An Empirical Study on Noisy Data and LLM Pretraining Loss Divergence

Qizhen Zhang[1,†] **Ankush Garg** [†] **Jakob Foerster** [1,2]
Niladri Chatterji[*,†] **Kshitiz Malik**[*,†] **Mike Lewis**[*,2]
[1] University of Oxford [2] Meta FAIR

Abstract

Large-scale pretraining datasets drive the success of large language models (LLMs). However, these web-scale corpora inevitably contain large amounts of noisy data due to unregulated web content or randomness inherent in data. Although LLM pretrainers often speculate that such noise contributes to instabilities in large-scale pretraining and, in the worst cases, loss divergence, this phenomenon remains poorly understood. In this work, we present a systematic empirical study on whether and how noisy data causes LLM pretraining divergences, costing more than 100,000 GPU hours in total. By injecting controlled, synthetic, uniform random noise into otherwise clean datasets, we analyze training dynamics across model sizes ranging from 480M to 5.2B parameters. We show that noisy data indeed induces training loss divergence, with divergence probability depending strongly on noise structure, amount of noise, and model scale. We further find that noise-induced divergences exhibit activation patterns distinct from those caused by high learning rates [1], a common source of pretraining instability. Building on these differences, we propose diagnostics for distinguishing the two failure modes and identify interventions that mitigate noisy data induced divergences. Together, these results provide a large-scale, controlled characterization of how noisy data affects loss divergence in LLM pretraining.

1 Introduction

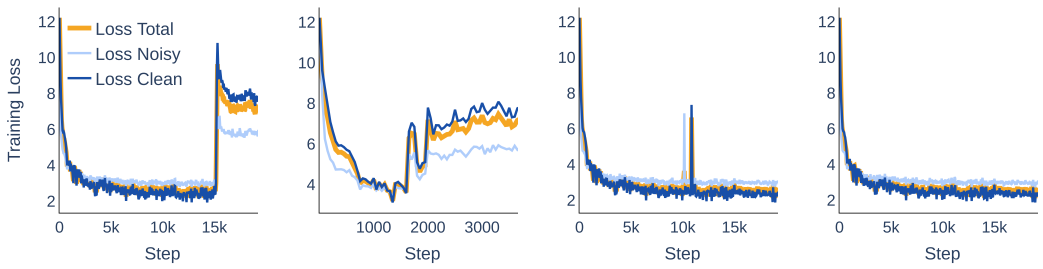

Figure 1: Examples of four pretraining runs with the same 1.3B model and 15% noisy data, differing only in random seed. The left two runs diverge, while the right two remain stable. We focus on divergences rather than fast spikes, so the third run is categorized as stable because its loss spike quickly recovers.

[*]Joint last authors.
[†]Work done at Meta
Corresponding author: qizhen.zhang@eng.ox.ac.uk

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Noise type	Example
Hash strings	`_integrity: "sha1-tSQ9jz7BqjXxNkYFvA0QNuMKtp8="`
Random identifiers	`guid: f4ad9adb86aca2949bb17cc2ea304593`
URLs	`href="/maf/wlin/api/tocectory?tocguid=I29AAA0404A3E11DF96C1A49DA6EE4E28&..."`
Credentials	`aws_secret_access_key=MmY9heKrWiNVCSZQ2Mf5XTJ6Ys93Bw2d2D6`
Numeric sequences w/o context	1.01458, 1.02937, 1.04438, 1.05961, . . .

Table 1: **Examples of noise patterns observed in The Stack dataset [6]** from a manual audit of 100 randomly sampled documents (approximately 43k tokens). Highlighted examples illustrate examples noise, including hash strings, randomly generated identifiers, URLs, credentials, and standalone numeric sequences.

Scaling up transformer models in terms of model size and datasets has led to remarkable capabilities, but the resulting large-scale pretraining runs are extremely expensive [2, 3]. Furthermore, not every pretraining run succeeds [4, 5]: researchers frequently observe instabilities that slow or disrupt learning, and in the worst case, the pretraining loss diverges entirely. Understanding these failures is therefore important. One hypothesized cause of loss divergence is noisy data in the training corpus. Web-scale datasets inevitably contain substantial noise due to unregulated web content or randomness inherent in data. Yet whether such noise truly drives divergence remains poorly understood. One of the most common types of noise in pretraining text data is *uniform random noise*. For example, from a manual audit of 100 randomly sampled documents from The Stack dataset [6], we found many examples of uniform random noise, including random identifiers and hash strings (see Table 1). Motivated by these examples, we define uniform random noise as sequences of tokens sampled uniformly from a subset of the tokenizer vocabulary. We restrict sampling to a subset because the noisy examples observed in practice, which consist largely of gibberish alphanumeric strings, typically draw from only a small fraction of the full vocabulary. These sequences contain no meaningful semantic structure for the model to learn. As a result, the optimal strategy is to predict uniformly over the subset.

