Small-scale proxies for large-scale Transformer training instabilities

Mitchell Wortsman Peter J. Liu Lechao Xiao Katie Everett
Alex Alemi Ben Adlam John D. Co-Reyes Izzeddin Gur Abhishek Kumar
Roman Novak Jeffrey Pennington Jascha Sohl-dickstein Kelvin Xu
Jaehoon Lee* Justin Gilmer* Simon Kornblith*

Google DeepMind

Abstract

Teams that have trained large Transformer-based models have reported training instabilities at large scale that did not appear when training with the same hyperparameters at smaller scales. Although the causes of such instabilities are of scientific interest, the amount of resources required to reproduce them has made investigation difficult. In this work, we seek ways to reproduce and study training stability and instability at smaller scales. First, we focus on two sources of training instability described in previous work: the growth of logits in attention layers (Dehghani et al., 2023) and divergence of the output logits from the log probabilities (Chowdhery et al., 2022). By measuring the relationship between learning rate and loss across scales, we show that these instabilities also appear in small models when training at high learning rates, and that mitigations previously employed at large scales are equally effective in this regime. This prompts us to investigate the extent to which other known optimizer and model interventions influence the sensitivity of the final loss to changes in the learning rate. To this end, we study methods such as warm-up, weight decay, and the μParam (Yang et al., 2022), and combine techniques to train small models that achieve similar losses across orders of magnitude of learning rate variation. Finally, to conclude our exploration we study two cases where instabilities can be predicted before they emerge by examining the scaling behavior of model activation and gradient norms.

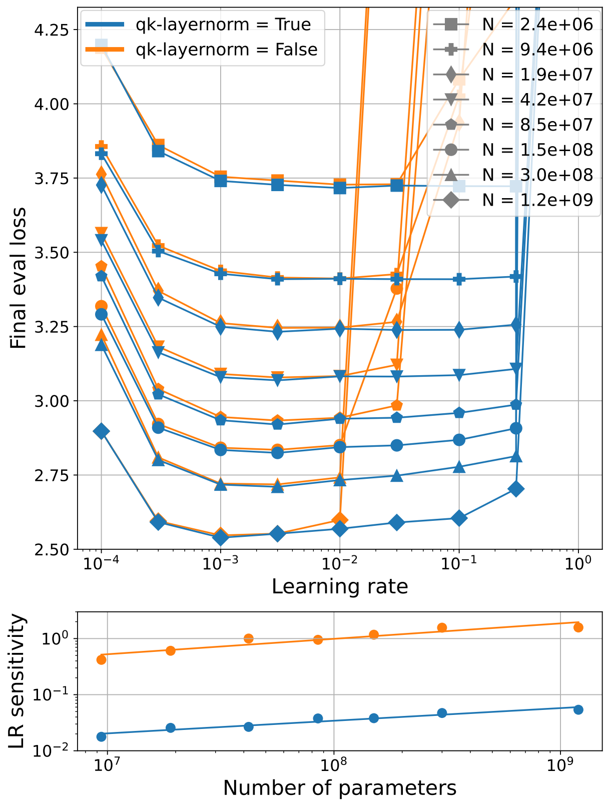

Figure 1: Qk-layernorm [11] enables stable training across three orders of magnitude of learning rate (LR) variation. **(Top)** For transformers with N parameters, we plot the effect of learning rate on final evaluation loss. **(Bottom)** We use LR sensitivity to summarize the top plot. LR sensitivity measures the expected deviation from optimal when varying learning rate across three orders of magnitude. Qk-layernorm reduces LR sensitivity, but LR sensitivity still increases with model scale.

1 Introduction

Scaling up transformers has led to remarkable progress from chat models to image generation. However, not every training run is successful. When training large Transformers, researchers have reported instabilities which slow or destabilize learning [6, 11, 52, 35, 8]. As the resources required for large runs continue to grow, it is important to examine the ways that Transformer training can fail.

In this report we reproduce, study, and predict training instability in Transformer models. We find that measuring the relationship between learning rate and loss across scales is a useful tool to identify instability (e.g., Figure 1). Therefore, we introduce learning rate (LR) sensitivity, which serves as a useful summary statistic for learning rate vs. loss curves. LR sensitivity measures the deviation from optimal performance when varying LR across orders of magnitude.

We show that two sources of instability, which have previously been described at scale, can be reproduced in small Transformers.[1] This enables their study without access to large resource pools. In particular, we examine the growth of logits in attention layers [11, 16, 50] and divergence of the output logits from the log probabilities [6]. As evident from the learning rate vs. loss curves and by inspecting model characteristics, both instabilities appear at high learning rates in small models. Moreover, interventions which have previously been employed at scale are also successful in this regime (e.g., Figure 1). These interventions—qk-layernorm [11][2] and z-loss regularization [6]—reduce LR sensitivity and enable successful training across three orders of magnitude of LR variation.

These observations raise the question of how other known optimizer and model interventions affect the shape of the learning rate vs. loss curves across scales. Therefore, we study the effect of techniques such as warm-up, weight decay, and μParam [49] in this context. When employing qk-layernorm and z-loss regularization, these other techniques usually have little impact on the range of learning rates at which models can be stably trained, but do affect the sensitivity to learning rate within this range. In line with previous work, we find that longer warm-up reduces learning rate sensitivity, as does the independent scaling of learning rate and weight decay recommended by Loshchilov and Hutter [33]. One interesting finding is that scaling depth increases LR sensitivity at a faster rate than scaling width.

The remainder of our investigation centers on the scaling behavior for model characteristics such as activation and gradient norms. Using the attention logit growth instability as an example, we show that it is possible to predict an instability before it emerges. This is in contrast to prior works on scaling which primarily focus on scaling trends related to loss [27, 22].

We conclude by using the scaling behavior of model characteristics to search for instabilities that are currently not well documented. Our investigation shows that gradient norms decrease with both scale and learning rate, such that the default AdamW [33] epsilon hyperparameter is too large. This causes updates that are too small. We connect this phenomenon and the attention logit growth instability to parameter norm growth [34, 29].

Overall, we believe our work presents new scientific opportunities for studying training stability without access to large resource pools.

2 Experimental methodology

This section details our experimental set-up (Section 2.1) and useful tools employed by our analysis: (i) measuring the relationship between learning rate and loss across scales (Section 2.2) and (ii) examining scaling trends for model characteristics (Section 2.3).

2.1 Experimental set-up

We train small Transformer models [44] with a similar experimental set-up as GPT-2 [38] implemented in Flax [20]: the models are decoder-only [31] and trained with an auto-regressive loss (refer to Section A for more infrastructure details). While we experimentally manipulate many of the following hyperparameters, this section provides their default values, which we use unless otherwise specified.

By default, we use AdamW [33] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon =$ 1e-8, and gradient clipping at global norm 1. The default warmup is 5e3 steps, and the default number of total steps is 1e5. We use a linear schedule for warmup and and a cosine-decay [32] schedule for the remainder, with minimum learning rate 1e-5. We use an independent weight decay of 1e-4 and auxiliary z-loss [6] with coefficient 1e-4. Sections 3.2.2 and 3.1.2 respectively provide additional information and ablations on decoupled weight decay and z-loss.

[1]We focus on instabilities which lead to slow divergence, not loss spikes (see Section 4).

[2]Based off currently unpublished investigations of Gilmer et al. [16].

We use pre-normalization [38] Transformers with qk-layernorm [11] (see Section 3.1.1 for information). We do not use any biases following Chowdhery et al. [6], and the layernorm [1] ϵ remains at the default value in Flax [20] of 1e-6. We jointly scale up the embedding size, depth, and number of heads when scaling parameters. We do not use weight tying of the first and last layer [37], and when reporting the number of parameters we exclude the embedding and head (as in Kaplan et al. [27]). We use rotary positional embeddings [43], and for training data we use C4 [39]. Letting d refer to the model dimension (i.e., the embedding size), the feed-forward component of the Transformer is an MLP with hidden dimension of $4d$ and gelu [21] activations. As in Vaswani et al. [44] we use factor $1/\sqrt{d}$ scaling in the self-attention. The embedding initialization is the default in Flax, which is normally distributed with standard deviation $1/\sqrt{d}$. The remainder of the weights are initialized with a truncated normal distribution with inverse root fan-in standard deviation [18]. The default batch size is 256, where each batch element has a sequence length of 512 tokens. Sequences are packed so that no padding is required. Finally, we use the vocabulary from Raffel et al. [40] which has size 32101 and uses a SentencePiece [28] tokenizer. We train on TPUs [26] in bfloat16 precision using Flax [20] and JAX [4].

2.2 LR vs. loss curves and learning rate sensitivity

To investigate how model instability emerges with scale, it is useful to plot the relationship between learning rate (LR) and loss for models of different sizes. For instance, an instability is often characterized by an explosion in the loss at high learning rates. LR vs. loss curves can reveal how the lowest unstable learning rate changes as a function of model size.

To summarize LR vs. loss curves, we use LR sensitivity. LR sensitivity measures the deviation in final validation loss from optimal when sweeping LR across three orders of magnitude. If a model fails to train at high learning rates, then LR sensitivity will be high. There are cases where LR vs. loss curves and LR sensitivity are no longer meaningful, for instance if an intervention changes the meaning of learning rate—see Appendix B for a detailed discussion.

Let $\theta = \mathcal{A}(\eta)$ denote the model weights θ obtained when training with learning rate η, and let $\ell(\theta)$ denote the validation loss when using weights θ. For a learning rate range $[a, b]$, let ℓ^* denote the loss obtained with the best learning rate, i.e., $\ell^* = \min_{\eta \in [a,b]} \ell(\mathcal{A}(\eta))$. Then, LR sensitivity is defined as LR sensitivity $= \mathbb{E}_{\eta \in [a,b]} \left[\ell\left(\mathcal{A}\left(\eta\right)\right) - \ell^* \right]$.

Unless otherwise mentioned, we use the learning rate range 3e-4 to 3e-1 with AdamW [33] to measure LR sensitivity, where LR refers to the maximum value in a cosine decay schedule with warm-up [32]. We consider LRs in {3e-4, 1e-3, 3e-3, 1e-2, 3e-2, 1e-1, 3e-1} when computing the minimum and expectation.

2.3 Scaling trends for model characteristics

To study instability, we also find it useful to examine scaling trends for model characteristics such as gradient or activation norms. This method is helpful for predicting instabilities and contrasts with previous work on scaling, which primarily focuses on trends relating model scale and loss [27, 22].

3 Results

This section presents our results on training stability for small Transformers. Equipped with LR sensitivity (Section 2.2), we study two known instabilities and their corresponding mitigation at small scale (Section 3.1). This raises the question of how other model and optimizer interventions effect sensitivity of final loss to learning rate, which we investigate in Section 3.2. Finally, we examine whether instabilities can be reliably predicted before they emerge: Section 3.3 predicts when the logit growth instability may cause divergence in a larger model, while Section 3.4 aims to find other issues that may occur when scaling up with our default hyperparameters.

3.1 Reproducing two known instabilities at small scale

Here, we examine two instabilities that have previously been described at scale: the growth of logits in attention layers [11, 16, 50] and divergence of the output logits from the log probabilities [6]. By examining LR vs. loss curves, we show that these instabilities can be reproduced in small models by using high learning rates and that mitigations employed at scale are effective in this regime.

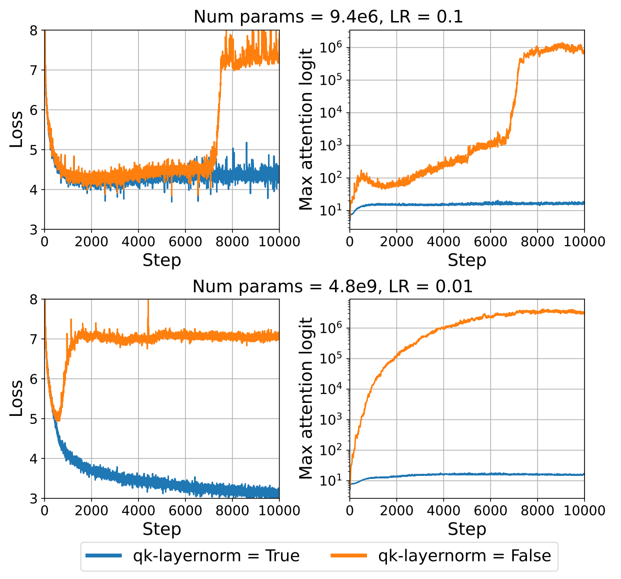

Figure 2: The attention logit growth instability [11, 50] appears in small models at high learning rates. The mitigation of applying qk-layernorm proposed by Dehghani et al. [11] is equally effective in the small-scale regime. The max attention logit is reported for layer 0, which we typically observe to have the largest logit values.

