TRANSFER PARAMATTERS: OPTIMAL PER-MODULE HYPERPARAMETERS ACROSS ALL SCALING AXES

Anonymous authors
Paper under double-blind review

ABSTRACT

Hyperparameter tuning can dramatically impact training stability of large-scale models. Recent works on neural network *parameterisations*, such as μP, have shown that layer types and sizes should dictate how global hyperparameters should be rescaled in order to achieve efficient transfer across model sizes. On the other hand, the established practice for hyperparameter optimisation *search* is to look for optimal *global* base values that apply at some fixed model scale. We transfer hyperparameters across all scaling axes: width and depth, using an extension of *CompleteP* (Dey et al., 2025), training horizon, and batch size. Our study covers all optimisation hyperparameters of modern models: learning rates, Adam parameters, weight decay, initialisation scales, and residual block multipliers. Lastly, we demonstrate that hyperparameter transfer holds even in the per-layer hyperparameter regime. We characterise the empirical challenges of navigating the high-dimensional hyperparameter landscape, and propose practical guidelines for tackling this optimisation problem. We suggest a simplified parameterisation of the hyperparameter space that reduces the dimensionality of the search-space at no performance cost. Our experiments demonstrate training speed improvements when applying transferred hyperparameters to Large Language Models.

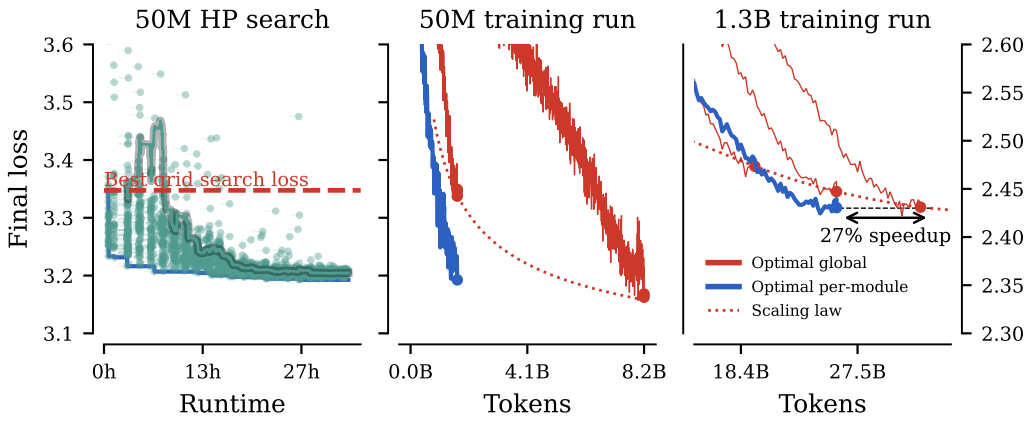


Figure 1: Optimising per-module-type hyperparameters leads to training speed improvements that transfer at scale with Complete$^{(d)}$P .

1 INTRODUCTION

The remarkable success of large transformer-based models (Vaswani et al., 2017) has been driven by scaling up model size and data. However, to get the most out of these large-scale training runs, or to even successfully complete them at all, several hyperparameters (HPs), such as learning rates, weight decay, or initialization scales, must be carefully set.

Parameterisation. To mitigate this HPs tuning cost, recent works have introduced principled *parameterisations*, such as the μ-parameterisation (μP) (Yang et al., 2022), with the goal of enabling

the *transfer* of optimal global hyperparameters from smaller, cheaper-to-train models to their large-scale counterparts. Effectively, these parameterisations propose to automatically adapt a global seeded HP to any layer, depending on its width or type. This process has been extended to handle changes in depth with Depth-μP (Yang et al., 2024) and further investigated for width and depth in transformers with CompleteP (Dey et al., 2025). These methods have been demonstrated to successfully transfer optimal global hyperparameters.

Per-module HP. Given the significant performance improvements and cost reduction from optimising HPs on smaller-scale experiments, it is natural to consider optimising HPs on a finer-grained scale as well, and explore per-module HPs. When one scales-up a model with μP, Depth-μP or CompleteP, different layers will receive different HPs depending on their architectural role – for instance, the learning rates for the embedding layers have to be scaled differently from those for the hidden weights. It is therefore reasonable to expect that different layers could benefit from independent hyperparameter tuning. Put differently, there is little reason to believe the optimal per-module HPs should all collapse to the same value at *some* base width at which we optimise them.

Parameterisation-aware per-module HP Optimisation. In this work, we systematically investigate the *transfer* of per-module hyperparameters across various scaling modalities. The challenge, however, is one of scale: tuning hyperparameters on a per-module basis creates a combinatorial explosion in the search space, making it truly intractable at large scale. We propose a practical methodology to unlock the benefits of per-module tuning by leveraging the power of HP transfer using parameterisations. We perform the expensive, high-dimensional search for optimal per-module HPs on a small proxy model, and demonstrate the transfer of the optimal HPs to a large target model. Our contributions are:

- **Complete$^{(d)}$P** . We refine the CompleteP parameterisation from Dey et al. (2025), extending it to modern Transformer components like Query-Key Normalization (Henry et al., 2020). We further identify and rectify minor issues in the original formulation. We illustrate the resulting parameterisation permits robust hyperparameter transfer for all theoretically-motivated variants of depth scaling ($\alpha \in \left[\frac{1}{2}, 1\right]$).
- **New scaling directions for HP transfer.** We systematically study transfer beyond model size, including in token horizons and batch size. We make new recommendations for weight-decay scaling with batch-size adapting the SDE approach of (Malladi et al., 2022) to AdamW.
- **Per-module HP transfer.** We empirically demonstrate hyperparameter transfer with the right parameterisations holds for per-module hyperparameters. Optimising per-module hyperparameters at a small scale yields significant training speed-ups that persist after transfer to larger scale.
- **A practical recipe to find per-module HPs.** We empirically characterise the per-module hyperparameter optimisation landscape. We highlight its challenging nature, marked by sharp "cliffs" where training diverges, making random search inefficient and standard Bayesian Optimisation ineffective. We show that the landscape is close to "invex", and demonstrate that simple *local* search strategies are well-suited to navigate this space and find high-performing configurations.

2 Hyperparameter Transfer

In this section, we describe the hyperparameter transfer modalities we consider, and the principles that we follow to adjust HPs while varying other aspects of the training configuration.[1] We first describe hyperparameter transfer in model size (width and depth) in Section 2.1, where we introduce a variant of the CompleteP parameterisation (Dey et al., 2025). In Section 2.2, we describe principles we follow for hyperparameter transfer across batch-size. Lastly, we consider hyperparameter transfer in the number of training tokens in Section 2.3, illustrating that optimal HPs do not transfer out of the box across token horizons.

Experimental Setup All experiments are conducted using a decoder-only transformer model (Radford et al., 2019; Phuong & Hutter, 2022) on the RedPajama dataset (Weber et al., 2024). We use a modern transformer variant with pre-normalisation, Query-Key normalisation (Henry et al.,

[1]To disambiguate, we'll refer to *hyperparameters* as aspects of training we want to find optimal values for, which we contrast with *training configuration* – the aspects of training we want to control to facilitate scaling (number of training tokens, number of parameters, batch-size) that are typically integers.

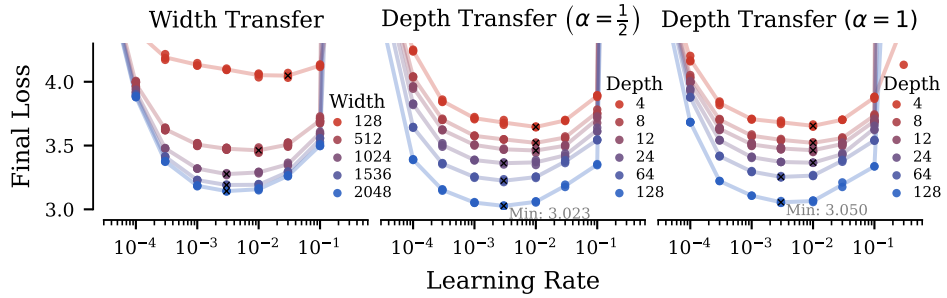


Figure 2: Hyperparameter transfer for global learning rate across depth and width.

2020), trained with a mixture of cross-entropy and Z-loss (de Brébisson & Vincent, 2016). We always train with a cosine schedule. As a performance metric, we always report the final validation loss on the pre-training data, which is a strong indicator of downstream performance. For remaining training and architecture details, see Appendix C.

