawn Tan, Aditya Prasad, Adriana Meza Soria, David D Cox, and Rameswar Panda. Power scheduler: A batch size and token number agnostic learning rate scheduler. *arXiv preprint arXiv:2408.13359*, 2024b.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE, 2017.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, 2017.

Xi Wang and Laurence Aitchison. How to set AdamW's weight decay as you scale model and dataset size. *arXiv preprint arXiv:2405.13698*, 2024.

Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. Small-scale proxies for large-scale transformer training instabilities. *arXiv preprint arXiv:2309.14322*, 2023.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with SGD. *arXiv preprint arXiv:1802.08770*, 2018.

Table 2: Model architecture and batch sizes for main experiments

Model	d_{model}	n_{layers}	d_{head}	batch size
111M	768	10	64	192
617M	2048	10	64	504
1.7B	2048	32	64	504

Table 3: Training steps for main experiments

Model	TPP	Warmup	Steps	Tokens
111M	2	56	557	219M
111M	20	556	5568	2.19B
111M	200	5560	55680	21.9B
617M	2	118	1176	1.21B
617M	20	1175	11752	12.1B
617M	200	11750	117520	121B
1.7B	20	3322	33220	34.3B

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

Greg Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer. In *Advances in Neural Information Processing Systems*, 2021.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022.

A Experimental Details

Table 2 provides details on model architecture and hyperparameters for the main experiments (i.e., results presented in the main paper). Table 3 provides information on the training steps. All the models in our main experiments were trained on the SlimPajama dataset (Soboleva et al., 2023), a cleaned and deduplicated version of the RedPajama dataset. We use the GPT-2 (Radford et al., 2019) vocabulary of size 50257, and a context length of 2048 tokens. Following standard practice, we do not apply weight decay or bias to LayerNorm layers. Validation loss is always computed over 1.1B tokens, regardless of training TPP. By default we parameterize with μP.

For a given TPP, all models have the exact same warmup phase: a linear warmup of the LR from 0 to the maximum value. In all our runs, warmup was 10% of the total steps. LR warmup is standard practice in LLM training.[2]

[2]While prior work has suggested LR warmup is less valuable in modern Pre-LN Transformers (Xiong et al., 2020), various other studies have shown warmup leads to lower loss (Goyal et al., 2018; Liu et al., 2019; Tissue et al., 2024; Kosson et al., 2014), and may reduce sensitivity to the maximum LR (Wortsman et al., 2023). In

Table 4: Tuned hyperparameters for μP proxy model

$\sigma_{W,\text{base}}$	8.67e-02
$\tilde{\eta}$	1.6e-02
α_{input}	9.17
α_{output}	1.095

Table 5: Downstream evaluations for 617M-parameter models corresponding to Figure 1. A model trained for 80 tokens-per-parameter with linear D2Z has equivalent downstream loss to the same model trained for 200 TPP with 10$\times$ decay.

	MMLU (Avg.)	**Commonsense Reasoning**								**Reading Comp.**	**Truthfulness & Bias**		**Down-stream (Avg.)**
		Wino-grande	**Hella-swag**	**Open-Book QA**	**Lamb-ada OpenAI**	**Lamb-ada Stand.**	**SIQA**	**PIQA**	**Arc-e**	**RACE**	**Truth-ful QA**	**CrowS-Pairs**	
10$\times$@80TPP	23.6%	52.2%	43.9%	31.4%	46.7%	36.7%	32.8%	68.7%	47.6%	32.3%	39.8%	60.7%	43.05%
D2Z@80TPP	23.5%	53.4%	44.6%	31.6%	46.9%	37.3%	33.2%	68.8%	48.8%	33.4%	40.2%	60.8%	43.54%
10$\times$@200TPP	23.3%	53.4%	46.6%	31.2%	46.2%	38.8%	32.2%	68.8%	47.9%	34.4%	38.4%	60.4%	43.46%
D2Z@200TPP	24.7%	54.5%	48.2%	32.4%	50.0%	42.6%	32.9%	70.1%	50.4%	32.6%	38.9%	62.5%	45.00%

All models in the main experiments were trained on a Cerebras CS-3 system. 617M-parameter models take roughly 6 hours each to train on a single CS-3. If a training run did not complete due to numerical instabilities, the values are left off our plots or marked as NaN in our tables.

Proxy model hyperparameter tuning To find the optimal μP hyperparameters (HPs), we trained a 39M-parameter proxy model using a width $d_{\text{model}} = d_p$ of 256, with 24 layers and a head size of 64. We trained this proxy model on 800M tokens with a batch size of 256 and context length 2048, using 10$\times$ decay. We randomly sampled 350 configurations of base learning rates, base initialization standard deviation, and embedding and output logits scaling factors, and used the top-performing values as our tuned HPs (Table 4).

B Additional experimental results

In this section, we include some additional results to support the findings in the main paper. All validation losses reported in this section are from models trained with *Linear* decay.

B.1 Downstream evals

Table 5 presents a variety of downstream evaluations of the four models presented in Figure 1. Differences between the models here are largely consistent with the differences in training and validation loss, showing that D2Z is meaningful not just for the autoregressive training objective, but for real-world applications.

B.2 Standard parameterization

Figure 10 presents results for a 617M-parameter model trained with the standard parameterization. Here $\tilde{\eta}$ is therefore not a μP-corrected base LR, but rather a LR that we swept directly for this model scale. Results are obviously quite similar to results using μP, suggesting the benefits of D2Z are not μP-specific. Further results using the standard parameterization, but for NanoGPT models, are in Section B.9 below.