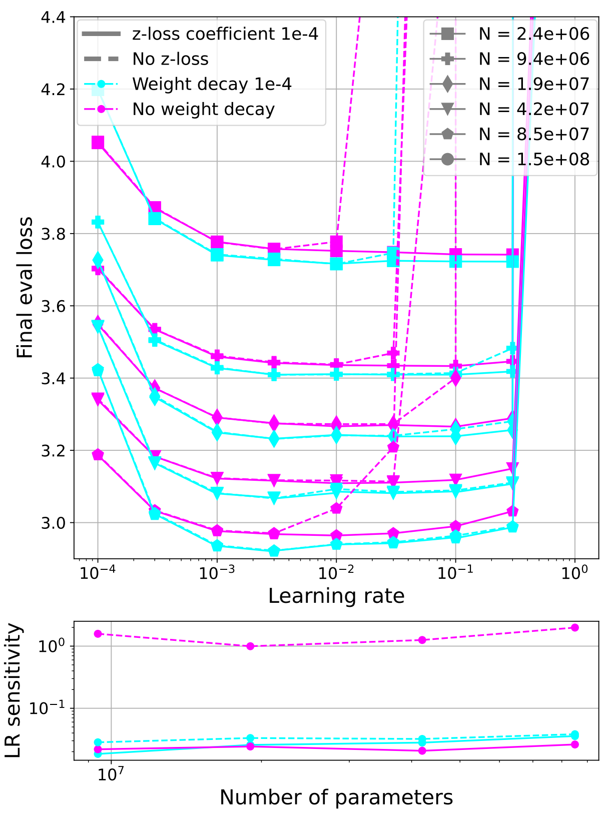

Figure 3: The effect of the output logit divergence instability [6] and the z-loss mitigation [6] (Section 3.1.2). Models in this experiment have qk-layernorm [11].

3.1.1 Attention logit growth

Researchers have previously documented that Transformer training fails when the attention logits become large [11, 50]. In Dehghani et al. [11], this issue emerged when training a ViT model [14] with 22 billion parameters.

In the self-attention layer of a Transformer [44], queries q_i and keys k_i are combined to compute the attention logits $z_{ij} = \langle q_i, k_j \rangle / \sqrt{d_h}$, where d_h is the head dimension. Next, the attention logits are passed through a softmax to produce attention weights, which are used to combine values v_i. Dehghani et al. [11] observed that the attention logits z became large, which they refered to as attention logit growth. As a result, the attention weights collapse to one-hot vectors, which was named attention entropy collapse by Zhai et al. [50]. To resolve this issue, Dehghani et al. [11] proposed qk-layernorm, which applies LayerNorm [1] to the queries and keys before computing the attention logits.

In our experiments, we find that models need not be large to exhibit instability related to attention logit growth. As shown in Figure 1, the maximum learning rate at which small models can be trained increases when using qk-layernorm. Without qk-layernorm, the learning rate at which models diverge becomes smaller with increasing model size. By contrast, models with qk-layernorm exhibit considerably lower LR sensitivity and train to low loss at high learning rates. As a highlight, qk-layernorm allows training a model with 1.2B parameters at learning rate 0.3. Both with and without qk-layernorm, LR sensitivity increases with scale.

Figure 2 displays the loss and max attention logit for two model scales that differ by three orders of magnitude. In both cases, the loss diverges without qk-layernorm. Our results in Appendix Figure E.1 suggest that attention logit growth is due to growth in the queries and keys, not due to an increase in their alignment. Finally, Appendix C connects this instability to the quadratic dependence of attention logits on parameter norms.

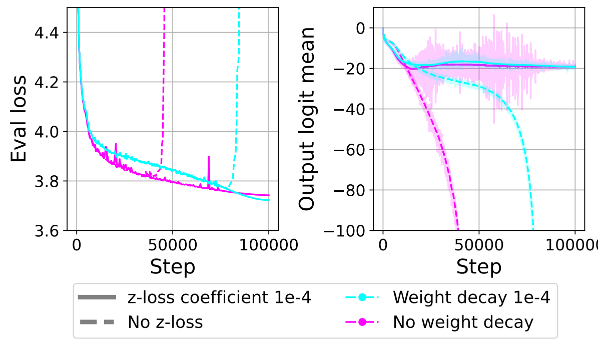

Figure 4: An example of the output logit divergence instability [6] (Section 3.1.2) in a 2.4M parameter Transformer at learning rate 0.1.

3.1.2 Output logit divergence

Another instability reported by researchers training large models is divergence in the output logits from the log probabilities [6]. Just as before, we reproduce this instability with small models at large learning rates, and the proposed mitigation ameliorates the issue. Overall, Figure 3 summarizes the effect.

Let y denote the model's output logits, which are used to compute class probabilities p_i via a softmax $p_i = e^{y_i}/Z$ where $Z = \sum_j e^{y_j}$. This instability occurs when the logits diverge and become very negative, as illustrated in Figure 4 for a 2.4M parameter model at learning rate 0.1. In contrast to the attention logit growth instability, this divergence occurs towards the end of training. The mitigation proposed by Chowdhery et al. [6] is to encourage $\log Z$ to remain close to zero. They add an auxiliary loss $\log^2 Z$, referred to as z-loss, with coefficient 1e-4.

As illustrated in Figures 3 and 4, we find that instability related to output logit divergence occurs in models with no weight decay regardless of scale, and z-loss resolves this instability. Weight decay also mitigates this instability for the larger models we test.

3.2 Measuring the effect of other known interventions

The previous section used the relationship between learning rate and loss as a useful tool for examining two known instabilities and their mitigation. This raises the question of how other known model and optimizer interventions affect the shape of LR vs. loss curves across scales. In particular, can LR sensitivity help identify additional issues or resolutions when scaling? This section aims to answer this question for common techniques such as warm-up, weight decay, and μParam [49].

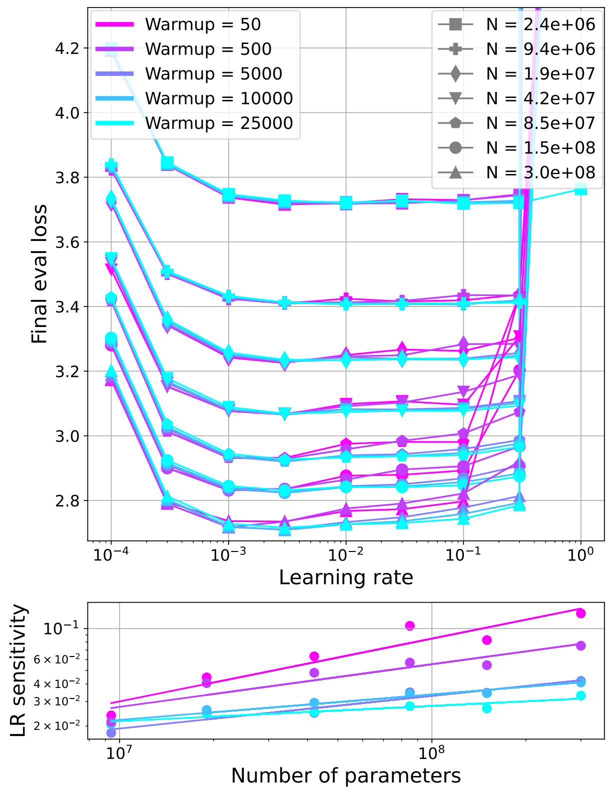

Figure 5: The effect of warm-up length for different model sizes. Longer warm-up reduces LR sensitivity and loss, especially for the larger models we test. Models in this experiment use qk-layernorm [11].

3.2.1 Warm-up

As illustrated by Figure 5, a longer warm-up period reduces LR sensitivity. This is most clear for the larger models, which are not stable at LR 3e-1 without long warm-up. The number of total steps is fixed to 1e5 in this experiment, and all models use qk-layernorm. The importance of warm-up for stability has previously been highlighted [17, 42, 30], although these works do not measure scaling behavior.

3.2.2 Independent weight decay

Parameterizing weight decay independently of learning rate reduces LR sensitivity, as illustrated in Figure 6. While this was recommended by Loshchilov and Hutter [33], it is not common practice in the

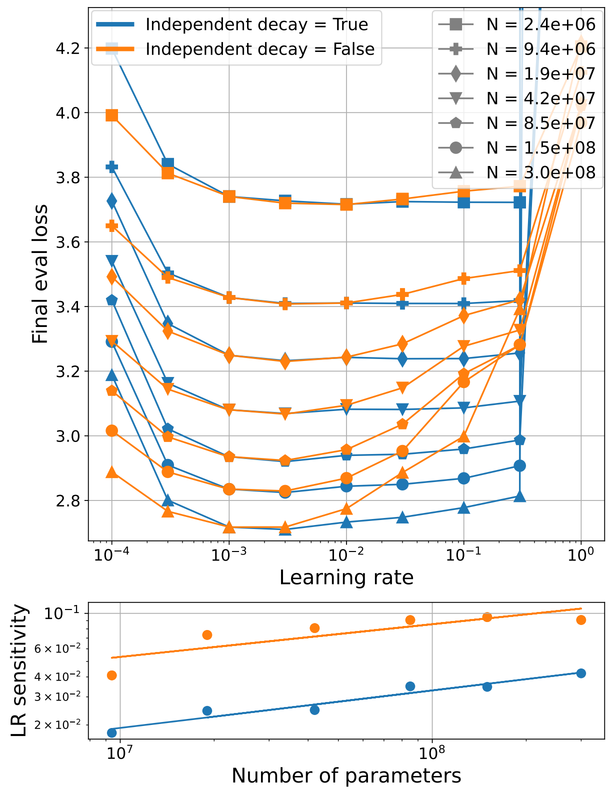

Figure 6: Independently scaling LR without also scaling weight decay reduces LR sensitivity. While this was recommended by Loshchilov and Hutter [33], it is not common practice in the default AdamW implementations in popular libraries. Refer to Section 3.2.2 for more information. Models in this experiment use qk-layernorm [11].

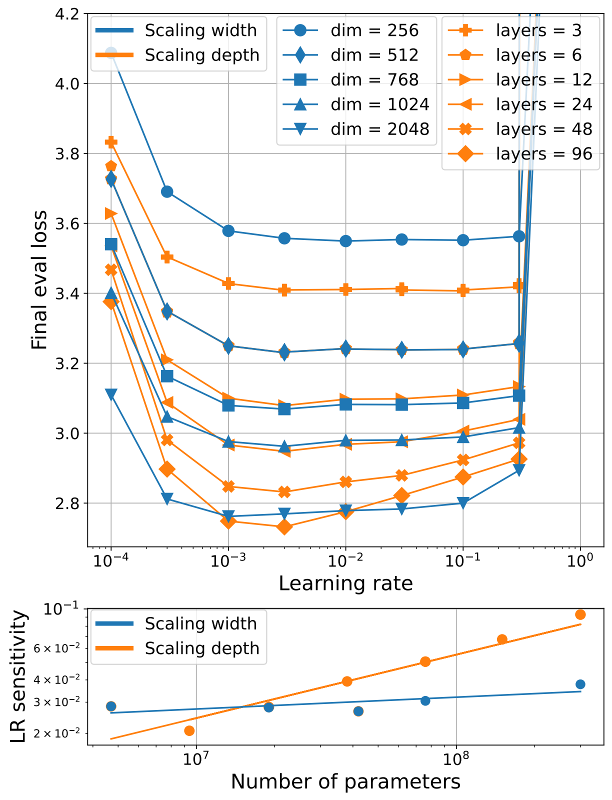

Figure 7: Independently scaling depth increases LR sensitivity at a faster rate than scaling width, though also produces a model with lower loss at the largest scale we test. Refer to Appendix Figure E.2 for this experiment without qk-layernorm.

default AdamW implementations of PyTorch [36] or Optax [2]. We explain the differences below.

For parameters θ, let $\Delta = v/(\sqrt{u}+\epsilon)$ denote the AdamW update without learning rate or weight decay. For weight decay coefficient λ, max learning rate η, and schedule $s_t \in [0,1]$, Loshchilov and Hutter [33] recommend the update $\theta \leftarrow \theta - s_t(\eta\Delta - \lambda\theta)$, which we refer to as *independent decay*. On the other hand, the default implementation in PyTorch or Optax applies the update $\theta \leftarrow \theta - s_t\eta(\Delta - \lambda\theta)$, i.e., η now scales both terms.

When reporting LR sensitivity without independent decay in Figure 6, we report the minimum LR sensitivity over ranges [1e-4, 1e-1] and [3e-4, 3e-1] because the former is sometimes better centered on the minimum. The default setting in this paper is to use independent decay. When using independent decay we set λ=1e-4, and without independent decay we set λ=0.1. A sweep on weight decay values is conducted in Figure E.10.

3.2.3 Scaling width vs. depth

We have so far consistently observed that increasing the number of parameters increases LR sensitivity. We now examine which part of scaling is most responsible.

Our results, illustrated by Figure 7, indicate that scaling depth increases LR sensitivity at a faster rate than scaling width. However, at the largest scale we test, independently scaling depth produces a model with lower validation loss. A validation loss comparison between width scaling, depth scaling, and joint scaling is in Appendix Figure E.3. The standard practice of joint scaling performs best at the largest scale and also has a more reliable scaling prediction when extrapolating.