2.1 Hyperparameter Transfer across Model Size

The core idea underlying hyperparameter transfer across models of different sizes is to view models as discretisations of infinite-size limits. Intuitively, two models of different sizes that are both sufficiently close to the same infinite limit will behave similarly, and if their infinite limits are the same over the set of considered hyperparameters, then they should share similar optimal hyperparameters.

The challenge is that, depending on the *parameterisation* — i.e. the rules for adjusting the hyperparameters as a function of size — we can obtain different infinite width or depth limits with fundamentally different behaviours (Yang & Hu, 2021). Most of these limits are pathological in various ways. For instance, the Standard Parameterisation (SP) (Sohl-Dickstein et al., 2020) leads to the features blowing up with size, whereas the Neural Tangent Parameterisation (Jacot et al., 2018) results in a lack of *feature learning* (Yang & Hu, 2021). μP was identified by Yang & Hu (2021) as the unique parameterisation for Stochastic Gradient Descent (and later for a broad class of adaptive algorithms (Yang & Littwin, 2023)) that precludes the emergence of many such pathologies at scale.

In this work, we build upon the CompleteP (Dey et al., 2025), which itself is an adaptation of Depth-μP to transformers, to which we make several adaptations. These new scaling rules, which we call **Complete$^{(d)}$P** , are summarised in Table 1. Firstly, we extend the parameterisation to Query-Key (QK) normalisation layers (Henry et al., 2020), which have become a staple in modern transformer implementations (Yang et al., 2025; Dehghani et al., 2023). The challenge of QK norms is that, unlike any other component in transformers, these layers share weights across transformer heads. If scaling in width is performed by increasing the number of heads while keeping the head dimension fixed (as was done in (Dey et al., 2025; Yang & Hu, 2021)), then QK norms introduce weight-sharing across the scaled dimensions. This necessitates different scaling considerations than for regular normalisation layer multipliers or biases. The adjustments for AdamW (Loshchilov & Hutter, 2017) are shown in Table 1, which we justify in Appendix B.

Secondly, we note that Dey et al. (2025) mistakenly derived the wrong scaling for the AdamW ϵ scaling for the input embedding. We justify our modification in the Appendix B. Although the resulting modification is minor, we found that the lack thereof was sufficient to break a thorough sweep of the coordinate checks described by Yang et al. (2022) in our implementation.

Lastly, we eliminate the explicit scalar multiplier on the output of the final linear projection, $f : \mathbb{R}^E \to \mathbb{R}^V$, by reparameterising its effect into the learning rate and initialisation scale. This enables memory-efficient algorithms like Cut Cross-Entropy (Wijmans et al., 2025), which avoid materialising the full $V \times E$ projection matrix, drastically reducing GPU memory requirements for modern large vocabulary models (Kamath et al., 2025; Dubey et al., 2024; Agarwal et al., 2025).

In Figure 2, we verify the HP transfer with Complete$^{(d)}$P across width and depth. An important factor in Depth-μP is the depth-dependent re-scaling factor for the residual connection in transformers ($\mathbf{h}^{\ell}$

Table 1: **Parameterisation Comparison** as a function of width (m_N) and depth (m_L) ratios. For Complete$^{(d)}$P , differences to CompleteP (Dey et al., 2025) are shown alongside in gray.

	Parameterisation:	μP (Table 3)	Complete$^{(d)}$P
Multipliers	MHA Residual	$\mathbf{x} + \texttt{MHABlock}(\mathbf{x})$	$\mathbf{x} + m_L^{-\alpha}\texttt{MHABlock}(\mathbf{x})$
	MLP Residual	$\mathbf{x} + \texttt{MLPBlock}(\mathbf{x})$	$\mathbf{x} + m_L^{-\alpha}\texttt{MLPBlock}(\mathbf{x})$
	Unemb. Fwd	*Unaugmented*	*Unaugmented* $[\times(m_N^{-1}]$
Init. Variances	Input Emb.	σ^2_{base}	σ^2_{base}
	Hidden weights	$\sigma^2_{\text{base}} \cdot m_N^{-1}$	$\sigma^2_{\text{base}} \cdot m_N^{-1}$
	Hidden biases/norms	σ^2_{base}	σ^2_{base}
	Unemb. LN	σ^2_{base}	σ^2_{base}
	Unemb. Weights	$\sigma^2_{\text{base}} \cdot m_N^{-2}$	$\sigma^2_{\text{base}} \cdot m_N^{-2}$ $[\sigma^2_{\text{base}}]$
Learning Rates	Input Emb.	η_{base}	η_{base}
	Hidden weights	$\eta_{\text{base}} \cdot m_N^{-1}$	$\eta_{\text{base}} \cdot m_N^{-1} \cdot m_L^{\alpha-1}$
	Hidden biases/norm	η_{base}	$\eta_{\text{base}} \cdot m_L^{\alpha-1}$
	Unemb. LN	η_{base}	η_{base}
	Unemb. weights	$\eta_{\text{base}} \cdot m_N^{-1}$	$\eta_{\text{base}} \cdot m_N^{-1}$ $[\eta_{\text{base}}]$
AdamW ϵ	Hidden weights/biases/norms	$\epsilon_{\text{base}} \cdot m_N^{-1}$	$\epsilon_{\text{base}} \cdot m_N^{-1} \cdot m_L^{-\alpha}$
	QK norms	NA	$\epsilon_{\text{base}} \cdot m_L^{-\alpha}$ [NA]
	Input Emb.	$\epsilon_{\text{base}} \cdot m_N^{-1}$	$\epsilon_{\text{base}} \cdot m_N^{-1}$ $[\epsilon_{\text{base}}]$
	Output weights/biases/norms	ϵ_{base}	ϵ_{base}
Weight decay	Hidden weights	$\times m_N$	$\times m_N$
	Unemb. weights	$\times m_N$	$\times m_N$
	Rest	Unchanged	Unchanged

the output of layer ℓ, $\mathcal{F}_\ell$ the function applied to it):

$$\mathbf{h}^{\ell+1} = \mathbf{h}^\ell + m_L^{-\alpha}\mathcal{F}_\ell(\mathbf{h}^\ell), \quad \ell \in \{1, \dots, L\}$$

which is governed by a single parameter $\alpha \in [0.5, 1]$.We make the following observation:

Complete$^{(d)}$P with $\alpha = \frac{1}{2}$ and $\alpha = 1$ permits hyperparameter transfer across depth.

Our parameterisation seems to allow for HP transfer with all theoretically justified values of $\alpha \in \left[\frac{1}{2}, 1\right]$. This is in contrast to the findings of Dey et al. (2025) who notice a degradation of transfer for $\alpha = 0.5$. The added QK norms in our implementation improve stability (see Figure 14 for a comparison without); however, removing them does not lead to the breakdown of transfer reported by Dey et al. (2025). We note that in their publicly-released reference implementation, they apply the same AdamW's ϵ to all weights (including embeddings), against their own paper recommendation. Interestingly, the optimal loss is slightly better for the largest model for $\alpha = \frac{1}{2}$, potentially suggesting that the theoretical arguments for this parameterisation on the basis of *feature diversity* (Yang et al., 2024) might be beneficial in a language transformer context.

2.2 Hyperparameter transfer across batch-size

Model size and dataset size are two levers to achieve lower loss – increasing each predictably leads to model improvements as implied by scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022). This, in turn, requires scaling up significantly the compute budget, which is only feasible through parallelization. In that context, the set of usable batch-sizes is heavily constrained and largely dictated by the memory configuration of a specific parallel architecture. However, training for long token horizons is challenging with a small batch-size, as it demands many more sequential training iterations. On the one hand, for smaller hyperparameter sweep runs, a smaller batch-size is often desirable to reduce the per-run memory footprint and enable running on fewer GPUs. On the other hand, that batch-size will no longer be suitable for larger runs. Unfortunately, as with scaling model

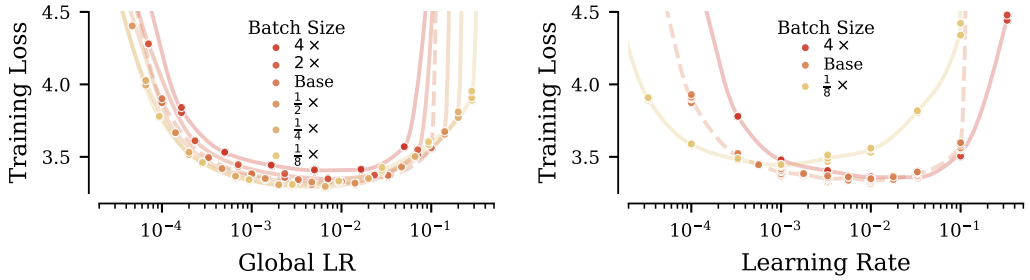


Figure 3: **Learning rate transfer with batch-size. Left:** Learning rates transfer when using the square-root rule in Equation 2 **Right:** Learning rates fail to transfer without adjustment.

size or training duration, hyperparameters do not transfer across batch-sizes without further reparameterisation. In this work, we transfer hyperparameters across batch-size via a similar limiting argument as for transfer across model size. In particular, we follow and extend the principles for batch-size transfer laid out in Malladi et al. (2022).