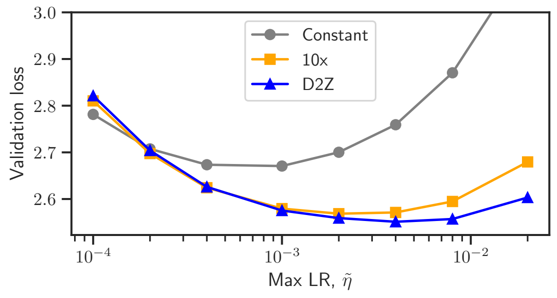


Figure 10: Validation loss for different LR and decay combinations, for a 617M-parameter model trained with the *standard parameterization*.

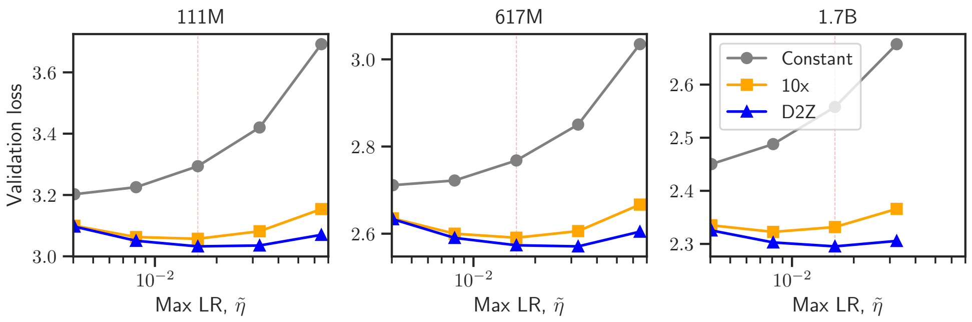


Figure 11: Validation loss at 20 TPP for different *model sizes*. Across all model sizes, *Linear*-D2Z outperforms *Linear*-10×. Note: Missing high-LR values in all plots correspond to failed training runs due to `NaN` instabilities).

B.3 Model sizes

Figure 11 presents results across 111M, 617M, and 1.7B model sizes, all trained to 20 TPP. Note the absence of results for the highest LR setting at the 1.7B-scale; at the very highest LR, numerical instabilities led to failed training runs. Otherwise, results are fairly similar across model sizes. At the proxy-tuned max LR, the gap between D2Z and 10× is 0.81%, 0.67%, and 1.56%, at the 111M, 617M, and 1.7B scales, respectively. We further investigate the issue of whether the gap between D2Z and 10× varies with model size as part of our scaling law experiments below (Section B.10).

B.4 TPP

As we vary TPP, we consistenly see increasing gains with D2Z. Here we plot the results for the 617M-scale models in Figure 12, as a counterpart to main Figure 3.

B.5 Batch sizes

Additional batch size experiments are plotted in Figure 13 and Figure 14. In fact, Figure 13 is the same data as in Figure 5, but with each batch separated into a separate subplot in order to better see how the differences between D2Z and 10× evolve as batch size changes. Both of these plots train for the same number of total tokens (20 TPP). In contrast, in Figure 14, we keep the number of *steps* constant (11752), so each model will see the same total number of batches; the batches will just be of varying size. Note, for the purposes of scale, the results at B=504 are the same in Figure 13 and Figure 14; the latter just has a larger range on the y-axis.

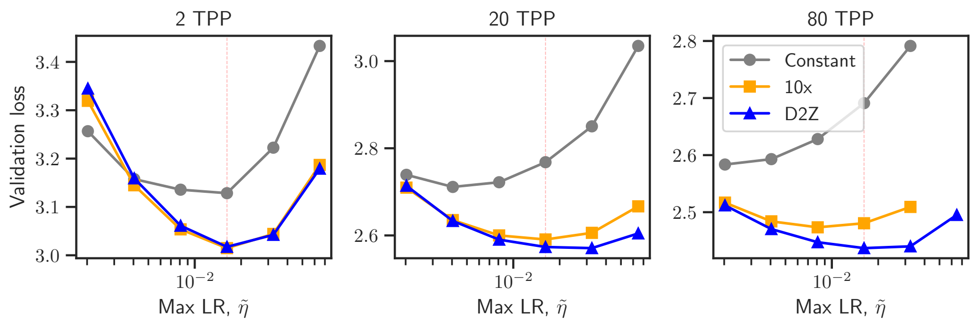


Figure 12: Validation loss for 617M models at different *TPP*. As TPP increases, *Linear*-D2Z begins to outperform *Linear*-10×, especially at the proxy-tuned max LR (red lines). The optimal LR also shifts significantly lower for *Constant*, somewhat lower for 10×, and hardly at all for D2Z. Compare to Figure 3 for 111M models.

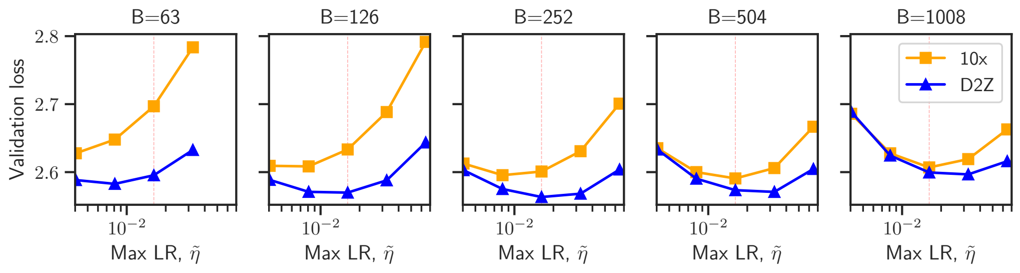


Figure 13: Validation loss for 617M models trained for different *batch sizes* but *all at 20 TPP*. As batch size decreases, the relative gain of D2Z over 10× increases. Same data as in Figure 5.