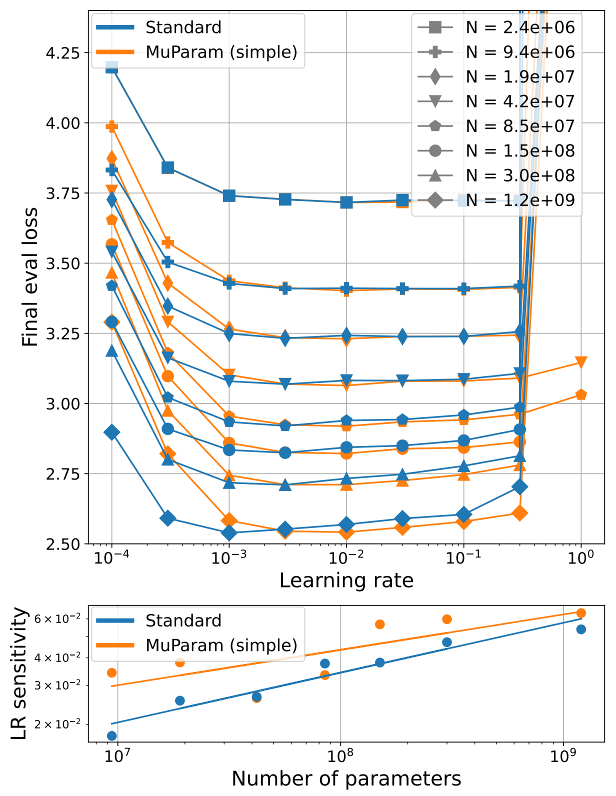

Figure 8: Measuring the effect of μParam on LR sensitivity for models with qk-layernorm [11]. In our setting μParam succeeds in stabilizing the optimal LR, though it does not improve loss or reduce LR sensitivity. For more information refer to Section 3.2.4.

When scaling depth, we use $d = 512$, and when scaling width, we use 6 layers. The number of heads is scaled proportionally with width, so that the head dimension remains the same.

Figure E.2 repeats this experiment without qk-layernorm, finding that the attention logit growth instability occurs more frequently at scale regardless of whether width or depth are scaled.

3.2.4 μParam

Yang and Hu [48] introduced the μParam method for parameterizing a neural network. As a product, the optimal LR remains consistent when scaling model width [49]. This section tests the effect of μParam on LR sensitivity, and examines whether μParam alleviates the need for qk-layernorm [11].

As illustrated by Figure 8, μParam does succeed in stabilizing the optimal LR at the scale we test. However, μParam does not improve loss or reduce LR sensitivity in our experiments. Appendix Figure E.4 repeats this experiment without qk-layernorm. Our results indicate that μParam does not alleviate the need for this intervention at high learning rates. We note that from a practical perspective, reducing LR sensitivity is not important if the optimal LR does not change.

We refer to the variant of μParam that we use in these experiments as μParam (simple) because it maintains only the core feature of μParam. We add additional features from Yang et al. [49] in Appendix Figure E.5 without measurable improvement at the largest scale we test. For μParam (simple) we make the following changes from our standard baseline: scale the LR for linear layers by base-fan-in/fan-in. For μParam (full) there are three additional changes: (i) initialize the head with standard deviation $\sqrt{\text{base-fan-in}/\text{fan-in}}$; (ii) change the $1/\sqrt{d_h}$ scaling factor in attention layers to $1/d_h$ where d_h is the head dimension; and (iii) initialize the query projection weights with zeros. For base-fan-in we use the fan-in values for the smallest model we test, which has width 256.

We comment briefly on the aforementioned changes (ii) and (iii). First, we ablate on change (ii) in isolation in Appendix Figure E.6. While this intervention reduces loss slightly at the smallest scale we test, the reverse is true for the largest scale we test. Also, removing the square root from the scaling factor in attention layers does not alleviate the need for qk-layernorm. Finally, with regards to change (iii), we note that in preliminary experiments this change had no noticeable effect.

3.2.5 Additional interventions

This section recreates the previous plots with additional interventions or hyperparameter changes. Corresponding figures are displayed in the appendix.

- Changing the number of training steps from 1e5 to 5e4 or 2e5 does not meaningfully change LR sensitivity (Appendix Figure E.7).
- We try applying qk-layernorm across the whole model dimension instead of individually per-head with shared paramters. As illustrated in Appendix Figure E.8, the latter performs better. We use per-head qk-layernorm as the default in all other experiments.
- Increasing the batch size from 256 to 512 or 1024

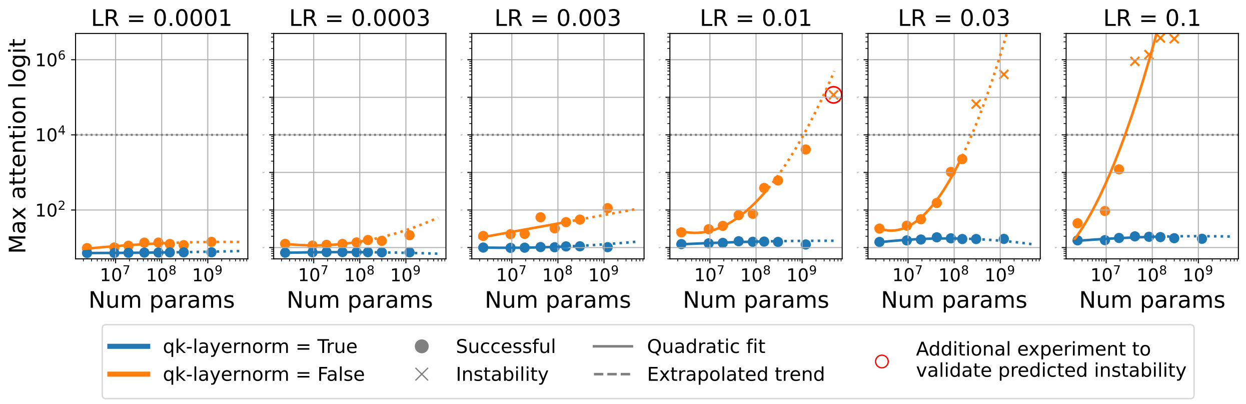

Figure 9: Predicting the attention logit growth instability via scaling behavior of model characteristics. We extrapolate to predict that a larger model will become unstable at LR 1e-2, and run an experiment to confirm the prediction. Refer to Section 3.3 for more information.

does not meaningfully change LR sensitivity (Appendix Figure E.9, each batch element contains 512 tokens). When increasing batch size we decrease the number of training steps so that the amount of data seen is constant. We believe a similar effect would be observed if instead we held the number of steps constant because changing the number of steps has no impact on LR sensitivity at batch size 256 (Appendix Figure E.7).

- The effect of changing the weight decay from 1e-4 is illustrated in Figure E.10. Increasing decay appears to slightly shift the optimal LR right.

- We find that the logit growth instability is not due to the softmax in the self-attention layer, as it still occurs with a pointwise variant of attention (Appendix Figure E.11).

3.3 Predicting attention logit growth instability from scaling behavior of model characteristics

A central question when studying instabilities is whether they can be predicted. We now examine whether it is possible to predict the logit growth instability before it occurs. We track the attention logit maximums across model scales and fit a curve to the data. We use this to predict that a 4.8B parameter model will be unstable at LR 1e-2 without qk-layernorm and run an experiment to confirm this prediction.

Figure 9 plots the number of parameters vs. max attention logit at different learning rate values.[3] At each learning rate, we fit a quadratic to predict how the max attention logit will change with model scale.

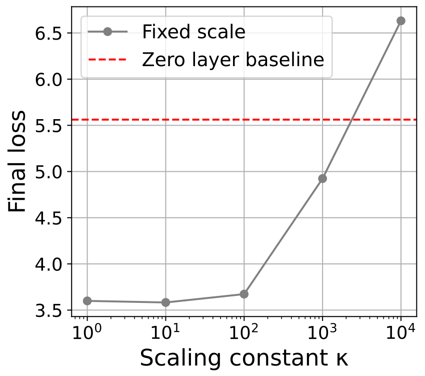

Figure 10: Enforcing a max attention logit of approximately κ in a small model to determine which value of κ inhibits learning.

We first noticed that all points with attention logits above 1e4 diverged. Moreover, the quadratic fit predicted that for LR 1e-2 the next model scale would also cross that value. Based on this prediction, we trained a new 4.8B parameter model at LR 1e-2. This model diverged as predicted. Not only do we predict the divergence, but our fit closely extrapolates to predict the value of the max attention logit.

[3]We use block 0, which typically has the largest logits, and consider the value at step 2e3. Much earlier than 2e3 was uninformative, and much later the unstable points had long past diverged.

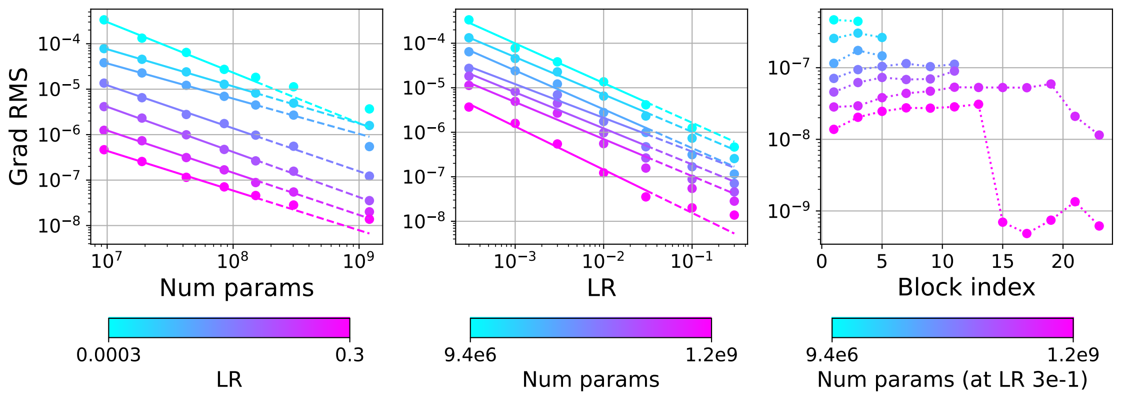

Figure 11: Predicting a potential instability from the scaling behavior of model characteristics. The gradient root mean square (RMS) decreases with num params (left) and learning rate (middle). These trends indicate that hyperparameter adjustment may be required to successfully scale further, as the RMS is approaching the default AdamW ϵ hyperparameter. If the gradient RMS becomes too small without adjusting ϵ or weight decay, a layer may collapse. The gradient RMS in the left and middle plot is reported for the first MLP layer of block 0, but we observe similar trends for other layers (e.g., Appendix Figure E.12). Gradient RMS across different blocks is also reported (right). Gradient and update RMS are averaged over the final 500 steps, refer to Appendix Figure E.13 for the data during training.

One question unresolved by our analysis so far is whether we could have predicted that instability arises when the max attention logit exceeds 1e4 without manipulating learning rate and model size. We take initial steps towards an answer by transplanting different values of max attention logit into a small network with 10M parameters. For different constants κ we pass the queries and keys through $g(z) = \sqrt{\kappa} \cdot z / \sqrt{\mathbb{E}_i[z_i^2]}$ before computing the attention logits. Results are illustrated in Figure 10. Loss deteriorates around κ =1e3, and by κ =1e4 the loss exceeds that of a zero-layer bigram model consisting of the Transformer we use without any self-attention or MLP layers.

3.4 Searching for new instabilities via scaling trends of model characteristics

This section examines whether the scaling behavior of model characteristics can be used to predict new issues with the default model and hyperparameter settings.

In Figure 11 we examine scaling trends for the gradient root mean square $\text{RMS}(g) = \sqrt{\mathbb{E}_i\left[g_i^2\right]}$. This figure reports the RMS for the first layer of the MLP, though we observe similar trends for other layers (Appendix Figure E.12).

As models get larger, the value that grad RMS approaches is cause for concern. At the largest scale and learning rate we test, grad RMS is around the default AdamW ϵ hyperparameter. Recall that the unscaled AdamW update is $\Delta = v / \left(\sqrt{u} + \epsilon\right)$, where v and u are the first and second gradient moment EMA, respectively. If the grad RMS is on the same order as ϵ, then Δ will decrease in magnitude as illustrated by Figure 13, and parameters will not receive learning signals as intended.

An obvious mitigation for this issue is to simply lower the AdamW ϵ hyperparameter from its default of 1e-8. We conduct this experiment for a 4.8B parameter model at LR 0.3 and present the results in Figure 12. Decreasing ϵ to 1e-15 improves loss and mitigates a collapse in grad RMS. We believe this improvement will only increase at scale. On the other hand, increasing ϵ to 1e-6 results in an instability (shown in Figure E.15).

Figure 13 expands on this result by illustrating the grad and update RMS throughout training at the largest scale and learning rate we test. When the grad RMS reaches ϵ, the update RMS becomes small. Figure E.13 presents data from an analogous experiment at many different scales and LRs, demonstrating that this issue is most apparent for the larger models and LRs we test.