Training as discretising an SDE We consider the same simplifying example as in Malladi et al. (2022). We consider that the gradients queried at each step k are a noisy version of a fixed direction $\boldsymbol{g}^k = \boldsymbol{g} + \sigma \boldsymbol{e}^k$, where $\boldsymbol{e}^k$ are i.i.d. Gaussian vectors of identity covariance. We further use the RMSProp algorithm as an example, and place ourselves in the high-variance regime, where $\sigma \gg \|\boldsymbol{g}\|$. Contrary to Malladi et al. (2022), we also consider a weight decay term as in AdamW. We let η the learning rate, and λ the weight decay. We obtain the simplified RMSProp iterations (see (Malladi et al., 2022, Sec. 4.1) for more details) that define the iterates $\boldsymbol{\theta}(k; \eta, \lambda, \sigma)$ with the equation

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \left(\frac{\boldsymbol{g}^k}{\sigma} + \lambda \boldsymbol{\theta}^k \right) = \boldsymbol{\theta}^k - \frac{\eta}{\sigma} \left(\boldsymbol{g} + \lambda \sigma \boldsymbol{\theta}^k \right) - \eta \boldsymbol{e}^k, \tag{1}$$

which is a discretization of the SDE $d\Theta_t = \frac{1}{\eta\sigma}(\boldsymbol{g} + \lambda\sigma\Theta_t)dt + dW_t$ with step-size η^2, in the sense that $\boldsymbol{\theta}^k \simeq \Theta_{k\eta^2}$. Therefore, we find that the multipliers to keep fixed iterate distributions, i.e., such that $\boldsymbol{\theta}(k; \eta, \lambda, \sigma) = \boldsymbol{\theta}(m_k k; m_\eta \eta, m_\lambda \lambda, m_\sigma \sigma)$, should verify $m_k m_\eta^2 = 1$, $m_\eta m_\sigma = 1$, and $m_\lambda = m_\eta$. In particular, if the batch-size is multiplied by κ, we have $m_\sigma = \kappa^{-1/2}$ and we find that the new hyperparameters matching the SDE limit should follow the square-root scaling rule

$$\eta' = \sqrt{\kappa}\eta, \quad k' = \kappa k, \text{ and } \lambda' = \sqrt{\kappa}\lambda \ . \tag{2}$$

To the best of our knowledge, we are the first to identify this scaling rule for the weight decay λ. We report the effect of using these multipliers when scaling the batch-size in Figure 3; using the square-root rule is critical to get LR transfer. We also isolate the effect of scaling the weight decay in Figure 4, demonstrating that the rule identified above is critical for transfer.

AdamLH and multipliers Equation 1 mirrors the Pytorch implementation of AdamW, where the weight decay λ is multiplied by the learning rate η. If one instead uses the original AdamW implementation, often coined AdamLH, as proposed by Loshchilov & Hutter (2017), we get the simplified iterations $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \frac{\boldsymbol{g}^k}{\sigma} - \lambda \boldsymbol{\theta}^k$, and we find that the multipliers are the same as for AdamW, except that $m_\lambda = m_\eta^2$: doubling the batch-size means that the weight decay should now be doubled. Hence, using AdamLH leads to a bigger drift across batch-sizes if the scaling is not done correctly, amplifying further the drift observed in Figure 3, right. We posit that this is one of the reasons why the Pytorch implementation is more widely used.

2.3 Hyperparameter transfer in training duration

Unlike transfer in model size or batch-size, transfer in the token horizon has received comparatively less attention in the literature. Nonetheless, it is one of the two main levers to scaling compute. Like Bjorck et al. (2025), we observe that the optimal learning rate decays with the number of training iterations, holding all other things constant. **Optimal learning rate keeps SDE time constant.** In Figure 5, we notice that the optimal learning rate decays at a rate roughly proportional to $\frac{1}{\sqrt{\kappa}}$, where κ is the factor by which we've increased the number of training iterations. We plot the

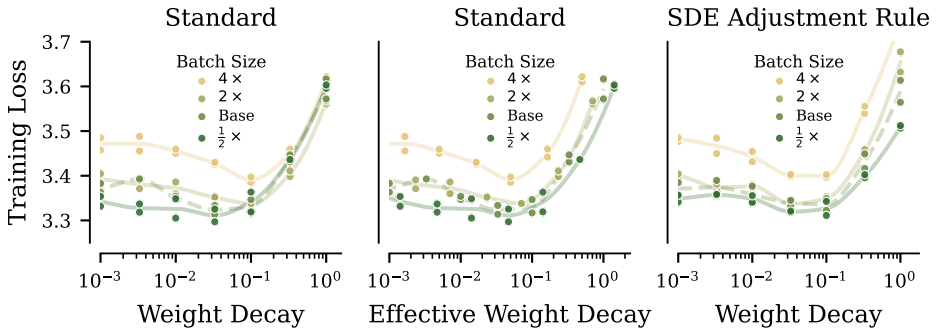


Figure 4: **Weight decay transfer with batch-size. Left:** Weight decay fails to transfer with batch-size without any adjustments. **Middle:** The rescaled (effective) weight decay $\lambda/\sqrt{\kappa}$ where κ is the increase does transfer. **Right:** The effective weight decay transfers when rescaling all hyperparameters following our AdamW SDE scaling rule.

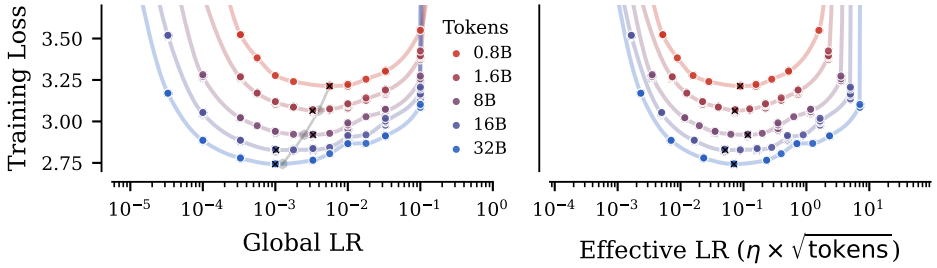


Figure 5: Learning rate transfer across training horizon – adjusting the number of tokens by changing the number of training iterations while holding batch-size constant. **Left:** Break-down of transfer of the global learning rate. **Right:** Stability of the "effective" learning rate – one that preserves the AdamW SDE integration horizon.

optimal learning rate from the shortest training duration (η_{opt}) transferred with the scaling rule $\frac{\eta_{\text{opt}}}{\sqrt{\kappa}}$ in gray, and observe that it aligns almost perfectly with the true optima. In contrast, Bjorck et al. (2025) fit a scaling law to find the exponents β for the scaling rule $\frac{\eta_{\text{opt}}}{\kappa^{\beta}}$; and they identify β to be in the ranges of $0.3 - 0.7$ depending on the model size, which matches our scaling rule.

Square-root reparameterization for learning rate permits transfer across training horizon

In the light of the SDE interpretation in subsection 2.2, scaling the learning rate by $\frac{1}{\sqrt{\kappa}}$ while holding the batch-size constant can be seen as reducing the signal-to-noise (SNR) ratio in the SDE, while keeping the time horizon constant. Indeed, we orthogonally observe that when simulating the Adam SDE from (Malladi et al., 2022), improving the signal-to-noise ratio (i.e. reducing the size of the diffusion coefficient) while holding other parameters constant consistently leads to improved performance. Hence, we hypothesise the right way to scale the token horizon might be only adjusting the signal-to-noise parameter in the AdamW SDE, while keeping all other things constant. We validate this empirical observation in Figure 9, where we scale the number of tokens by increasing batch-size only (which has the desired effect of changing the signal-to-noise ratio); we observe a near-perfect learning rate transfer across the token horizon. We expect this transfer to break at larger batch-sizes, where the discretisation will be too coarse for AdamW to approximate the underlying SDE, but when taken together with the batch-size reparameterisation rules in subsection 2.2, this finding suggests how to scale all HPs across token horizons while choosing the batch-size freely. This is the token horizon scaling procedure we follow in all the per-module HP results in the remainder of this paper. We note that this finding might be specific to the fixed (cosine) schedule that we use.