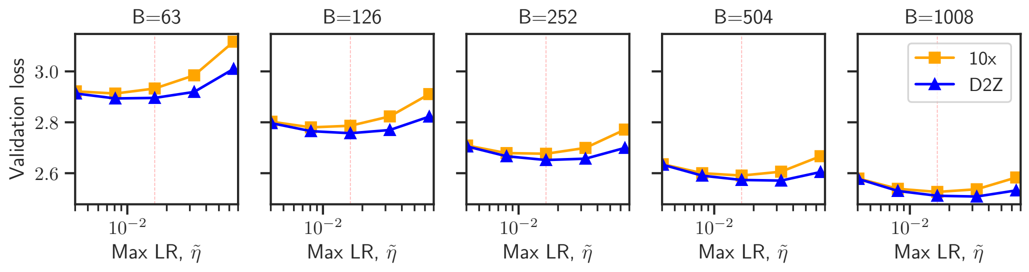


Figure 14: Validation loss for 617M models trained for different *batch sizes*, but all trained for *11752 steps* (not iso-FLOP, smaller batches see fewer TPP). Companion to Figure 5 and appendix Figure 13 which instead keep TPP constant. D2Z remains superior in the iso-Step context.

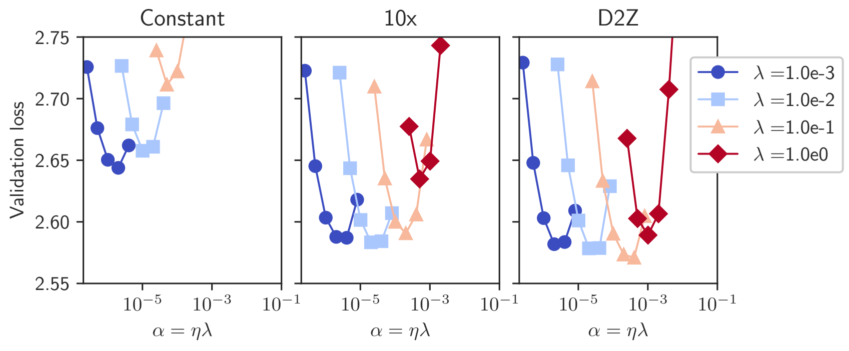


Figure 15: Validation loss for 617M models (20 TPP) as α=$\eta\lambda$ varies. Subset of the data in Figure 6, but now curves trace points with same weight decay λ (in color) and LR η varies across each curve. Only D2Z models significantly improve as we increase weight decay. D2Z models are also less sensitive to choice of λ.

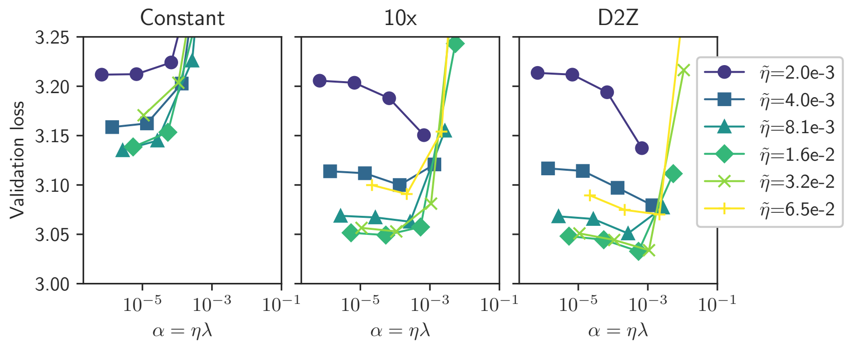


Figure 16: Validation loss for 111M-parameter models trained to 20 TPP, for different settings of decay, learning rate, $\tilde{\eta}$ (marked by color), and weight decay, λ (corresponding to points on the LR curves). The optimal loss is obtained when $\tilde{\eta}$ is high, but not too high, typically around 1.6e-02 to 3.2e-02. As the smoothing α increases, *Constant* models suffer, 10× models see marginal gains, while D2Z models benefit significantly. See Figure 6 for 617M-model results.

B.6 Weight Decay

We also provide additional weight decay results. Figure 15 is the same data as in main paper Figure 6, except we now group the points by weight decay λ rather than max LR $\tilde{\eta}$ (Figure 6 also includes some additional λ settings specifically for $\tilde{\eta}$ = 1.6e-02, and we leave those off Figure 15 to reduce clutter).

Figure 16 and 17 provide the 111M counterparts to the 617M-scale weight decay plots. Results are broadly similar.

Note that while *Constant* and 10× are much worse at higher λ values, D2Z performs reasonably well even at $\lambda = 1.0$ (particularly at the 617M scale). Recall that increasing λ effectively decreases the timescale over which weight updates are integrated (Section 3). Since D2Z has lower η later in training, it somewhat counterbalances the increase in λ. Another view of this is that as fewer updates are combined, noise increases; D2Z is evidently better at mitigating such noise.

Step decay In Figure 18, we present results investigating the impact of *Step* decay on model training. Here, for 111M-parameter models, *Step* decay can improve the loss versus keeping the LR constant, but the resulting losses are still much worse than those obtained with D2Z or 10× decay.

light of the similar benefits of D2Z, it would be interesting to investigate the value of warmup for models that are specifically trained using D2Z.