Although we identified the instability above by empir-

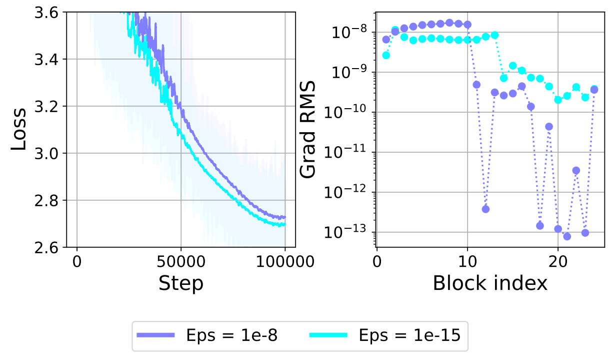

Figure 12: Decreasing the AdamW ϵ from its default value of 1e-8 to 1e-15 improves loss for a 4.8B parameter model at LR 0.3. When increasing ϵ to 1e-6, loss diverged. Grad RMS is averaged over the final 500 steps for the first layer in the MLP; refer to Figure 13 for data throughout training.

ically measuring the scaling behavior of the gradients, a mechanistic explanation exists. As learning rate increases, so does the parameter RMS. A larger parameter RMS leads to a larger RMS for the features output by each Transformer block. Then, the overall output RMS in turn increases with depth due to residual connections. The overall effect is that for larger networks and learning rates, the Transformer output RMS entering the final layernorm will grow. Since the layernorm gradients are scaled by the inverse of their input RMS, the gradient received by the Transformer will shrink. Refer to Appendix C for a more detailed discussion.

4 Related work

This paper mainly focuses on the effect of known interventions and instabilities, and so related work has been primarily discussed when relevant. This includes the attention growth instability observed by Dehghani et al. [11], Zhai et al. [50], and the final logit divergence issue encountered by Chowdhery et al. [6]. However, we highlight similar experimental methods in previous work. For instance, Yang et al. [49] also measure the relationship between LR and loss across scales, but their focus is on centering the optimum (see Section 3.2.4). In addition, Zhai et al. [50] elicit instability in base models by doubling learning rate, and Dettmers et al. [12] measure the presence of outlier features as a function of scale.

There are also important instabilities and related topics we have not directly discussed so far. For instance, we have primarily focused on instabilities that lead to a slow divergence, and we now summarize research on *fast loss spikes.* This instability is characterized by a quick increase in the loss that often eventually recovers.

The Edge of Stability and fast spikes

The conventional understanding of gradient descent predicts that loss instability only occurs when the learning rate exceeds $2/\lambda_{\max}(H)$, where H is the Hessian. However recent investigations into large batch neural network training dynamics have revealed a more complicated picture via edge of stability (EoS) [7]. When training neural networks with large batch SGD, the loss curvature constantly evolves via the interaction of two processes: progressive sharpening and self stabilization. Progressive sharpening is the empirical observation that when LR $< 2/\lambda_{\max}(H)$, the curvature gradually increases until the stability threshold is violated. When the learning rate becomes too large relative to the curvature, *fast loss spikes* occur and the parameters oscillate into a region with smaller $\lambda_{\max}(H)$ where stable training and progressive sharpening resumes. The latter process where instability results in smaller $\lambda_{\max}(H)$ is self-stabilization, a theoretical model of which is given in Damian et al. [9]. Gradually shrinking $\lambda_{\max}(H)$ via self stabilization was shown to be a primary mechanism behind the success of learning rate warmup in Gilmer et al. [17], who closely studied the connections between curvature, initialization, architecture and max trainable learning rates.

Cohen et al. [8] further analyze edge of stability of dynamics with adaptive optimizers, showing that progressive sharpening interacts with both the self-stabilization process and the adaptive optimizer state. This interaction results in the preconditioned sharpness $\lambda_{\max}(P^{-1}H)$ oscillating around an optimizer specific threshold (38/LR in the case of Adam with β_1=0.9). Adaptive EoS (AEoS) can also result in periodic loss spikes when progressive sharpening pushes the preconditioned sharpness above the stability threshold, however the optimizer hyperparameters play a role. In particular, when LR$>38/\lambda_{\max}(P^{-1}H)$, two mechanisms are now in play to resolve the step size being too big—either H can shrink or P^{-1} can shrink (or both). Cohen et al. [8] found that when β_2 is large, H tends to shrink and fast loss spikes result during the process, resembling the self stabilization process observed with gradient descent. However

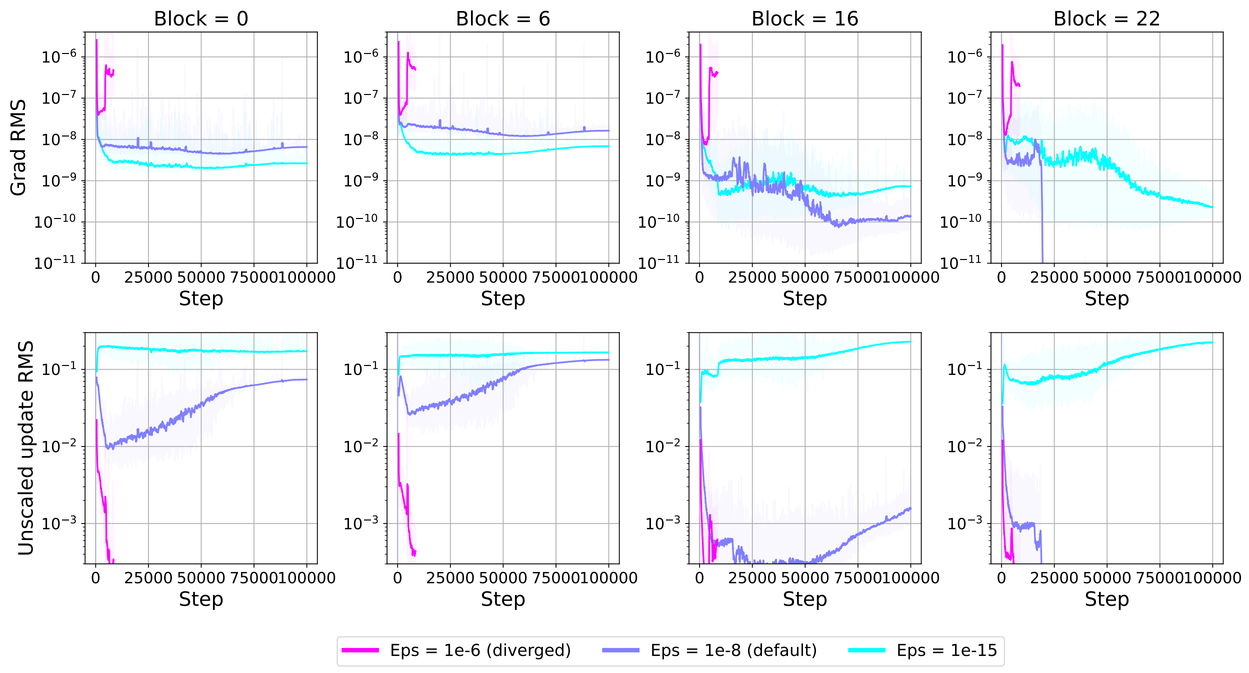

Figure 13: The top row displays the root mean square (RMS) of the gradient for the first MLP layer at different blocks throughout the network. When the grad RMS drops below the AdamW ϵ hyperparameter, the magnitude of the update decreases, as illustrated by the bottom row. Experiment conducted with a 4.8B parameter model trained with LR 0.3. The experiment with ϵ = 1e-6 was stopped when loss diverged.

when β_2 is small, P^{-1} tends to shrink, no loss spikes are observed, and $\lambda_{\max}(H)$ tends to gradually increase throughout training.

It is noteworthy that the adaptive edge of stability process (and the role of β_2) studied in Cohen et al. [8] offers a more complete understanding for loss spikes studied in a body of literature [42, 6, 35, 46, 51, 5]. For example, Shazeer and Stern [42] argue that during training of Transformers with adaptive optimizers the optimizer update can become too big resulting in a loss spike followed by recovery. This is sometimes attributed to the adaptive optimizer state becoming "stale", which is consistent with the observation the reducing β_2 resolves the loss spikes [42, 46, 51]. This is perhaps the same observation as Cohen et al. [8] that reducing β_2 allows P^{-1} to change quicker to adjust to the process of progressive sharpening. AEoS also offers an explanation for the periodic loss spikes observed when training large transformer models [35].

Parameter-free methods and more parameterizations. While our work has studied sensitivity to learning rate, there is also research that aims to eliminate the need to specify a learning rate [24, 10]. Based on their analysis, Ivgi et al. [24] set the step size for iteration t to the maximum distance from the initialization divided by the root sum of historical gradient squares. Moreover, while our work investigated μParam, there are additional parameterizations for which it would be interesting to explore LR vs. loss [13, 47, 3, 25].

5 Conclusion

As the compute required to train the largest models continues to increase, it becomes increasingly important to understand if training will be stable. This paper has shown that useful insights on stability can be found when studying small Transformers. We hope that this opens new opportunities for impactful research which benefits large runs without access to large resource pools.

Acknowledgements

We thank George Dahl for thorough comments and suggestions, and Hugo Larochelle and Rif A. Saurous for helpful discussion. Also, we thank the members of the Google DeepMind PAGI team for their support of this effort, Noah Fiedel, Noah Constant, Aaron Parisi, Alex Rizkowsky, Avi Singh, Azade Nova, Bernd Bohnet, Daniel Freeman, Gamaleldin Elsayed, Hanie Sedghi, Isabelle Simpson, James Harrison, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Max Bileschi, Merrie Morris, Rosanne Liu, Yundi Qian, Sharad Vikram, Tris Warkentin.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

[3] Blake Bordelon and Cengiz Pehlevan. Dynamics of finite width kernel and prediction fluctuations in mean field neural networks. *arXiv preprint arXiv:2304.03408*, 2023.

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[5] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9620–9629, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society. doi: 10.1109/ICCV48922.2021.00950. URL https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00950.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[7] Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

[8] Jeremy M Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E Dahl, et al. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022.

[9] Alex Damian, Eshaan Nichani, and Jason D Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *arXiv preprint arXiv:2209.15594*, 2022.

[10] Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. *arXiv preprint arXiv:2301.07733*, 2023.

[11] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. *arXiv preprint arXiv:2302.05442*, 2023.

[12] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[13] Emily Dinan, Sho Yaida, and Susan Zhang. Effective theory of transformers at initialization. *arXiv preprint arXiv:2304.02034*, 2023.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. https://arxiv.org/abs/2010.11929.

[15] Colin Gaffney, Dinghua Li, Ruoxin Sang, Ayush Jain, and Haitang Hu. Orbax, 2023. URL http://github.com/google/orbax.

[16] Justin Gilmer, Andrea Schioppa, and Jeremy Cohen. Intriguing properties of transformer training instabilities. To appear.

[17] Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instability in deep learning. *arXiv preprint arXiv:2110.04369*, 2021.

[18] 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*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[19] Google. Grain - feeding jax models, 2023. URL http://github.com/google/grain.

[20] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL http://github.com/google/flax.

[21] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[22] 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. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[23] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International Conference on Machine Learning*, pages 9099–9117. PMLR, 2022.

[24] Maor Ivgi, Oliver Hinder, and Yair Carmon. Dog is sgd's best friend: A parameter-free dynamic step size schedule. *arXiv preprint arXiv:2302.12022*, 2023.

[25] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. https://arxiv.org/abs/1806.07572.

[26] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

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

[28] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[29] Jaehoon Lee. A random walk model of transformer parameter growth, 2023.

[30] 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.

[31] Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hyg0vbWC-.

[32] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2016. https://arxiv.org/abs/1608.03983.

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. https://openreview.net/forum?id=Bkg6RiCqY7.

[34] William Merrill, Vivek Ramanujan, Yoav Goldberg, Roy Schwartz, and Noah Smith. Effects of parameter norm growth during transformer training: Inductive bias from gradient descent. *arXiv preprint arXiv:2010.09697*, 2020.

[35] Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, et al. A theory on adam instability in large-scale machine learning. *arXiv preprint arXiv:2304.09871*, 2023.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://arxiv.org/abs/1912.01703.

[37] Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of*

the Association for Computational Linguistics: Volume 2, Short Papers, pages 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025.

[38] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019. https://openai.com/blog/better-language-models/.

[39] 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. http://jmlr.org/papers/v21/20-074.html.

[40] 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*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[41] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. {ZeRO-Offload}: Democratizing {Billion-Scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564, 2021.

[42] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[43] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

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

[45] Mitchell Wortsman, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Replacing softmax with relu in vision transformers.

[46] Mitchell Wortsman, Tim Dettmers, Luke Zettlemoyer, Ari Morcos, Ali Farhadi, and Ludwig Schmidt. Stable and low-precision training for large-scale vision-language models. *arXiv preprint arXiv:2304.13013*, 2023.

[47] Sho Yaida. Meta-principled family of hyperparameter scaling strategies. *arXiv preprint arXiv:2210.04909*, 2022.

[48] Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

[49] 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. *arXiv preprint arXiv:2203.03466*, 2022.

[50] Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Josh Susskind. Stabilizing transformer training by preventing attention entropy collapse. *arXiv preprint arXiv:2303.06296*, 2023.

[51] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*, 2023.