Best Learning Rate (LR) annealing at different token horizons. Optimal schedules might have different shapes at different token horizons (Luo et al., 2025). We conduct a greedy search to de-

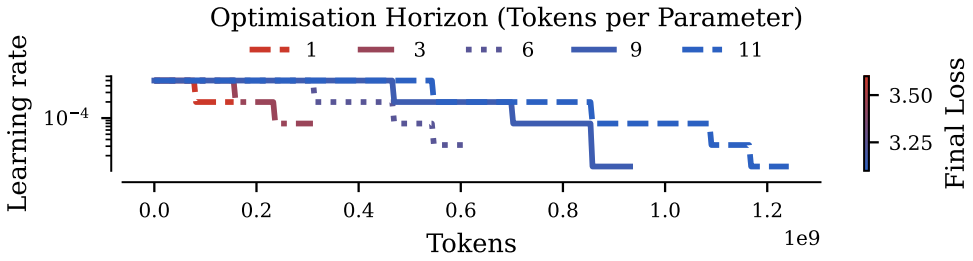


Figure 6: **Best Learning Rate annealing over 4,842 runs for five different token horizons**. The best schedule at a short horizon is never a prefix for the best schedule at a longer horizon. The optimal schedule cannot be found by a greedy approach: the best LR annealing is not data agnostic.

termine optimal learning rate schedules in the following way. We enumerate all the non-increasing piecewise-constant LR schedule over the discrete set $\{0.0015/2.5^k | 0 \leq k \leq 4\}$. We sub-divide the total training duration in 16 intervals of 77M tokens each. At the end of each interval, either the LR remains constant, either it is decayed by one or more steps. For five different token horizons, we report the best scheduling among the 4,842 tested. We report the results in Figure 6. We notice that the best scheduling at short horizon is never a prefix of the best scheduling at long horizon. This empirical observation is compatible with the findings of Luo et al. (2025): there is a tension between the optimisation bias induced by the terminal LR value (the lower the better) and the progress of optimisation which requires higher LR values at start.

3 Investigating transfer of per-module hyperparameters

Equipped with the tools for hyperparameter transfer described in the preceding section, in this section we investigate 1) how much there is to gain from per-parameter hyperparameter optimisation, and 2) how well do per-module hyperparameters transfer.

3.1 Optimising per-module hyperparameters

To show improvements and transfer of per-module hyperparameters, we need a good way to optimise them at a fixed scale. Although there is ample literature on hyperparameter optimisation in deep learning (Snoek et al., 2012; Maclaurin et al., 2015; Lorraine et al., 2020), optimising HPs on a per-module basis introduces many new difficulties. Below, we highlight why many standard approaches fail in the per-parameter optimisation setting.

Per-module hyperparameter loss landscape In Figure 7, we plot slices through the per-module learning rate (LR) loss landscape — i.e. the landscape of the mapping from LRs to the final loss. We observe that, fortuitously, it's pretty close to being invex (stationary points are global minima), and hence might be tractable even despite its high dimensionality. Several other aspects, however, render it challenging for common HP optimisation methods: **1)** The values of per-module learning rates at which training becomes unstable are module-dependent, and can differ by multiple orders of magnitude. **2)** The boundary at which training becomes unstable has a complex shape, with non-trivial interactions among different modules, implying it's difficult to predict with simple predictive models (e.g. linear models or Gaussian Processes (Williams & Rasmussen, 2006)). Our observation is similar to that made by Sohl-Dickstein (2024) — who observed the stable regime boundary is a fractal — but we also note a lack of an emergent simple structure at the macro scale. This means common hyperparameter optimisation strategies, like random search or standard Bayesian Optimisation, fail in this regime. For instance, random search lacks any locality bias; we observe that without careful manual tuning of the search boundaries, either all runs will fail due to unstable training, or the boundaries will fail to include the actual optimum. Bayesian Optimisation with Gaussian Processes (GPs) can exploit locally around previous good trials. However, it is well-known that GPs struggle on highly non-stationary data (Snoek et al., 2014), and alternatives capable of dealing with the non-stationarity can scale poorly. We found that, in contrast, 'trust region' methods —

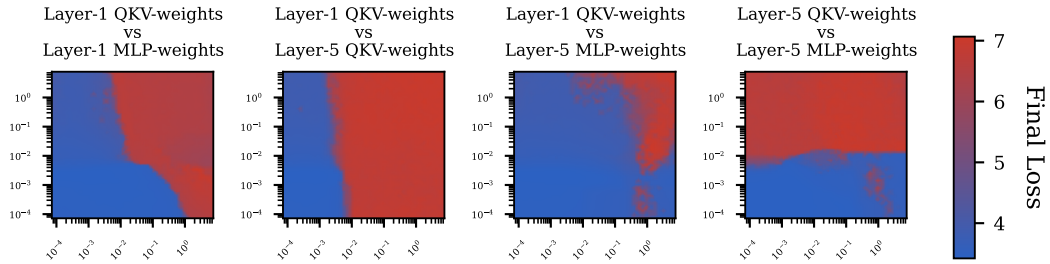


Figure 7: The boundary for stable training has a complex shape. Each plot shows the final training loss for different combination of learning rates for the modules indicated, while fixing the remaining learning rates to the optimal "global" value. MLP weights refer to the MLP in the attention layer. If unstable training results in NaNs, the last stable training loss is reported.

approaches that optimise in neighbourhoods of previous best solutions — work well. We describe a simple trust region variant of random search that we use for our experiments in subsection B.4.

Parameterising per-module hyperparameters We adopt a depth–type Kronecker parameterisation of per-module hyperparameters that is compatible with Complete$^{(d)}$P transfer across width and depth. Let $\mathcal{M}$ index module types in a Transformer (QKV and attention output projections, SwiGLU-MLP up/down and gate, layer norms, input embeddings, unembedding, and QK-normalisation scalars), and let $\ell \in 1, \ldots, L$ index depth. For a hyperparameter $\theta \in \eta, \lambda, \epsilon$:

$$\log \theta_{g,\ell}(T; N, L, B) = \underbrace{\log \theta_0(T, B)}_{\text{SDE/time–batch}} + \underbrace{\log s_g(N, L)}_{\text{Complete}^{(d)}\text{P}} + \underbrace{\log \alpha_g}_{\text{type}} + \underbrace{\log \delta_\ell}_{\text{depth}}, \tag{3}$$

where $\theta_0(t; B)$ carries all training-horizon T and batch-size B dependence via the AdamW SDE-based transfer rules, $s_g(N, L)$ is the Complete$^{(\mathbf{d})}$P scale map for type g (including our QK-norm and unembedding refinements; Table 1), and $\{\alpha_g : g \in \mathcal{M}\}$, $\{\delta_\ell : \ell \in \{1, \ldots, L\}\}$ are dimensionless, time-invariant multipliers shared across the global schedule. This factorisation reduces the number of free multipliers from $|\mathcal{M}|L$ to $|\mathcal{M}| + L$ while preserving the base width/depth transfer. Empirically, per-depth multipliers are beneficial beyond type-only tuning, and relaxing the Kronecker constraint to fully uncoupled per-layer multipliers yields little additional gain at substantially higher search cost (Figure 11b). Moreover, the per-module loss landscape exhibits sharp stability boundaries (Figure 7), so we optimise the multipliers in log-space using a trust-region random search (subsection B.4) while keeping the cosine schedule shared across modules. When transferring to a larger depth, we note that the depth-SDE ($\alpha = \frac{1}{2}$) or depth-ODE limits ($\alpha \in (\frac{1}{2}, 1]$) should still exist if the base hyperparameters $\delta(t)$ vary continuously with sufficient regularity across depth $t \in [0, 1]$. In this sense, the finite-depth multipliers $\delta_\ell \approx \delta(\frac{\ell}{L})$ can be seen as a discretisation of the continuous limit HPs. Hence, to transfer to a large depth we simply linearly interpolate all HPs in depth.