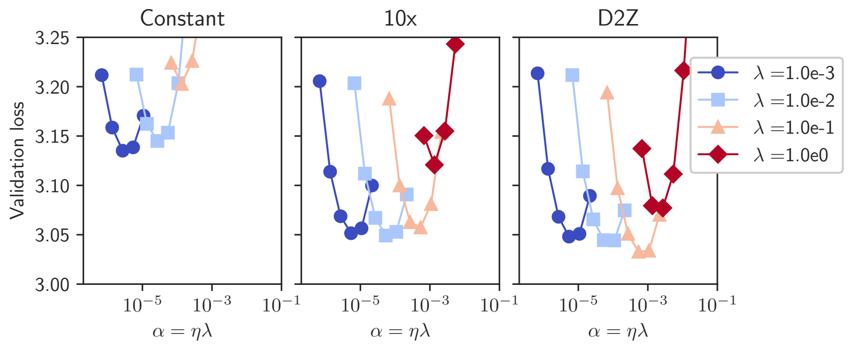


Figure 17: Validation loss for 111M models (20 TPP) as α=$\eta\lambda$ varies. Same data as in Figure 16, but now curves trace points with same weight decay λ (in color) and LR η varies across each curve. Only D2Z models significantly improve as we increase weight decay. D2Z models are also less sensitive to choice of λ. See Figure 15 for 617M-model results.

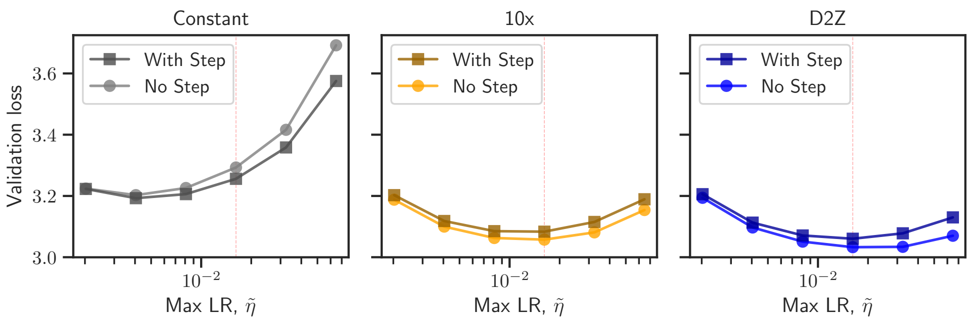


Figure 18: Validation loss for different LR and decay combinations, for a 111M-parameter model, *with and without* Step *decay*. *Step* decay is applied after 90% of training, stepping to a value equal to 0.1% of the maximum LR. Dropping the LR in this manner helps *Constant*, although it is still below the level of 10× and D2Z. Adding stepping to a 10× or D2Z schedule is not beneficial.

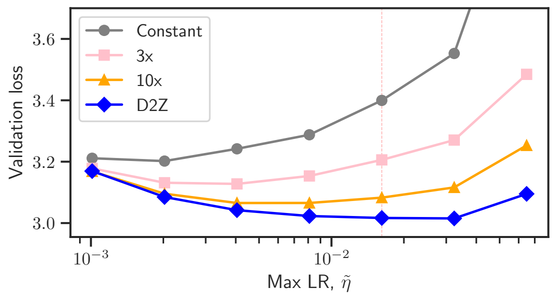


Figure 19: Validation loss for different LR and decay combinations, for a 617M-parameter model trained with SμPar, with 93.75% unstructured weight sparsity. As the decay rate increases, loss improves, while the optimal max LR is much more stable.

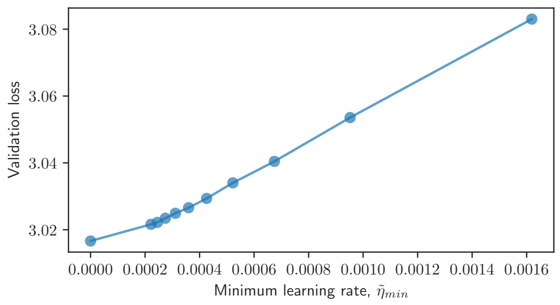


Figure 20: Validation loss for 617M-parameter models trained with SμPar, with 93.75% unstructured weight sparsity, as we vary the LR decay ratio. The x-axis plots the *minimum* LR, that is, the LR at the final training step. As the ratio increases from 10$\times$ (right-most point), where the minimum LR is 10% of the proxy-tuned rate, to 0% (D2Z, left-most point), validation loss decreases roughly linearly.

We also tried applying a *Step* decay to a LR that had been following a 10$\times$ or D2Z trajectory; this approach always led to inferior results.

While it is likely possible to improve the quality of *Step* decay by tuning the positioning of the drop, we hypothesize that these efforts will not surpass D2Z, since dropping the LR will fundamentally always result in higher emphasis being placed on earlier updates, as shown in Figure 2 and Figure 25. Moreover, introducing additional tunable hyperparameters (i.e., when and how much to decay) is a further drawback of the *Step* schedule.

B.7 Weight sparsity

In this section, we investigate the role of *Linear*-D2Z in the context of models trained with *unstructured weight sparsity*, a promising direction for improving the efficiency of large neural networks (Hoefler et al., 2021). We parameterize with μP's sparse extension, SμPar (Dey et al., 2024). SμPar allows us to use the same μP hyperparameters as with dense models, except we must now apply corrections due to both model scaling (i.e., ρ, Section 2.2) and layer sparsity. For these experiments, we sparsified all non-embedding layers of our 617M-parameter dense models by randomly fixing certain weights to zero for the duration of training. We trained all models for 11752 total steps using a batch size of 504, i.e., the same amount of training data as we used for training the corresponding 617M-parameter dense models to 20 TPP (Tables 2 and 3).

At 93.75% sparsity ($^1/_{16}$ density), the optimal D2Z model improves by 1.64% over the optimal 10$\times$ model, with a clear trend of optimal LRs shifting lower and loss becoming worse as we go from 10$\times$ to 3$\times$ to *Constant* decay (Figure 19).