[52] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

A Additional infrastructure details

This Section provides more details on the training infrastructure, which is built on Flax [20], Jax [4], and TPUs [26], and we call NanoDO. To enable larger model training, we shard the model and optimizer states as in FSDP [41], then specify these shadings when compiling with JIT. We use Orbax [15] for checkpointing, and Grain [19] for deterministic data loading. When loading data, sequences are packed so that no padding is required—if a sequence is less tokens than the context length hyperparameter, then an end of sequence token is appended, followed by the beginning of a new sequence.

B When is learning rate sensitivity a useful metric

There are cases where LR sensitivity (defined in Section 2.2) is no longer a useful metric. This section details these scenarios and justifies the use of LR sensitivity for the interventions in this paper.

Interventions which change the meaning of learning rate

When an intervention changes the meaning of learning rate then comparing LR sensitivity is not useful. A clear example of this would be taking the square root of the LR before passing it to the optimizer, but there are more subtle cases to be cautious of when using LR sensitivity.

In general, we avoid manipulations where the meaning of LR meaningfully changes. In some cases, we have good empirical evidence that the meaning of the learning rate has not changed when intervening. For instance, the LR vs. loss curves are indistinguishable up to some critical learning rate when using qk-layernorm (Figure 1), adding z-loss (Figure 3), or changing warm-up.

In other cases, such as when testing μParam (Section 3.2.4), we believe that LR sensitivity is useful despite a per-layer modification of LR. This is because the per-layer LR is manipulated linearly, and this modification does not change for different points on the LR vs loss curve.

The one experiment in this paper where we believe LR sensitivity is likely not a useful metric is when scaling learning rate by the root mean square of the parameters (Figure E.14). Therefore, we do not measure LR sensitivity in that case.

Shifting of the optimal LR

The definition of LR sensitivity in Section 2.2 does not account for the optimal LR shifting when specifying the LR range $[a, b]$. In practice we recommend shifting the three order of magnitude range $[a, b]$ to correspond with this shift. For instance, we shift the range in Section 3.2.2, as discussed in more detail in the section. However, our main experiments (e.g., Figure 1) do not test at a large enough scale to necessitate this shift.

LR sensitivity is invariant to loss

Another limitation of the LR sensitivity metric is that it is invariant to the scale of the loss. If the network consistently achieves random performance across learning rates, then LR sensitivity will be zero. We do not offer a solution to this, and instead recommend that LR sensitivity should always be examined in combination with the LR vs. loss curves as we do in this paper. It is meant as a useful summary of the LR vs. loss curves, not as a metric to optimize in isolation.

C Parameter and output norm growth

This section discusses the growth of the parameter norm during Transformer training as previously studied by Merrill et al. [34], Lee [29], and relates this phenomenon to the attention logit growth and AdamW epsilon instabilities (Sections 3.1.1 and 3.4, respectively).

An observation of Lee [29] is that, when using an adaptive optimizer, the movement of parameters can be approximated by a random walk. We show parameter root mean square (RMS) throughout training in Figure C.1[4], which appears to follow a predictable trend. This aligns with the aforementioned observation, and is further supported by Figure C.2. Figure C.2 displays parameter RMS as a function model scale and learning rate, averaged over the last 500 training steps. As before, parameter RMS is determined primarily by learning rate. As parameter RMS grows, we would expect output RMS to also grow. This is validated by

[4]We show parameter RMS for the first MLP layer in various blocks, but expect other layers to exhibit similar trends.

Figure C.3, which shows that the RMS of the Transformer block output is mainly determined by learning rate, and follows a very similar trend to parameter RMS.

There are two interesting takeaways. First, this observation helps to explain why the attention output logits become large at high learning rates as observed by Dehghani et al. [11] and Section 3.1.1. This is the only feature in the network we test whose magnitude depends quadratically on parameter RMS. For inputs X with unit RMS, a typical matrix multiply XW with parameters W will result in features Y where RMS(Y) is a linear function of RMS(W). On the other hand, the attention logit entries are computed via $\langle XW_1, XW_2 \rangle$ so depend quadratically on RMS(W). They are therefore the first to become large when the parameter norm grows. Next, this helps to explain the decreasing trend in gradient scale observed in Section 3.4 (Figure 11). In a pre-normalization [38] Transformer [44] there is an output layernorm layer [1] after the last Transformer block and before the final linear layer. The gradient from this output layernorm layer is scaled by the reciprocal of the input RMS. In addition to growing with LR, this RMS is growing with depth because of the residual connections (Figure C.3). As the RMS leaving the last Transformer block grows, the gradient received shrinks.

For completeness we now compute the layernorm gradient to input x. We assume the input as mean zero and the layernorm has no bias for simplicity. Let

$$z = \text{LayerNorm(x)} = \alpha \cdot \frac{x}{\sqrt{\mathbb{E}_i\left[x_i^2\right] + \epsilon}} = \alpha \cdot \frac{x}{m^{1/2}} \tag{1}$$

where $m = \mathbb{E}_i\left[x_i^2\right] + \epsilon$.

Then

$$\frac{\partial \ell}{\partial x_j} = \sum_k \frac{\partial \ell}{\partial z_k}\frac{\partial z_k}{\partial x_j} \tag{2}$$

$$= \frac{\partial \ell}{\partial z_j} \cdot \frac{\alpha_j}{m^{1/2}} + \sum_k \frac{\partial \ell}{\partial z_k} \cdot \left(-\frac{1}{2}\right) \cdot \frac{\alpha_k x_k}{m^{3/2}} \cdot \frac{2}{n} \cdot x_j \tag{3}$$

$$= \frac{1}{m^{1/2}}\left(\alpha_j \frac{\partial \ell}{\partial z_j} - \frac{x_j}{nm^{1/2}} \sum_k \frac{\partial \ell}{\partial z_k}\alpha_k x_k\right) \tag{4}$$

$$= \frac{1}{m^{1/2}}\left(\alpha_j \frac{\partial \ell}{\partial z_j} - \frac{x_j}{nm^{1/2}} \langle \nabla_z, \alpha \cdot x\rangle\right) \tag{5}$$

Equivalently,

$$\nabla_x = \frac{1}{m^{1/2}}\left(\alpha \odot \nabla_z - \frac{\langle \nabla_z, \alpha \odot x\rangle}{nm^{1/2}} \odot x\right). \tag{6}$$

D Author contributions

Mitchell Wortsman led the project, ran the experiments and produced the figures, contributed substantially to the infrastructure for experimentation, the framing and direction, and the writing.

Peter J. Liu led the infrastructure and creation of NanoDO for experimentation, provided key insights and advice on multiple technical areas, and contributed to the writing.

Lechao Xiao and Katie Everett contributed to the infrastructure used for experimentation, provided key insight related to parameterization, and contributed to the writing.

Alex Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-dickstein, and Kelvin Xu were active participants in weekly brainstorming meetings which motivated, influenced, and elucidated technical concepts pertaining to this work.

Jaehoon Lee and Justin Gilmer were senior authors advising on the project, contributed substantially to the framing and direction, provided key insight and advice on multiple technical areas, and contributed to the writing. Jaehoon led the connection with parameter norm growth. Justin proposed to plot loss as a function of learning rate for different model sizes, and performed initial experiments demonstrating that attention logit growth could be reproduced at high learning rates in small models.

Simon Kornblith was the lead advisor on the project, contributing substantially to the framing, direction, infrastructure, and writing. Simon initially brainstormed the project with Mitchell, and was Mitchell's host for the summer internship during which this research was conducted, providing substantial technical support.

E Additional figures

This Section contains the additional Figures referenced in the main text.

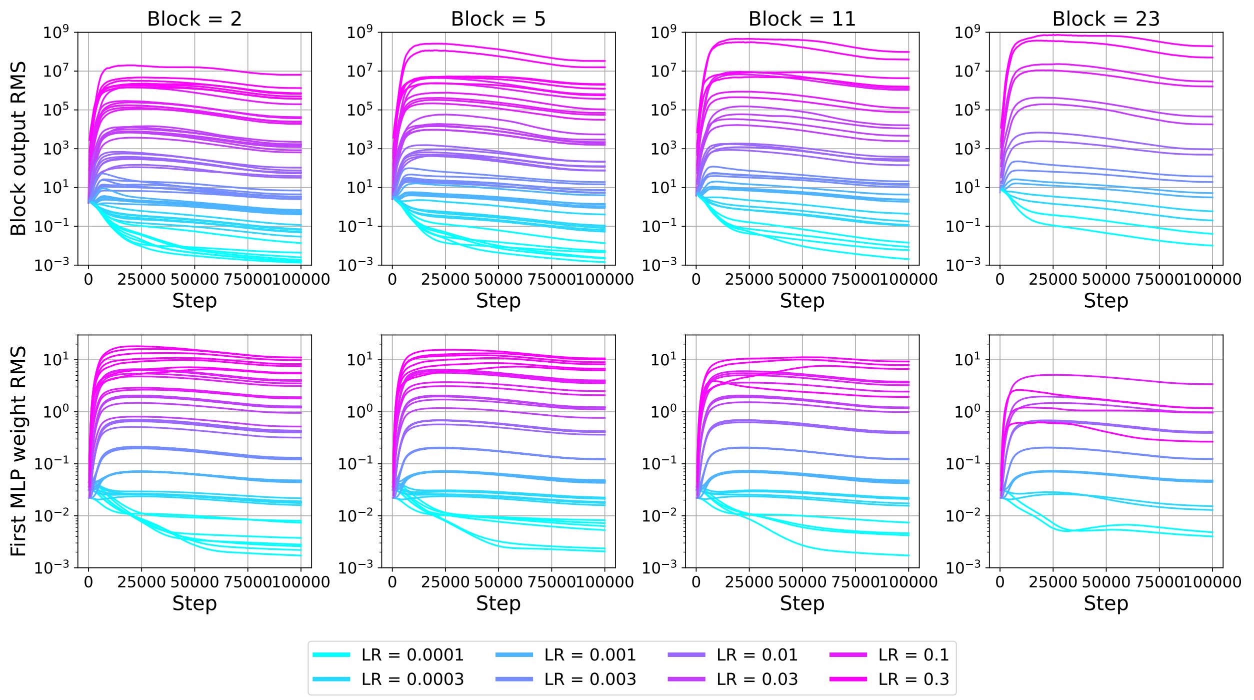

Figure C.1: (Top) The root mean square (RMS) of the Transformer block outputs throughout training. (Bottom) The RMS of the MLP weights throughout training, for the first of the two layers in the MLP. Recall $\text{RMS}(X) = \sqrt{\mathbb{E}_i[X_i^2]}$. RMS is mostly determined by LR, with higher LR corresponding with a higher RMS for block outputs and MLP weights. Different curves at the same learning rate correspond to different model scales. This experiment uses a decoupled weight decay values of 1e-4.

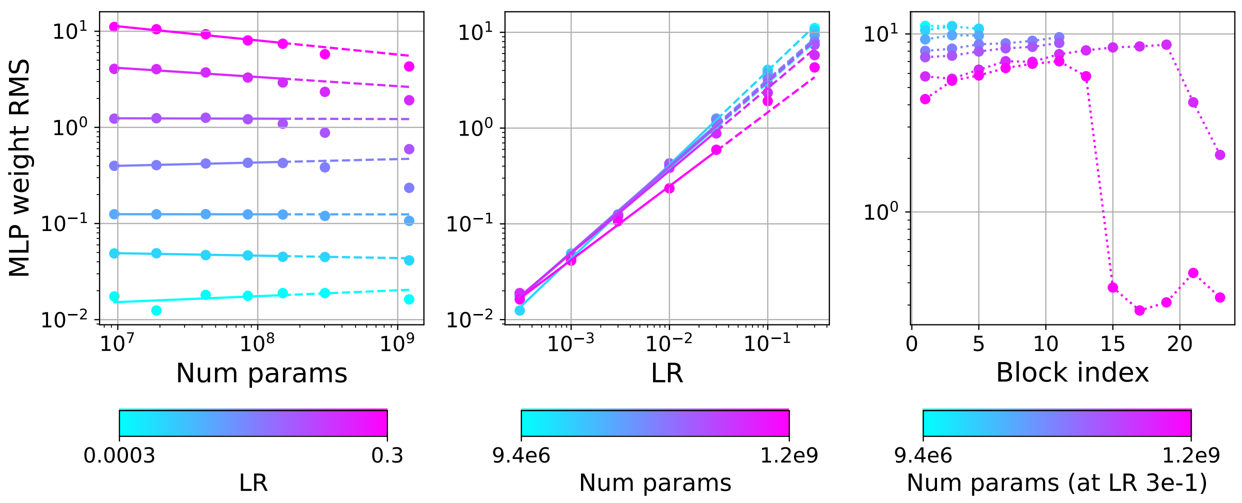

Figure C.2: The root mean square (RMS) of the MLP weights are roughly consistent with scale (left) but increase reliably with learning rate (center). At high learning rates, parameters later in the network can be affected by the AdamW epsilon instability discussed in Section 3.4 (right). This experiment considers the first of the two MLP layers in the block, and data for the first two plots are from block two. RMS is averaged over the final 500 training steps, where $\text{RMS}(X) = \sqrt{\mathbb{E}_i[X_i^2]}$.