3.2 Per-module hyperparameters matter

Depth multipliers matter We ablated away the effect of the learning rate per-depth multipliers, by instead considering only a search over learning rate multipliers for each layer *role*: Within each residual block, every parameter group gets an independent learning rate, which is shared *across* different residual blocks; similarly, each parameter group in the embedding and unembedding layers gets its own value. We initiate this search from a projection of the best per-module hyperparameters onto this linear subspace. In Figure 11c, we observe that the search value, although still substantially better than the best global learning rate, is worse than the one that includes per-depth multipliers. Hence, while **the majority of the gain comes from different module types within residual blocks getting different learning rates**, there is still notable benefit to per-depth multipliers.

How restrictive is the depth-Kronecker factorisation? To check how much performance we're leaving on the table with the depth-Kronecker factorisation constraint, we continue searching for fully uncoupled per-layer learning rates from the optimal Kronecker-factorised ones. The search results are shown in Figure 11b. Crucially, we observe virtually no improvement over the Kronecker-factorised ones. While this is not conclusive evidence that the optimal per-module learning rates are

depth-Kronecker factored – the fully uncoupled search-space is much higher-dimensional and more difficult to navigate, and its likely we didn't find the optimum – these runs imply that most of the benefits of per-module HP optimisation can be captured by Kronecker factorised learning rates.

3.3 Optimal per-module hyperparameters transfer with scale

Demonstrating upsides of per-module HP optimisation would be of little practical use if the HPs have to be tuned at the target model scale. In this section, we show that the improvements *do* persist across different model scales. Firstly, we demonstrate transfer in *model size*. Figure 8 illustrates that the optimal per-module learning rates transfer as we scale up both width and depth. Although we cannot easily visualise how the per-parameter HP loss landscape shifts as we vary model size (like was shown in Figure 2) due to its high-dimensional nature, we instead show the final training losses for a slice (a hyperplane) going through both the scaled-up optimal per-module learning rates and the optimal global learning rate. We observe that this landscape appears stable with model size, suggesting that the optimal per-module LRs do transfer with width and depth.

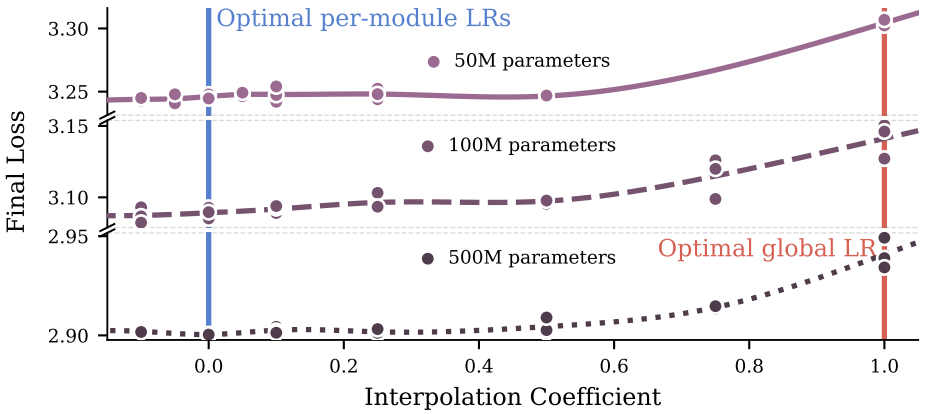


Figure 8: Transfer across model scale of the optimal per-module learning rates. We interpolate between the optimal global learning rate multiplier, and the optimal per-module multipliers across models of different scales, and show that the optimal per-module multipliers **1)** consistently improve upon the global multiplier baseline, and **2)** remain close to optimal in the hyperplane spanned by the optimal global and local multipliers.

Transferring all per-module HPs across the compute optimal horizon. We also investigate what improvements are possible when transferring per-module HPs to a compute-optimal model at the 1B parameter scale. Here, we jointly optimise the per-module learning rate, weight-decay, AdamW $\beta_1, \beta_2, \epsilon$ and initialisation scale, and the residual block multipliers. We continue the search from the optimal per-module learning rates identified with search in Figure 11a at the 50M parameter & 1.6B token scale. In Figure 1, we show that when transferred to the 1.3B & 26B token scale ($420\times$ compute) the optimal per-module HPs lead to a 27% speed-up to reach equivalent loss over the optimal global HP baseline.

4 Conclusion

In this paper, we proposed new transfer rules for hyper-parameters, valid across all scaling axes: model's width, model's depth, token horizon, and batch size. Furthermore, these transfer rules also hold for *per-module* hyper-parameters. We demonstrate that systematic optimisation at small scale with trust regions methods produce a configuration that transfers to larger scale, and significantly improve training speed.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita

Brett, Eugene Brevdo, Greg Brockman, Sébastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam P. Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card. *CoRR*, abs/2508.10925, 2025. doi: 10.48550/ARXIV.2508.10925. URL https://doi.org/10.48550/arXiv.2508.10925.

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal lr across token horizons. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Representation Learning*, volume 2025, pp. 83640–83657, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/cffa22c56c0df3b3edb1df8a9ad67804-Paper-Conference.pdf.

Alexandre de Brébisson and Pascal Vincent. The z-loss: a shift and scale invariant classification loss belonging to the spherical family, 2016. URL https://arxiv.org/abs/1604.08859.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International conference on machine learning*, pp. 7480–7512. PMLR, 2023.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102, 2006.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers, 2020. URL https://arxiv.org/abs/2010.04245.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Róbert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucinska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju-yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle K. Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clément Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry (Dima) Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report. *CoRR*, abs/2503.19786, 2025. doi: 10.48550/ARXIV.2503.19786. URL https://doi.org/10.48550/arXiv.2503.19786.

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. URL https://arxiv.org/abs/2001.08361.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the*

Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of *Proceedings of Machine Learning Research*, pp. 1540–1552. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/lorraine20a.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Kairong Luo, Haodong Wen, Shengding Hu, Zhenbo Sun, Maosong Sun, Zhiyuan Liu, Kaifeng Lyu, and Wenguang Chen. A multi-power law for loss curve prediction across learning rate schedules. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=KnoS9XxIlK.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2113–2122, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/maclaurin15.html.

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.

Mary Phuong and Marcus Hutter. Formal algorithms for transformers. *arXiv preprint arXiv:2207.09238*, 2022.

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

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. Input warping for bayesian optimization of non-stationary functions. In *International conference on machine learning*, pp. 1674–1682. PMLR, 2014.

Jascha Sohl-Dickstein. The boundary of neural network trainability is fractal, 2024. URL https://arxiv.org/abs/2402.06184.

Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. Redpajama: an open dataset for training large language models. *NeurIPS Datasets and Benchmarks Track*, 2024.

Erik Wijmans, Brody Huval, Alexander Hertzberg, Vladlen Koltun, and Philipp Kraehenbuehl. Cut your losses in large-vocabulary language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=E4Fk3YuG56.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Greg Yang and Edward J. Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11727–11737. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/yang21c.html.

Greg Yang and Etai Littwin. Tensor programs ivb: Adaptive optimization in the infinite-width limit, 2023. URL https://arxiv.org/abs/2308.01814.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer, 2022. URL https://arxiv.org/abs/2203.03466.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: Feature learning in infinite depth neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=17pVDnpwwl.

CONTENTS

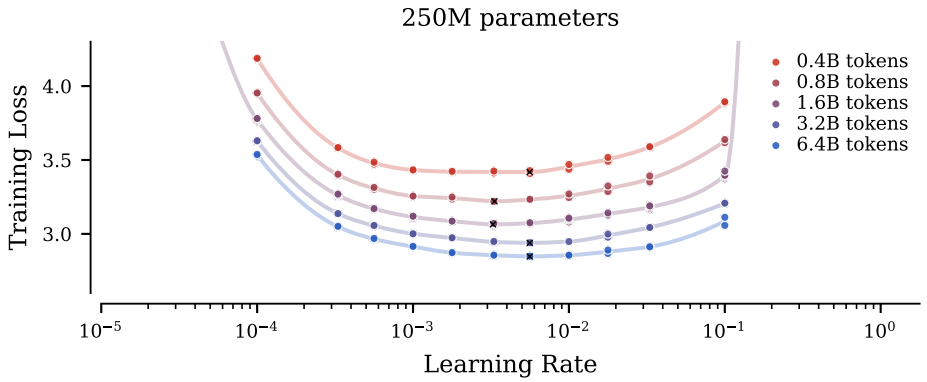


Figure 9: Learning rate transfer across token horizon when scaling up by increasing batch-size while holding training iterations constant. This scaling rule can be seen as improving the gradient signal-to-noise (SNR) ratio in the discretised AdamW SDE (Malladi et al., 2022), while holding all the other SDE parameters and the integration horizon fixed.