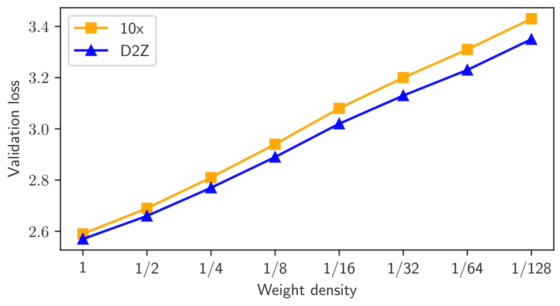


Figure 21: Validation loss for 617M-parameter models trained with SμPar, as unstructured weight sparsity increases (density decreases). All models trained with the same number of training tokens (11752 steps, equivalent to 20 TPP for the fully-dense models). As sparsity increases, the number of trainable parameters decreases, and thus tokens-per-(dense)-parameter increases.

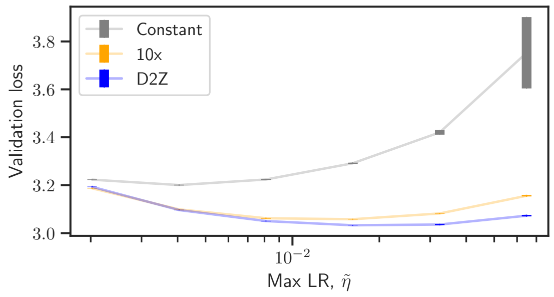


Figure 22: Validation loss variance for different maximum LRs for 111M-parameter μP models trained for 20 TPP, at different decay ratios. Each point corresponds to the mean validation loss over 5 separate training runs with different random seeds; the error bars give the standard deviation. Beyond the instabilities at high learning rates, run-to-run variance is remarkably low at this scale.

At the proxy-tuned max LR of 1.6e-02, D2Z is 2.15% better than 10×. We also trained 93.75% sparse models with a variety of other decay ratios between 10× and D2Z, and present these results in Figure 20. Here we see a largely linear decrease in loss with a linear increase in the decay ratio (i.e., a linear decrease in the minimum LR). These are encouraging findings in the sense that D2Z can seemingly be used directly on a range of problems, without having to worry about tuning a problem-specific LR decay ratio (e.g., 50× or 100×).

In Figure 21, we investigate the gap between D2Z and 10× at the proxy-tuned max LR across different spasity levels. Note that increasing sparsity effectively leads to a corresponding decrease in the number of trainable parameters. Since we use a fixed number of training tokens in each case, as the number of parameters decreases, the number of tokens-per-parameter (TPP) *increases*. In this sense, we note the relative differences between D2Z and 10× are consistent with our results in Figure 4 — as TPP (and gradient noise) increases, D2Z performs relatively better.

B.8 Error bars

Taken as a whole, our results are remarkably stable: empirical results for different model scales, training durations, batch sizes, weight decays, and weight sparsity settings largely behave as predicted by theory. Since all validation runs are performed on 1.1B tokens, any significant run-to-run variance must arise during training. To quantify this variance, we repeated 111M-model 20 TPP training four additional times, resulting in 5 total validation loss results for each original training run. Figure 22 confirms that run-to-run variance is remarkably low. The only significant variance arises in *Constant* results at the highest learning rate. Here, the training loss sometimes spikes at various points during training, and the final validation loss can be significantly higher for models that cannot recover sufficiently following the spike.

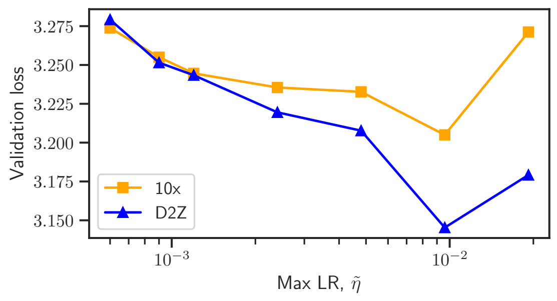


Figure 23: Validation loss for different LR and decay combinations, for a 111M-parameter *NanoGPT model*. With `float16` precision, we were not able to train above 6e-04 without instabilities (first point in curves). Moveover, at $\tilde{\eta}$ = 6e-04, 10× performed better than D2Z. After switching to `float32`, we were able to train at higher $\tilde{\eta}$ values, where D2Z demonstrates its familiar superiority over 10×.

B.9 NanoGPT experiments

We also compared 10× versus D2Z by training NanoGPT models using the NanoGPT codebase (Karpathy, 2024). NanoGPT uses the standard parameterization. We configured these models to be largely similar to our 111M-parameter models, also using a weight decay of 0.1, a context length of 2048, and the GPT-2 vocab size of 50257. Key differences here are that we do not include bias weights, and we trained on the OpenWebText dataset (Gokaslan & Cohen, 2019). Experiments are run on Nvidia A10 GPUs.

As mentioned in Section 5.2, we initially tested NanoGPT in `float16` precision. Here, at the default NanoGPT learning rate of 6e-4, 10× performed slightly better than D2Z. As we pushed the LR 50% higher (to 9e-4), both 10× and D2Z had higher loss, and by 1.2e-3, the loss from 10× doubled.

We suspected that numerical issues may be causing the instabilities, and repeated our experiments in `float32`. At this precision, we were able to successfully increase the LR by factors of two up to 32 times the default. At these levels, we do see the familiar gains of D2Z over 10× (Figure 23). We note there is nothing fundamentally limiting about `float16` precision itself - indeed all our main experiments were done using this precision. Rather, for whatever reason, `float16` is simply problematic in the NanoGPT codebase.