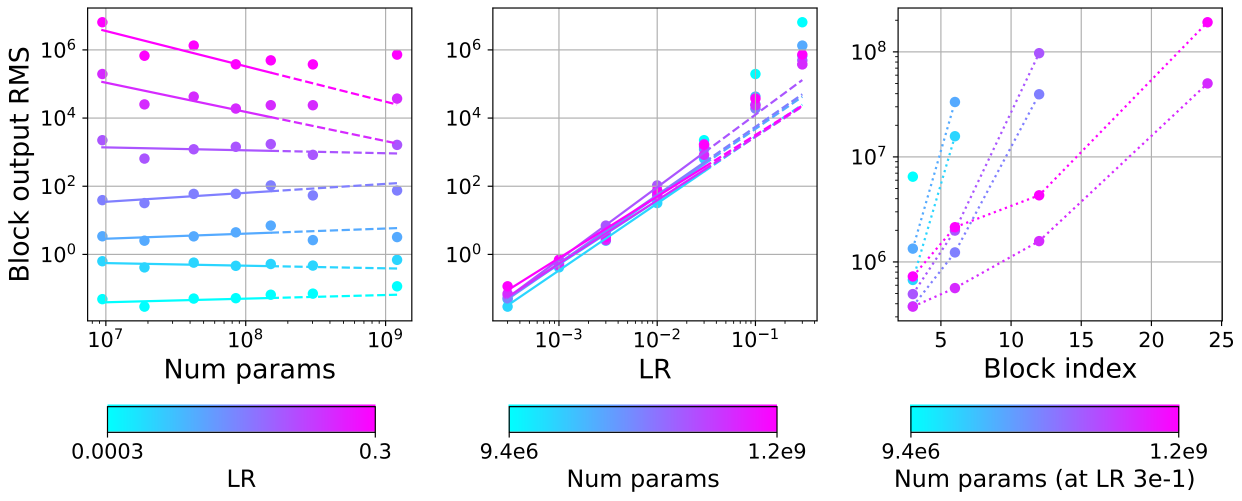

Figure C.3: The root mean square (RMS) of the Transformer block outputs are roughly consistent with scale (left) but increase with learning rate (center). RMS increases deeper in the transformer because of the residual connections, which is shown for very high learning rates (right). The first two plots are for block index two, and RMS is averaged over the final 500 training steps. Recall $\mathrm{RMS}(X) = \sqrt{\mathbb{E}_i[X_i^2]}$.

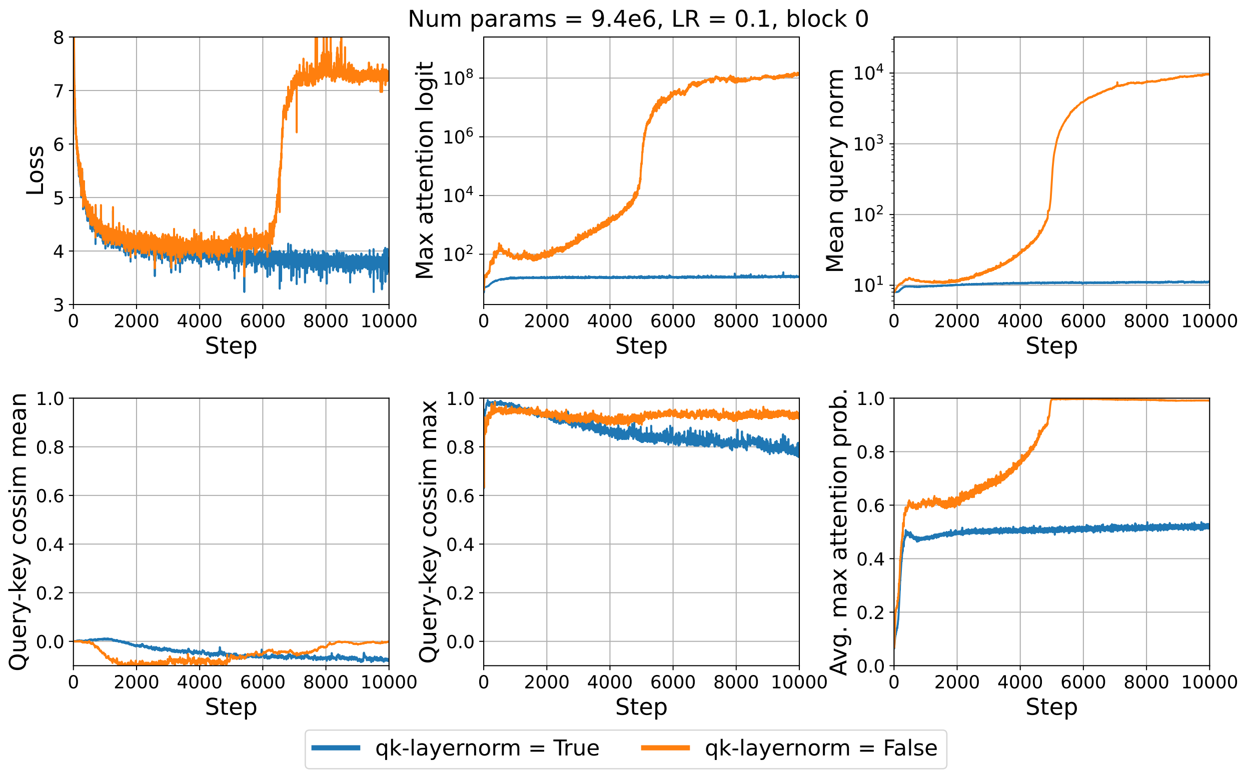

Figure E.1: The logit growth instability [11, 50] occurs when the norm of the query and keys increases, not due to an increase in their cosine similarity.

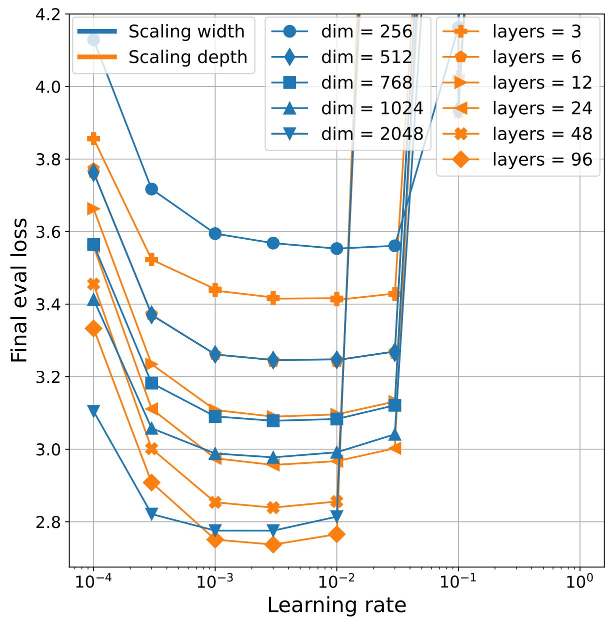

Figure E.2: The effect of scaling width vs. scaling depth without qk-layernorm [11].

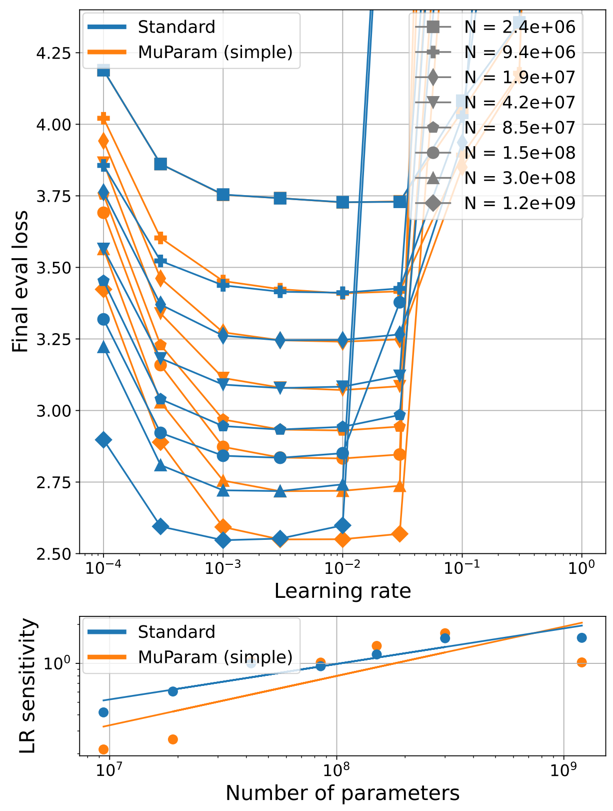

Figure E.4: The effect of μParam on LR sensitivity for models without qk-layernorm [11]. μParam succeeds in stabilizing the optimal LR, but does not alleviate the need for qk-layernorm. For more information refer to Section 3.2.4.

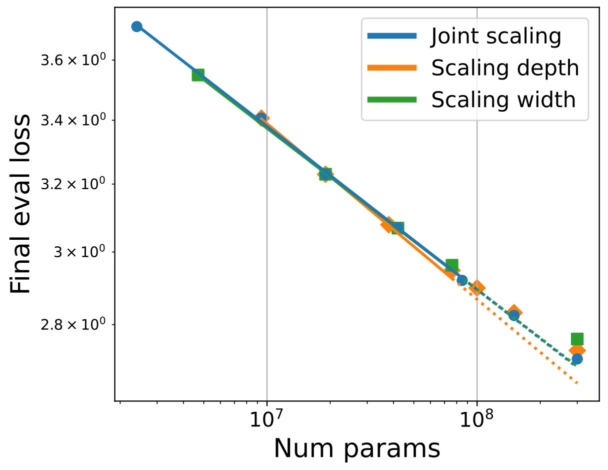

Figure E.3: Jointly scaling width and depth leads to lower loss than independently scaling depth or width at the largest scale we test. It also leads to a more reliable scaling prediction when extrapolating from models with less than 1e8 parameters. Best loss is reported in a sweep over learning rates.

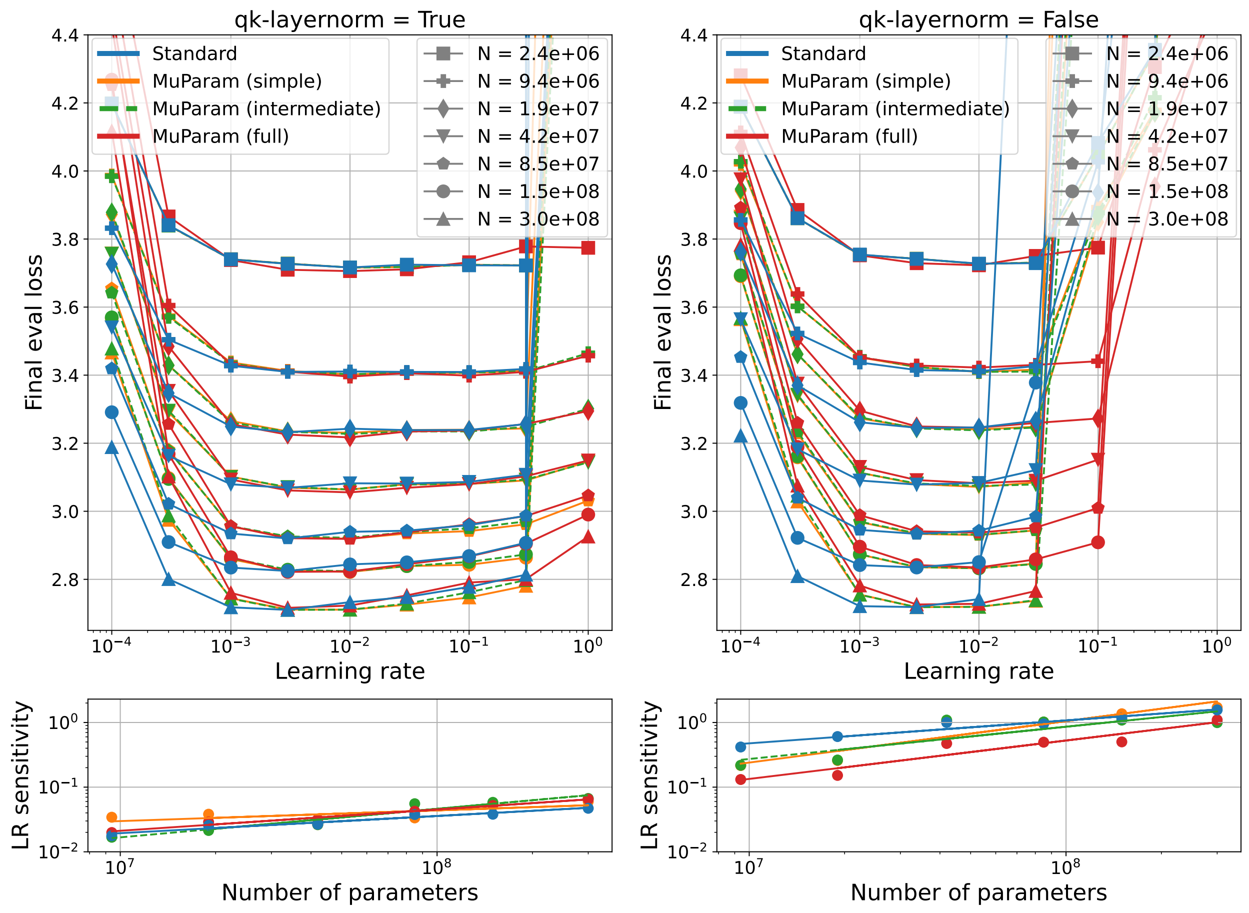

Figure E.5: Comparing μParam (full), which implements μParam as described in Yang et al. [49] with and without qk-layernorm, with μParam (simple) and μParam (intermediate). There are four changes in μParam (full), (i) Scale the LR for linear layers by base-fan-in/fan-in, (ii) initialize the head with standard deviation $\sqrt{\text{base-fan-in}/\text{fan-in}}$. (iii) change the $1/\sqrt{d_h}$ scaling factor in attention layers to $1/d_h$ where d_h is the head dimension, and (iv) initialize the query projection weights with zeros. μParam (intermediate) consists of (i) and (ii), while μParam (simple) is only (i). With μParam (full) and qk-layernorm, the model trains without diverging at LR 1. However at the best LR there is no measurable improvement over μParam (simple) at the largest scale we test.