A Additional Figures

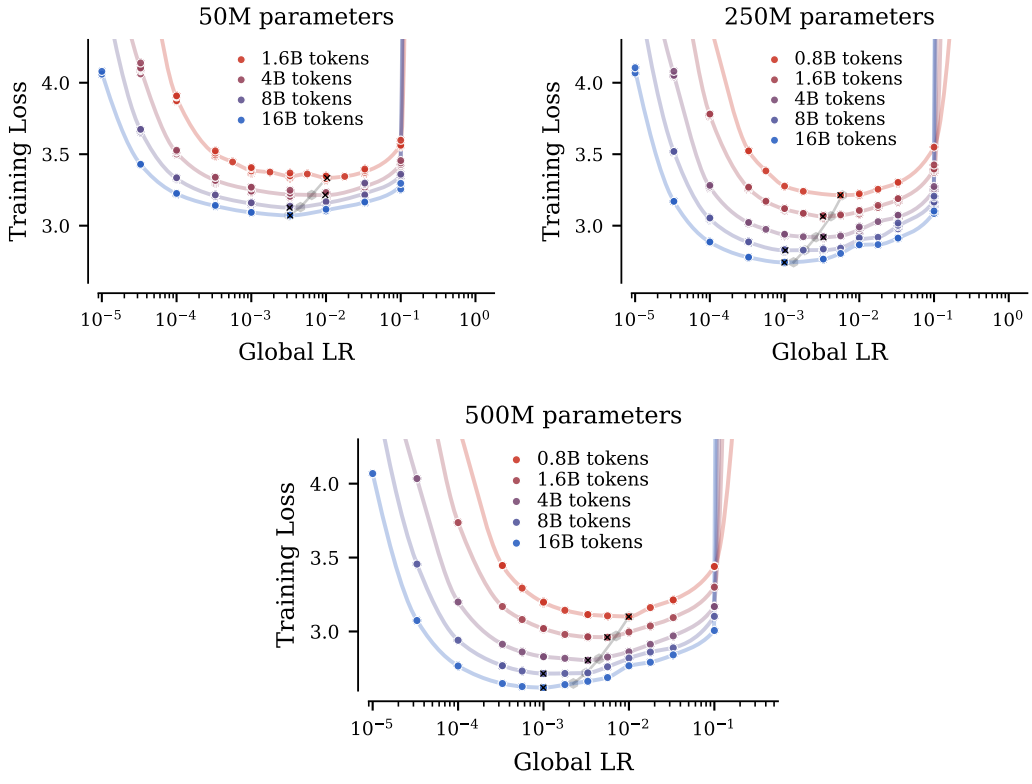
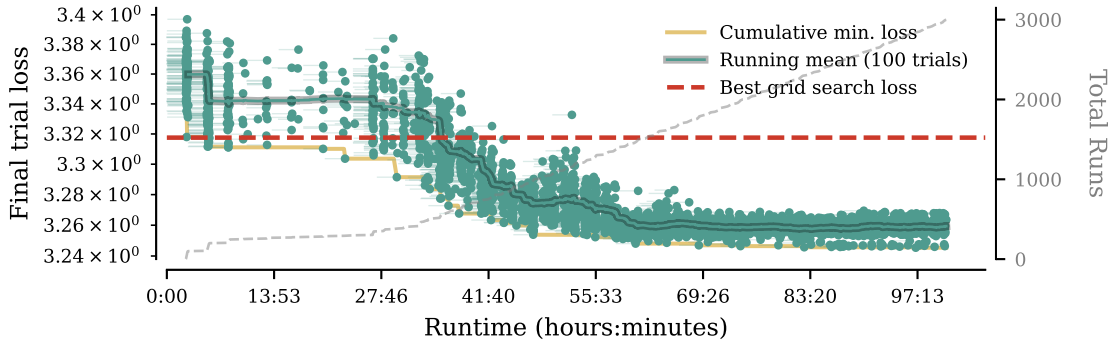
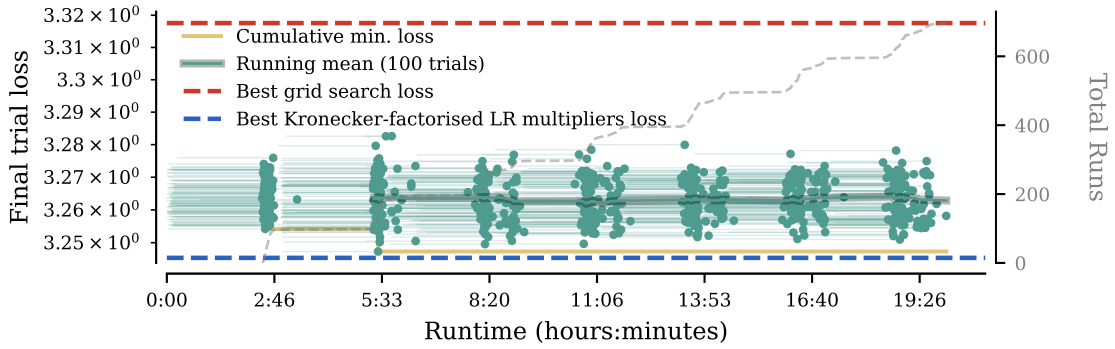


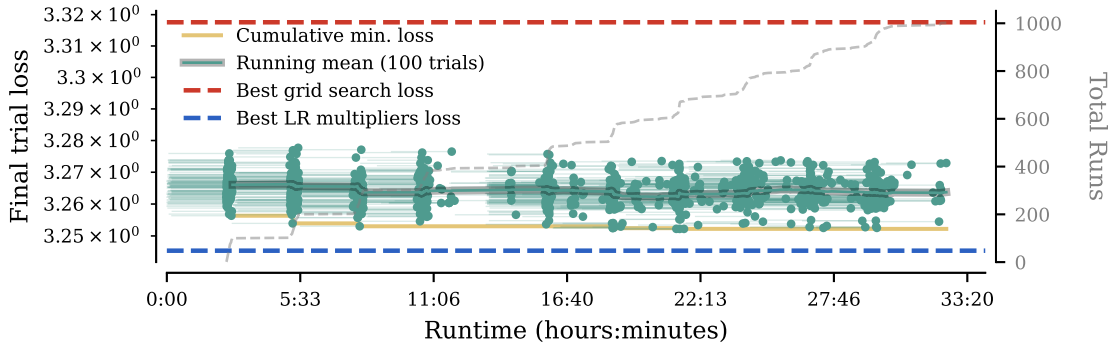
Figure 10: Lack of learning rate transfer across training horizons — increasing token horizon through number of iterations with a fixed batch-size — for different model sizes. The square-root transfer rule for the optimal learning rate identified at the smallest token horizon for each model size is plotted in .



(a) Hyperparameter search for the per-module learning rate multipliers parameterised with the depth-Kronecker factorisation.



(b) Hyperparameter search for the per-module learning rate multipliers with **fully uncoupled** multipliers. The search is initialised with the optimal HPs found in the search for optimal depth-Kronecker factorised multipliers in Figure 11a.



(c) Hyperparameter search for the per-module learning rate multipliers **with no depth multipliers**. he search is initialised with the projection of onto the constraint set of the optimal HPs found in the search in Figure 11a.

Figure 11: Hyperparameter search results with Trust Region Random Search. Each dot indicates the final loss of a single trial, and the lines indicate the training duration (start & end).

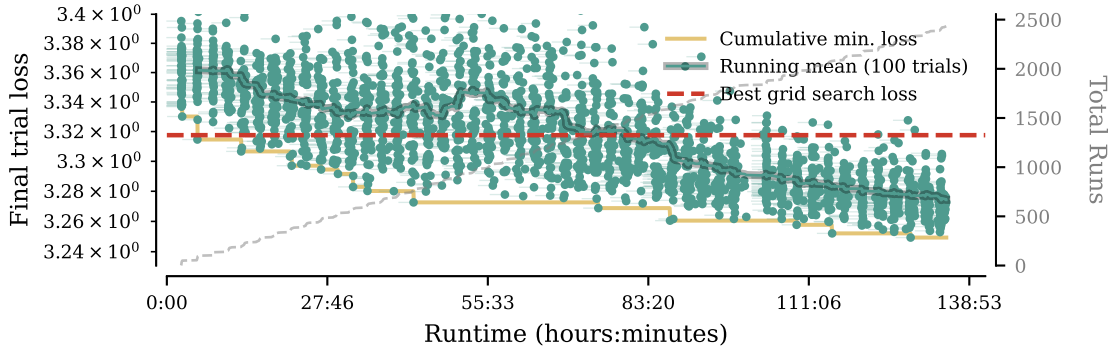


Figure 12: Hyperparameter search for the per-module learning rate multipliers parameterised with the Kronecker factorisation with CMA-ES. C.f. Figure 11a for the comparable Trust-region Random Search results.