These experiments demonstrate that a comparison between D2Z and 10× may serve as a kind of *diagnostic* of whether a model is being trained at optimal max LRs: if a 100M+ model trained to 20 TPP does not see roughly 1% gains from using D2Z, it is likely the LR is not high enough. In order to raise the LR further, efforts to stabilize the model, perhaps including μP or other techniques (Wortsman et al., 2023), may be warranted.

B.10 Scaling law experiments

Encouraged by the results of D2Z at smaller scales, we began testing D2Z in some of our frontier model efforts. Frontier models do not provide scope for hyperparameter tuning at scale. Thus it becomes important to derive scaling laws to forecast loss at larger scales, based on the loss with a sequence of smaller models.

For this set of experiments, we tested Llama-style (Touvron et al., 2023a) architectures, except using LayerNorm instead of RMSNorm, and multi-head attention instead of group-query attention. We use μP here as well, and a batch size scaling law to determine an optimal batch size for each model scale. These models also use ALiBi embeddings (Press et al., 2022) and SwiGLU (Shazeer, 2020). Here, the context length is 8192 tokens, and we use tied embeddings.

Table 6: Model architecture and batch sizes for scaling law experiments.

Model	d_{model}	n_{layers}	d_{head}
272M	1024	14	64
653M	1536	18	128
1.39B	2048	24	128
2.75B	2560	32	128

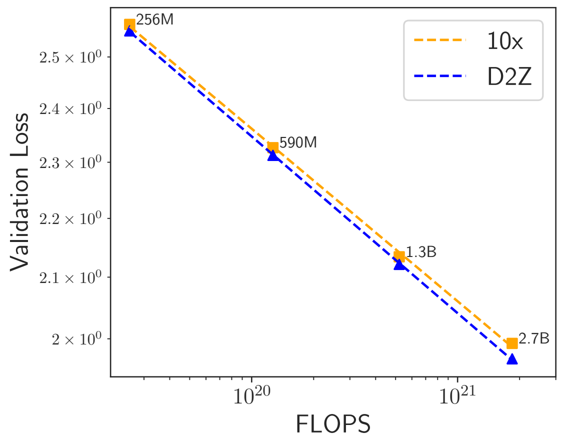


Figure 24: Loss-to-FLOPS scaling law fit for models trained to 20 TPP (8K context lengths, Llama architecture, Pile validation data). The power law fit for D2Z models has a steeper slope than the scaling law for 10× models.

We used a bilingual data mix of English, Arabic and source code samples mixed in a 2:1:0.4 mix ratio. English data is from the Pile (Gao et al., 2020), Arabic uses a proprietary dataset, and the source code comes from the GitHub portion of the Pile.

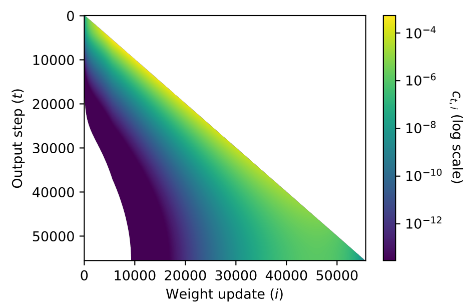
To derive scaling laws to compare D2Z and 10× models, we used the power law functional form $y = cx^m$, where x is the pre-training FLOPs, y is the loss on the Pile validation set, and c and m are parameters to be fit. We trained models at four sizes to compute-optimal 20 TPP (Table 6), and computed total FLOPs spent as well as validation loss on the Pile. We then fit the power law free parameters to obtain our scaling laws.

Encouragingly, here we find the scaling law slope of D2Z is roughly 2.5% better than 10× decay (Figure 24). This translates to an improvement of roughly 1% at 1.3B and 2.7B scales, broadly similar to our earlier results at 1.7B scale. Projecting our scaling law to a 70B model trained to a compute optimal 20 TPP, D2Z would achieve a roughly 2% loss improvement over 10× decay.

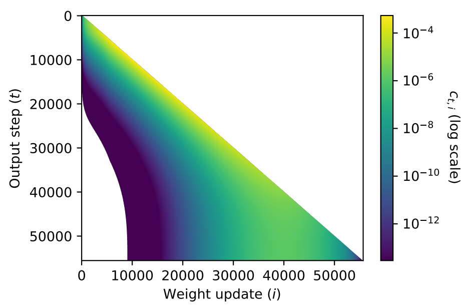
B.11 LR curves and dual coefficients

In this section, we provide some extra figures that were referenced in the main paper. Figure 25 shows the dual coefficients at every step of training, using color to indicate the coefficient value (log-scale). Every horizontal row/step of Figure 25 reflects the coefficients at that step, essentially providing a version of Figure 2 but at each step. Figure 26 provides the LR schedules and dual coefficients for some of the schedules discussed in the main paper, including our proposed *Rational* schedule, which combines all the prior weight updates equally at every step. Finally, Figure 27 gives the LR schedules and dual coefficients for the comparison to *WSD* and *Cyclic* in Figure 7.