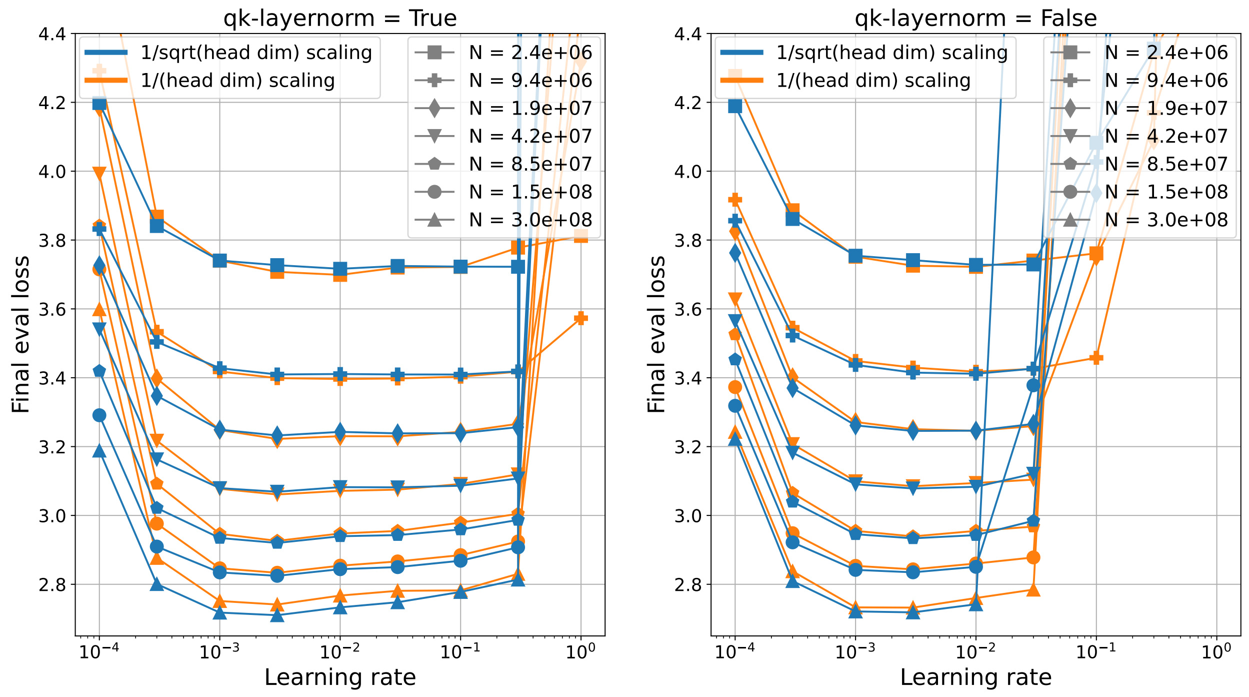

Figure E.6: Measuring the effect of changing the $1/\sqrt{d_h}$ term in attention to $1/d_h$, where d_h is head dimension. Vaswani et al. [44] use $1/\sqrt{d_h}$ while Yang et al. [49] use $1/d_h$.

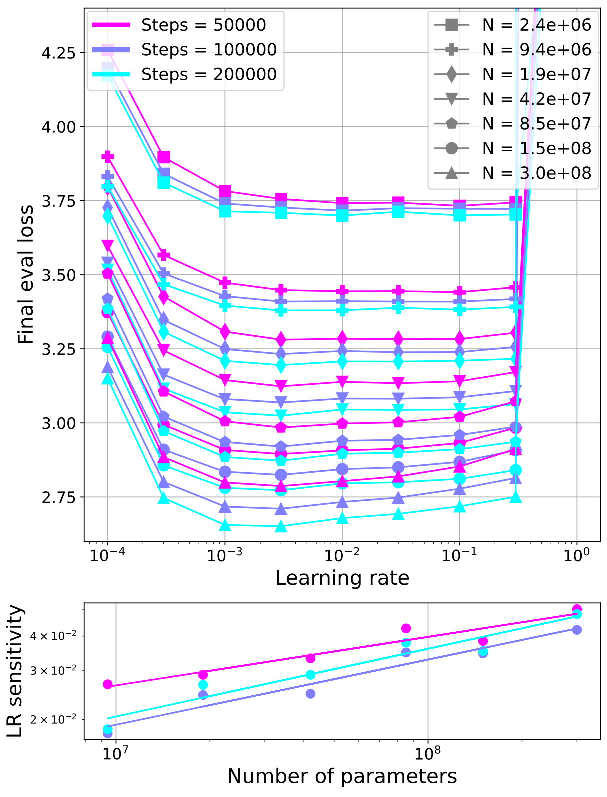

Figure E.7: Changing the number of total training steps from 1e5 to 5e4 or 2e5 does not have a large effect of the shape of the learning rate vs. loss curves at the scales we test.

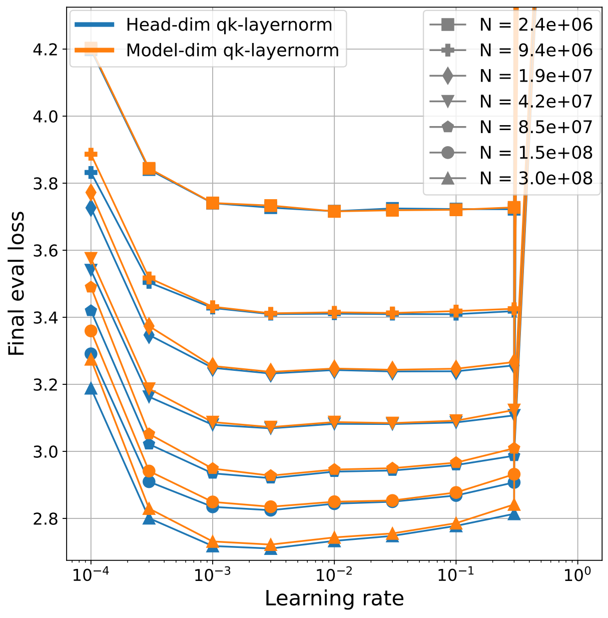

Figure E.8: We achieve slightly better performance when applying qk-layernorm individually per-head instead of across the model dimension. The per-head variant has only head-dim learnable parameters instead of model-dim parameters. We use the per-head variant as the default in this paper, and we never use biases.

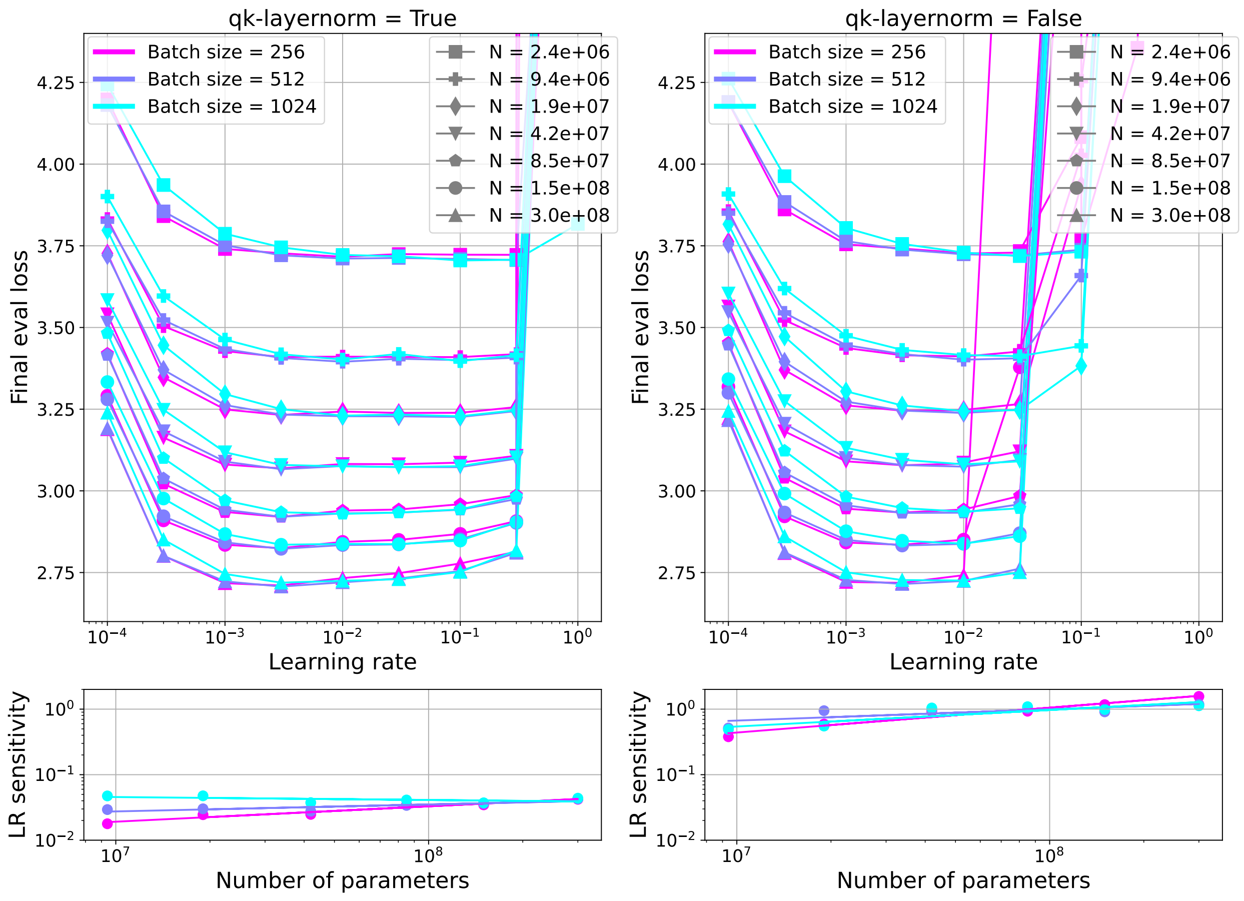

Figure E.9: Increasing the batch size from 256 to 512 or 1024 does not have a large effect on the shape of the learning rate vs. loss curves at the scales we test. Each batch element contains 512 tokens, and we use 256 as the default.

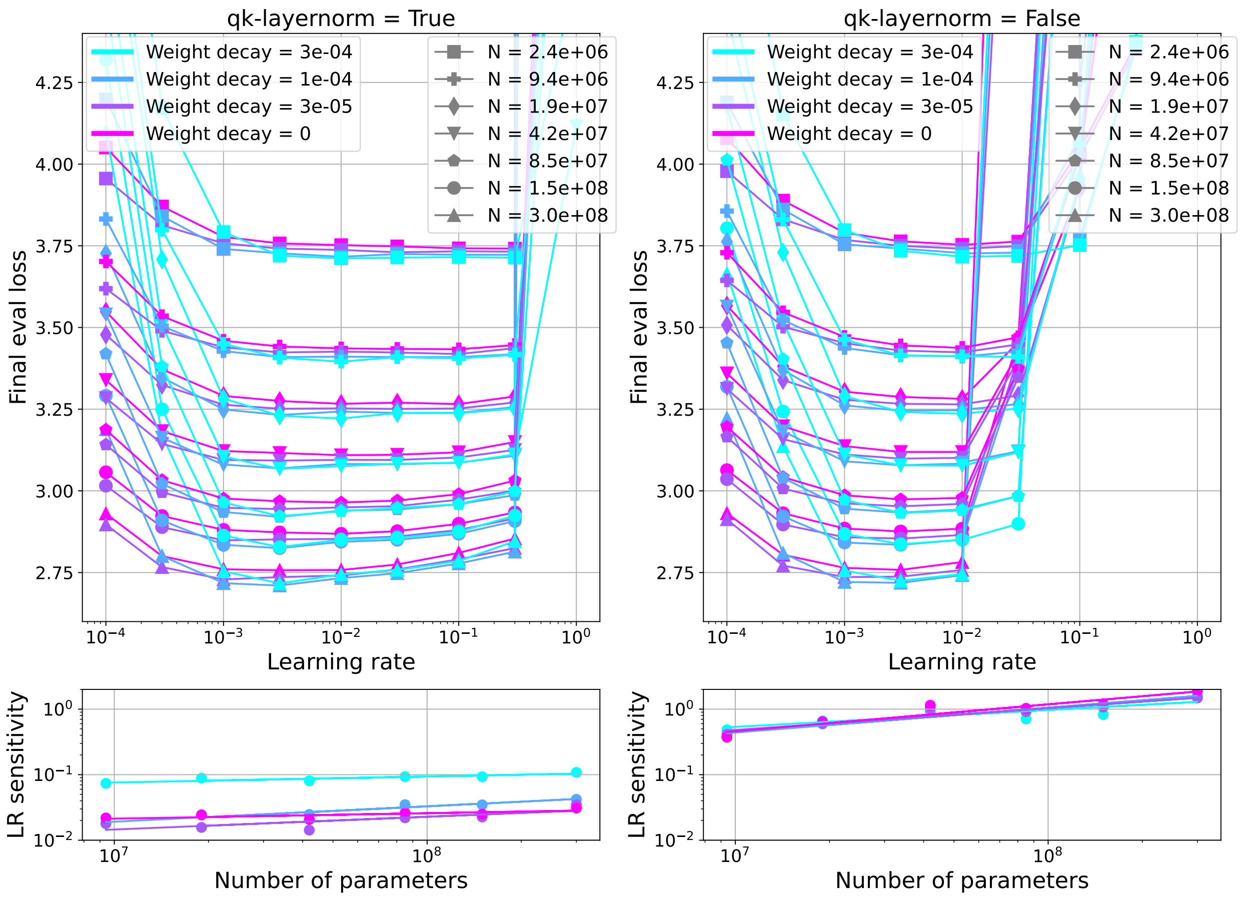

Figure E.10: The effect of weight decay on LR sensitivity. We use independent weight decay as described in Section 3.2.2 and recommended by [33].