B MOTIVATION OF THE COMPLETE$^{(D)}$P ADJUSTMENTS

Here, we give a justification for each of the modifications made in Complete$^{(d)}$P in Table 1. We directly rely on the properties that μP parameterised neural networks are known to possess that were formally shown in (Yang et al., 2024; Yang & Littwin, 2023)

These modifications primarily concern scaling the infinite-width limit.

B.1 QK NORM MULTIPLIER WEIGHTS

In standard implementation of QK norms, the elementwise affine operation $\mathbf{x} \mapsto \mathbf{m} \odot \mathbf{x} + \mathbf{b}$ with multipliers $\mathbf{m}$ and bias $\mathbf{b}$ is shared across the transformer heads. When scaling width by increasing the number of heads — as is common in most of the relevant model scale parameterisation literature (Dey et al., 2025; Yang et al., 2022) — this effectively means that these parameters are shared across the scaled width dimension N. For instance, for a collection of query vectors $\mathbf{q} \in \mathbb{R}^{N_{\text{heads}} \times d_{\text{head}}}$, we have that the normalised query elements $\hat{q}_{ij} := m_j q_{i,j} + b_j$ all share the same parameters $m_j, b_j \in \mathbb{R}$ for $i = 1, \ldots, N_{\text{heads}}$, where $N_{\text{heads}} = \Theta(N)$. We denote by $\mathbf{q}_{:,j}$ the $\mathbb{R}^{N_{\text{heads}}}$ vector $(q_{i,j} : i = 1, \ldots, N_{\text{heads}}$ ($\hat{\mathbf{q}}_{:,j}$ respectively). By the results of Yang & Hu (2021); Yang & Littwin (2023), we have that for the μP parameterisation $\mathbf{q}_{:,j}$ has entries of size $\Theta(1)$ throughout training. The loss gradients for any hidden (pre-)activation, are known to have entry size $\Theta(1/N)$, and so the post-normalised query activation gradients $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{q}}_{:,j}}$ will also be of size $\Theta(1/N)$. The backpropagated gradient with respect to the multiplier m_j is:

$$\sum_{i=1}^{N_{\text{heads}}} \left[\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{q}}_{:,j}}\right]_i \mathbf{q}_i = \frac{1}{N} \sum_{i=1}^{N_{\text{heads}}} N \left[\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{q}}_{:,j}}\right]_i \mathbf{q}_i,$$

where the rescaled random variables $N \left[\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{q}}_{:,j}}\right]_i \mathbf{q}_i$ have entry size $\Theta(1)$ as $N \to \infty$. Informally, in Yang & Hu (2021), the random variables $N \left[\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{q}}_{:,j}}\right]_i \mathbf{q}_i$ for $i = 1, \ldots, N_{\text{heads}}$ were shown to approach *i.i.d.* as $N \to \infty$. Hence the sum above has a Strong Law of Large Numbers like behaviour, converging to the mean of the entrywise limit of $N \left[\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{q}}_{:,j}}\right]_i \mathbf{q}_i$. As such, we effectively have that the gradient with respect to the width-shared parameters is also $\Theta(1)$ with width. The scale of the AdamW ϵ should match the scale of the gradient (Yang & Littwin, 2023), and so we have that the AdamW ϵ parameter for the width-shared multipliers should also be scaled as $\Theta(1)$ with width. A near-identical argument follows for the bias terms.

B.2 EMBEDDING LAYER ADAMW ϵ

The $\Theta(1/N)$ scaling with width N for the embedding layer AdamW ϵ follows from the observation that the gradients with respect to the embedding parameters have element size $\Theta(1/N)$. To see this, note that the gradients with respect to the output of the embedding layer are $\Theta(1/N)$, whereas inputs are obviously constant with width. Hence, it naturally follows that AdamW ϵ should be scaled as $\Theta(1/N)$ to match the scale of the gradient (Yang & Littwin, 2023).

B.3 CHANGES TO THE UNEMBEDDING WEIGHT

The changes to the unembedding weight scaling rules are mostly a reparameterisation of the multiplier-based μP implementation in (Dey et al., 2025). Namely, for AdamW, a weight multiplier m_N^γ has the same effect throughout training (bar the finite-precision arithmetic effects) as: **1)** multiplying the initialisation variance by $m_N^{2\gamma}$, **2)** multiplying the learning rate by m_N^γ, **3)** and multiplying the AdamW ϵ parameter by m_N (Yang & Littwin, 2023). We re-parameterise with **(1)** and **(2)**, but we don't change AdamW ϵ as it appears to have been derived incorrectly in (Dey et al., 2025). To see this, note that with μP (the Table 3 variant without an output layer multiplier), the gradients for the unembedding layer weights are expected to have scale $\Theta(1)$ with width N. Hence, to remain of the same scale, the output embedding weight ϵ should also have a matching scale of $\Theta(1)$ (Yang & Littwin, 2023). After reparameterisation to a m_N^{-1} output layer multiplier — as is done in CompleteP — the ϵ would also have to had to be scaled as m_N^{-1} to match the reparameterised gradients.

B.4 PER-MODULE HYPERPARAMETER SEARCH ALGORITHM

As described in Section 3.1, standard random search is unsuitable for the task of optimising *per-module* hyperparameters. We make two minimal tweaks that make it into a workable method. We induce an exploitation bias by turning it into a trust region method: we constrain the search-space adaptively to the neighbourhood $\{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{x}_t^{\texttt{opt}}\|_\infty \leq r\}$ of the current best solution $\mathbf{x}_t^{\texttt{opt}}$ at a given iteration t. Hence, the bounds move with the best solution found so far. We optimise all parameter in the $\log_2$-space, and sample uniformly from within the bounding box. Even with this modification, however, we found that this trust-region random search quickly plateaued with a relatively high variance in the final loss values. Hence, to allow the algorithm to explore promising regions more thoroughly, we also decay the size of the bounding region r if the loss doesn't improve after a certain number of trials.

For all experiments, unless stated otherwise, we instantiate the search with the bounding box size of 1 (meaning that at each iteration, we multiply the best solution found so far by 2^x with x sampled uniformly from $[-1, 1]$), and decay size of the trust region r by 0.7 if no improvement is observed in 100 trials. We run the algorithm asynchronously with a maximum of 100 simultaneous trials.

The goal of this paper is not to identify the *best* HP optimisation strategy for this setting; we merely want to find a workable one in order to demonstrate potential for improvements from per-module HP search. Since the above tweaks borrow from the principles underlying many evolutionary search (ES) methods, we also wanted to directly compare to a strong ES baseline to check our method performs reasonably. In Figure 12, we compare CMA Evolutionary Search (CMA-ES) (Hansen, 2006) to the Trust-region Random Search described above (c.f. Figure 11a). CMA-ES is not natively an asynchronous HP search strategy, so we make a minor modification: for a population size P, we wait until at least P trials sampled from the current generation have finished running. At that point, there might be more than P new finished trials (left-over trials from the previous generations), so we update the CMA-ES state with P *best* trials only. In this instance, Trust-region Random Search outperforms this CMA-ES variant. This gives credence to our search method of choice being able to identify good per-module HPs in reasonable runtime. We hope that future work can explore alternative strategies that might be able to severely reduce the number of trials required to find good per-module HPs.

C EXPERIMENTAL DETAILS

C.1 BEST LEARNING RATE (LR) ANNEALING AT DIFFERENT TOKEN HORIZONS.

We pretrain a small GPT-2 model (121M parameters). We enumerate all the non-increasing piecewise-constant LR schedule over the discrete set $\{0.0015/2.5^k | 0 \leq k \leq k_{max}\}$. We sub-divide the total training duration in L intervals of 77M tokens each. At the end of each interval, either the LR remains constant, either it is decayed by one or more steps. We chose $L = 16$ and $k_{max} = 4$, which yields a total of 4842 runs. For efficiency, we use the same checkpoint to warm start all runs sharing the same prefix in the LR scheduling, which cut down the computational complexity of this naive enumeration from $\mathcal{O}(L^{k_{max}+1})$ to $\mathcal{O}(L^{k_{max}})$. Therefore, the total compute budget is kept under 7,000 A100 GPUh. For five different token horizons (155M, 310M, 621M, 932M and 1.24B) we report the best scheduling among the 4,842 tested. We report the results in Figure 6. We notice that the best scheduling at short horizon is never a prefix of the best scheduling at long horizon. This empirical observation is compatible with the findings of Luo et al. (2025): there is a tension between the optimisation bias induced by the terminal LR value (the lower the better) and the progress of optimisation which requires higher LR at start.