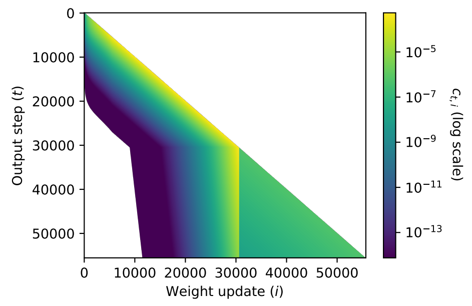
It is worth re-iterating that the dual coefficients can be computed separately from any actual training. They are mathematically equivalent to the LR schedule itself and simply provide a perspective on how the weight updates combine to form parameters using the AdamW optimizer. Furthermore, it is also worth noting the vertical bar at step 0 in the dual coefficient plots; this bar reflects the coefficient on the initial, random weights. To some extent, this $c_{0,t}$ value can serve as an indicator of how far



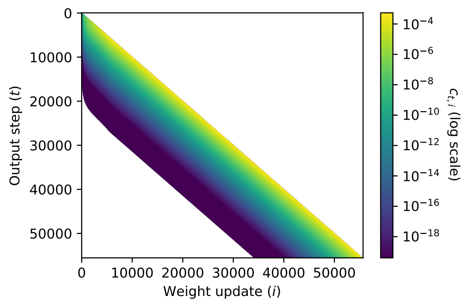
(a) *Linear*



(b) *Cosine*

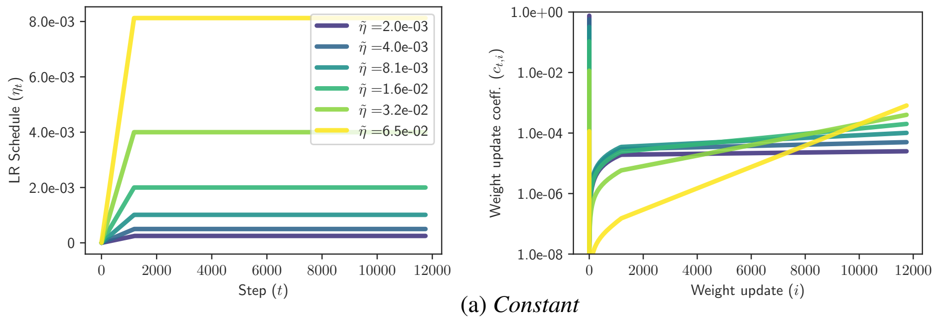


(c) *Step*

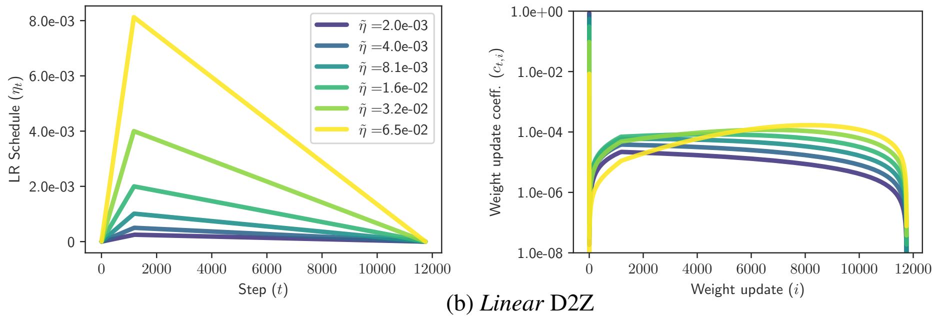


(d) *Constant*

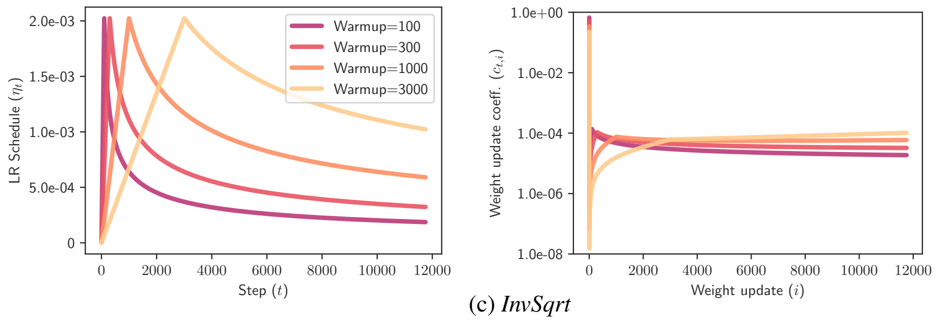
Figure 25: Convex combination of weight updates, with color indicating value of combination coefficients $c_{t,i}$: each $c_{t,i}$ gives the contribution of the ith update (across x-axis) to model weights θ_t across steps t (y-axis). Note that LR schedules and coefficients corresponding to the final step only were presented earlier in Figure 2 (except for *Constant*). Coefficients correspond to settings for 111M-param models trained to 200 TPP: t=55680, $\tilde{\eta}$=1.6e-02, ρ=$1/3$, λ=0.1.