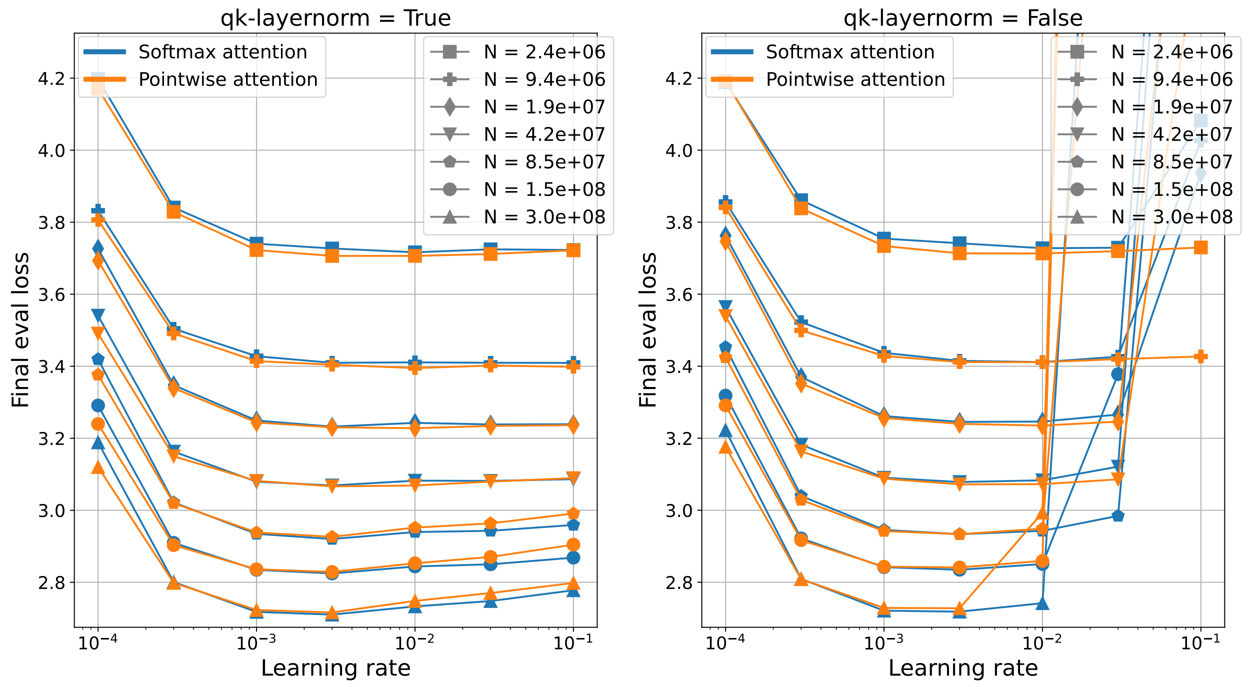

Figure E.11: The logit growth instability occurs even without softmax. For the pointwise variant of attention here, we replace softmax with squared-relu as described by [23]. As recommended in [45] we add a scaling factor which depends on sequence length. In this case, we use inverse square root.

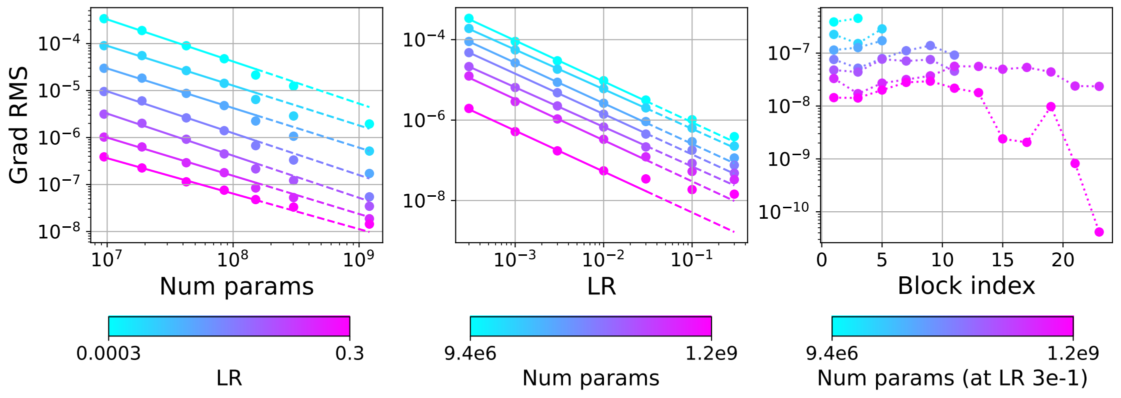

Figure E.12: Recreating Figure 11 with the kernel projection instead of the first MLP layer.

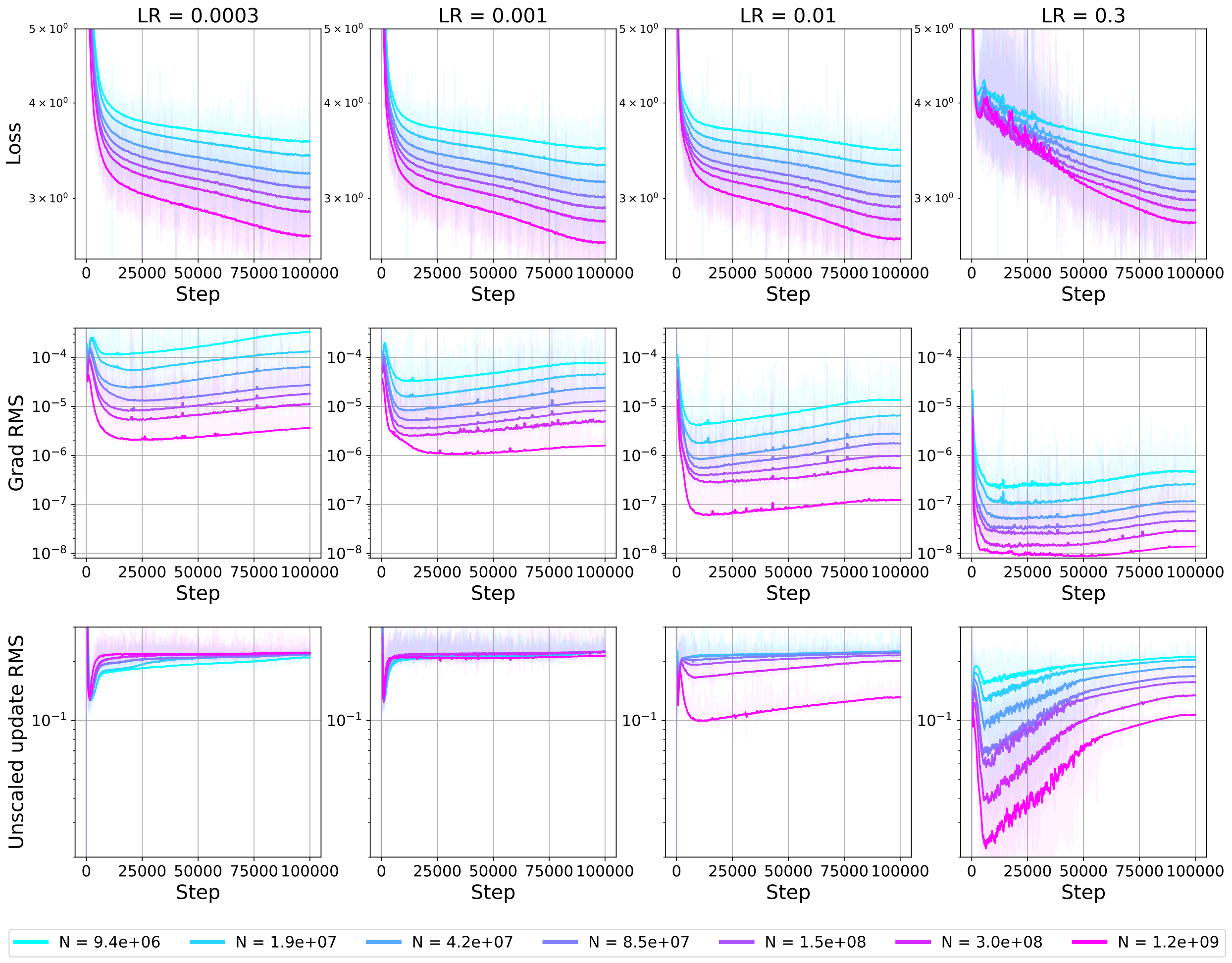

Figure E.13: For various learning rates and model sizes we display the gradient root mean square (RMS), and the unscaled update RMS. The unscaled udpate is the update returned by the optimizer before scaling by learning rate. The gradient and update are shown here for the first MLP layer of the Transformer. The update RMS falls when the grad RMS approaches the AdamW ϵ of 1e-8.

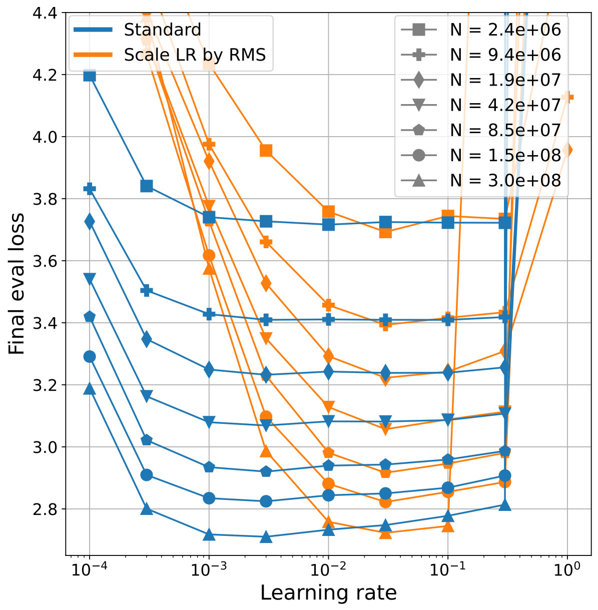

Figure E.14: The effect of scaling the learning rate for parameters p by $\max(\mathrm{RMS}(p), \text{1e-3})$ as in AdaFactor [42]. As discussed by Appendix B, it is not meaningful to compare LR sensitivity in this case as this intervention modifies the meaning of learning rate. Just as in μParam [49], RMS scaling appears to stabilize the optimal LR in the range we test.

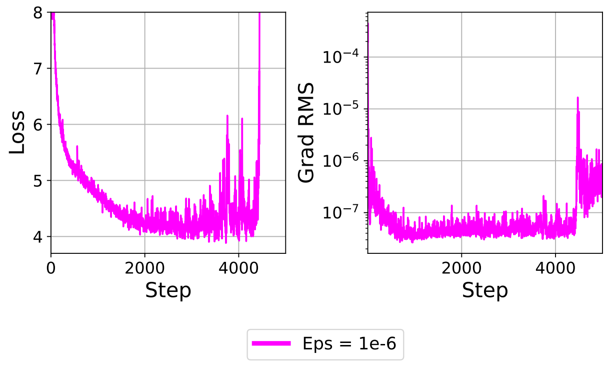

Figure E.15: Increasing the AdamW ϵ from its default value of 1e-8 to 1e-6 causes a loss divergence for a 4.8B parameter model at LR 0.3. Grad RMS is for the first layer in the MLP.