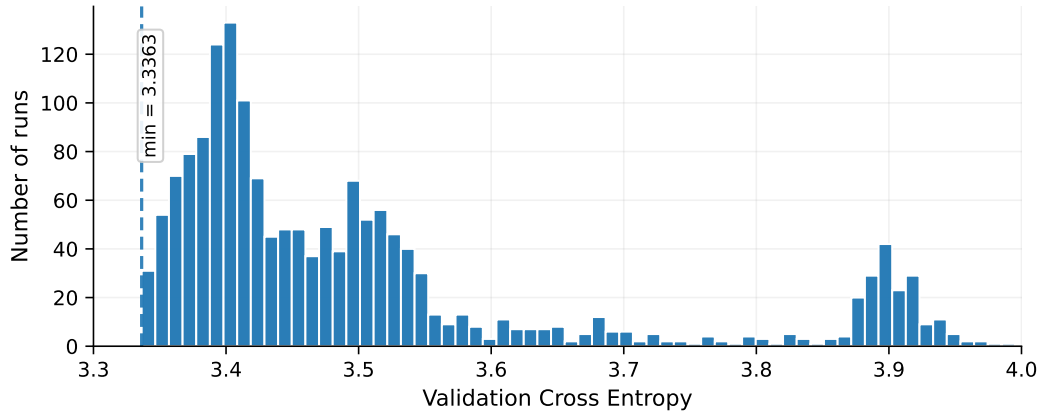
C.2 BASELINE GLOBAL HYPERPARAMETER TUNING

To establish a baseline, we perform an extensive random hyperparameter search consisting of 2048 trials. Each trial trains a 50M parameter model ($d_{\text{model}} = 512, L = 4$) for 1.64B tokens (33 tokens/parameter) over a discrete search space defined by:

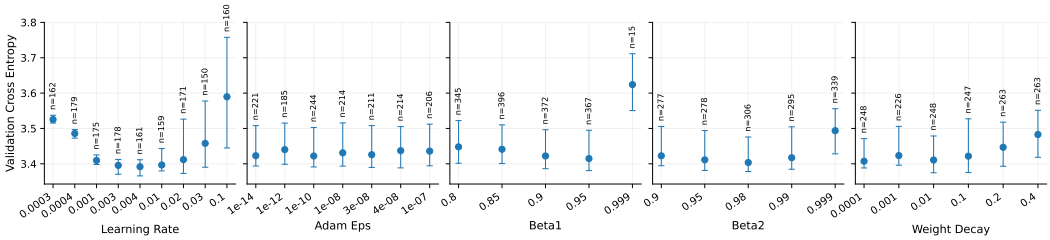
- LR $\in \{1\times10^{-4}, 3\times10^{-4}, 4\times10^{-4}, 1\times10^{-3}, 3\times10^{-3}, 4\times10^{-3}, 1\times10^{-2}, 2\times10^{-2}, 3\times10^{-2}, 1\times10^{-1}\}$

- Adam $\epsilon \in \{1 \times 10^{-14}, 1 \times 10^{-12}, 1 \times 10^{-10}, 1 \times 10^{-8}, 3 \times 10^{-8}, 4 \times 10^{-8}, 1 \times 10^{-7}\}$
- Adam $\beta_1 \in \{0.8, 0.85, 0.9, 0.95, 0.999\}$
- Adam $\beta_2 \in \{0.9, 0.95, 0.98, 0.99, 0.999\}$
- Weight Decay $\in \{1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}, 2 \times 10^{-1}, 4 \times 10^{-1}\}$

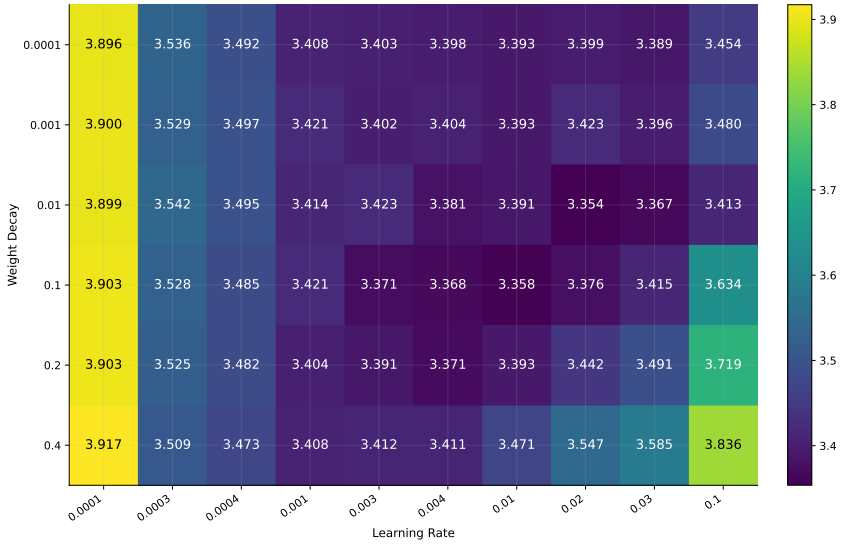
The results and hyperparameter sensitivities from this search are visualized in Figures 13a to 13c. The optimal configuration from this global search achieves a validation negative log-likelihood of 3.34 nats. This result is substantially higher than that achieved by our per-parameter search strategy, underscoring the advantage of discovering optimal configurations at a small scale before upscaling with principled rules like Complete$^{(d)}$P .



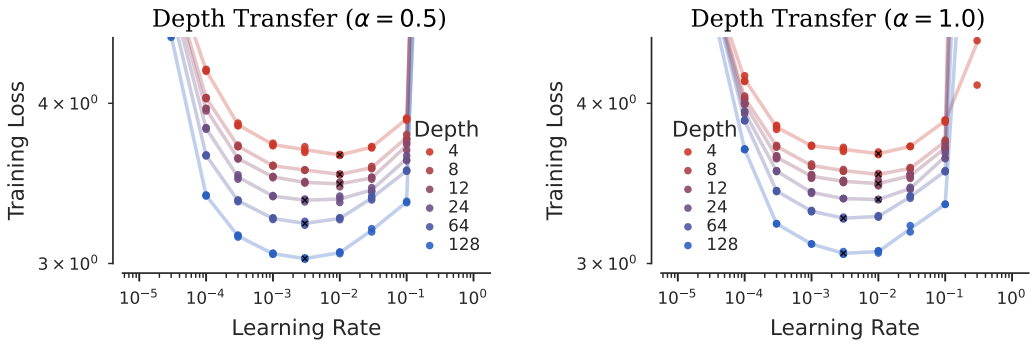
(a) Number of trials of baseline 50M model vs. evaluation loss.



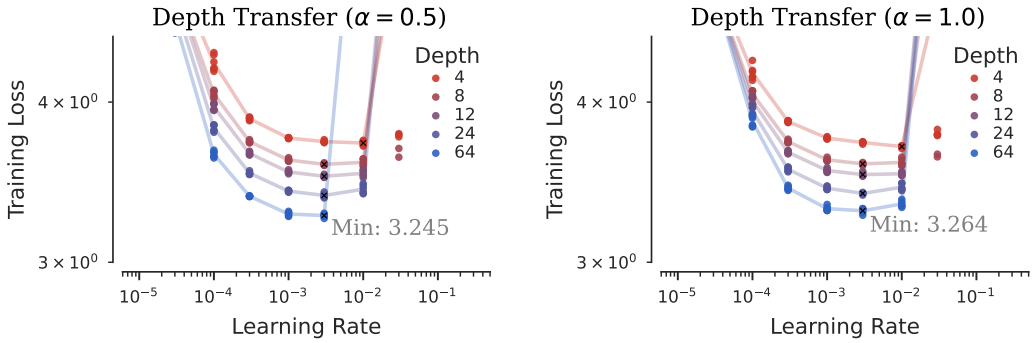
(b) Sensitivity of baseline 50M model for core hyperparameters.



(c) Weight decay vs. learning rate for baseline 50M model.



(a) **With** QK-norms



(b) **Without** QK-norms

Figure 14: The effect of QK-norms on hyperparameter transfer of the global learning rate across depth with two variants of Complete$^{(\mathbf{d})}$P with $\alpha \in \{\frac{1}{2}, 1\}$.