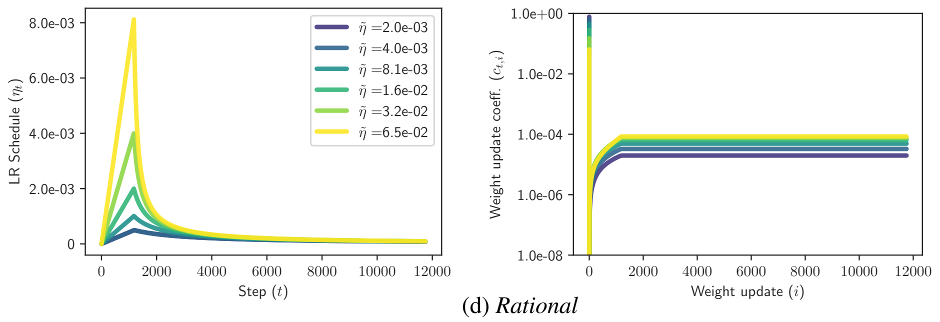
(a) *Constant*



(b) *Linear* D2Z

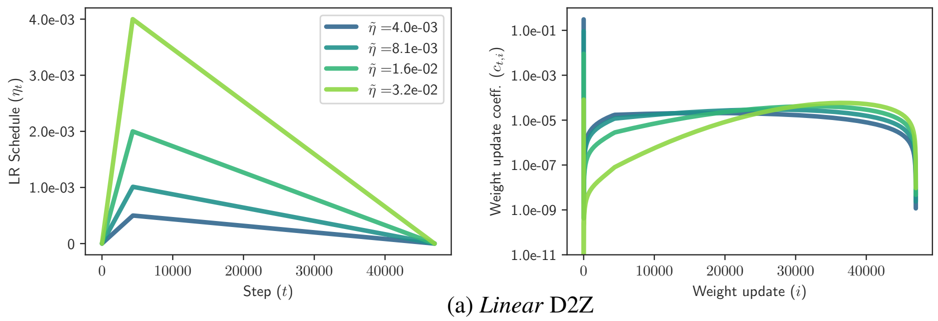


(c) *InvSqrt*

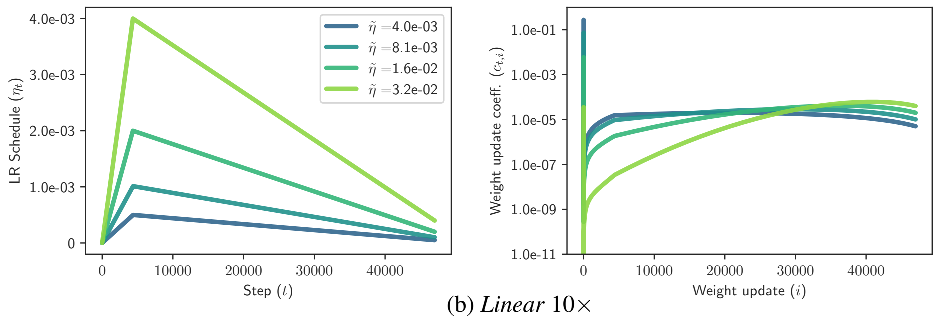


(d) *Rational*

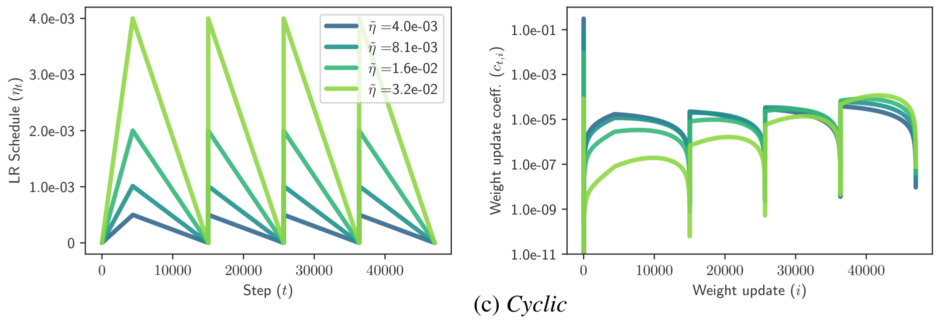
Figure 26: LR curves and dual coefficients for various common LR schedules, as well as the proposed *Rational* approach (Section 3.3). Dual coefficients shown at final training step for 617M-parameter, 20 TPP training (t=11752, ρ=1/8, λ=0.1). For *InvSqrt*, we vary the warmup and fix $\tilde{\eta}$=1.6e-02 for all curves.



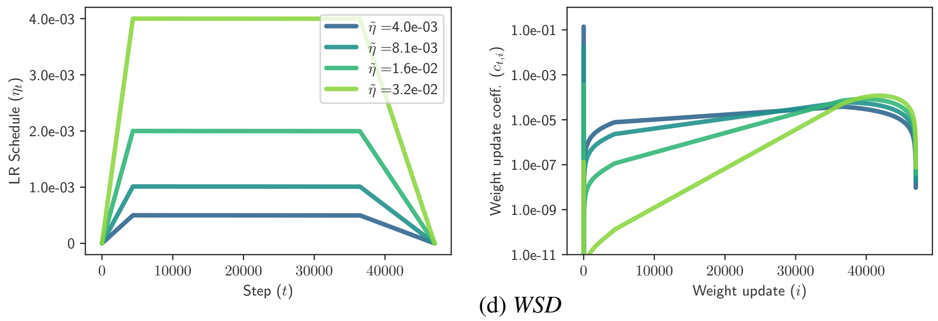
(a) *Linear* D2Z



(b) *Linear* 10×



(c) *Cyclic*



(d) *WSD*

Figure 27: LR curves and dual coefficients comparing standard *Linear* schedules versus *Cyclic* and *WSD* schedules. On the left are the exact LR schedules used in Figure 7 evaluations, i.e., 617M-parameter, 80 TPP training (ρ=1/8, λ=0.1). Dual coefficients shown at at final training step t=47008.

the model has moved from initial conditions (Section 3.2). In general, if $c_{0,t}$ is too high (e.g., when it outweighs the sum of the other coefficients), then bias is likely significantly hindering learning. Two effective ways to reduce $c_{0,t}$ (and also reduce bias) are to (1) raise the max LR, and (2) train for more TPP. Raising λ can also decrease $c_{0,t}$, but is counterproductive for reducing bias because it also reduces the scale of weight updates, as noted in Section 3.2. Likewise, using smaller batches also reduces $c_{0,t}$, but is likewise counterproductive if the batches become too small, to the extent that gradient noise increases excessively. However, D2Z is more robust to such noise than 10$\times$ or *Constant* decay. Further investigating the interplay of λ, batch size, and maximum learning rate is important future work.