In this work, we present a systematic empirical study investigating whether and how uniformly random noise in pretraining data affects LLM training stability. We focus on the worst-case instabilities that lead to loss *divergence* rather than fast loss spikes (see Figure 1). By injecting controlled synthetic uniform random noise into otherwise clean datasets, we examine training dynamics across model sizes from 480M to 5.2B parameters using compute resources totaling over 100,000 GPU hours. Our contributions are fourfold:

1. **Noisy data causes divergence, with more divergence at larger scales** (§ 4.1, § 4.2): We show that noisy data can cause probabilistic training divergence, with the divergence probability increasing with the noise ratio, model size.
2. **Noise structure affects divergence probability** (§ 4.3): Smaller noise vocabularies and insertion-based noise are more destabilizing, causing divergence even in models with less than 1B parameters, which are otherwise typically stable to train.
3. **Dense and MoE models have comparable noise sensitivity** (§ 4.4): Dense and active-parameter-matched Mixture-of-Experts (MoE) models exhibit similar divergence rates.
4. **Diagnostic & interventions** (§ 4.5, § 4.6): We show that noisy data induced divergence and high learning rate induced divergence, a common pretraining failure mode, exhibit different activation and parameter patterns. Using these differences, we propose diagnostics to identify the likely cause of a divergent run. In addition, we show that data cleaning is the best intervention. When data cleaning is not feasible, we show that QK-LayerNorm also stabilizes training at the cost of model performance.

Together, these results provide one of the first large-scale, controlled characterization of whether and how uniform random noise causes LLM pretraining loss divergence, which we hope motivates future work on the mechanisms linking data quality and training instability.

2 Related work

LLM pretraining instabilities Many works analyze pretraining instabilities through the lens of abnormal activation and parameter behavior. Approaches such as QK-layer normalization [7], z-loss regularization [4], and weight decay [8] stabilize training by modifying model architectures or loss functions to mitigate these abnormal behaviors. In contrast, relatively few works examine how *training recipe decisions* give rise to these activation and parameter pathologies in the first place. Prior research in this direction has examined numerical precision and quantization, showing that reduced precision can induce pathological activation and gradient behavior [9–11], as well as learning rate choices, where Wortsman et al. [1] demonstrate that overly high learning rates can cause abnormal activation and parameter growth and lead to loss divergence even in small language models. Our work studies an orthogonal training recipe choice: data quality. We show that noisy pretraining data can cause loss divergence, and that noisy data-induced failures exhibit activation and parameter patterns distinct from those caused by high learning rates. Based on this distinction, we introduce diagnostic methods that separate learning rate induced failures from noisy data induced failures, enabling practitioners to identify when data cleaning is required.

LLM pretraining data quality Prior work on pretraining data quality, including deduplication[12] and data-selection pipelines [13], has largely focused on improving model performance. In contrast, we study how data quality affects *training stability*, focusing on uniform random noise, a common form of noise in web data. Other data quality issues, such as duplication, involve different mechanisms and would require substantial additional compute, so we leave them outside the scope of this work. The most closely related work is Ru et al. [14], which finds that noisy data have limited effects on training dynamics, reporting only modest loss increases and no training spikes or divergence. However, their analysis considers a narrow noise setting in which clean text is corrupted using tokens sampled from the full tokenizer vocabulary. We show that noise structure matters: certain forms of random noise, particularly noise restricted to a subset of the vocabulary, substantially increase the probability of loss divergence. By systematically varying noise type, noise ratio, and model size, we show that noisy data can destabilize training and characterize when this instability arises, how it can be diagnosed, and how it can be mitigated.

Noisy label works prior to LLMs A substantial body of theoretical and empirical research has studied neural network training under noisy labels in supervised learning settings [15–23]. These works typically focus on small-scale models and datasets, studying how noisy labels affect model performance, generalization, and overfitting. In contrast, our work studies *large-scale*, *transformer-based* language models and investigates how noisy data leads to *training loss divergence*. As demonstrated in § 4.2, these instabilities become increasingly pronounced at larger model scales: configurations that are stable for small models, where most prior noisy-label studies are conducted, do not remain stable when scaled to the regimes we consider. Moreover, we provide diagnostic insights for identifying noisy data induced failures specific to transformer architectures.

3 Experimental methodology

3.1 Noisy data generation

Our study focuses on *uniform random noise*, which arise from unregulated web content or inherently random sequences (see Table 1). Since collecting a sufficiently large corpus of genuine random noise from web crawls is computationally infeasible, we instead simulate it synthetically in a controlled setting. We construct noisy training data from a clean corpus D_c, a subset of the Llama 4 pretraining mixture[3]. Let V denote the tokenizer vocabulary and $\alpha \in (0, 1)$ the target noise ratio. Each noise token is independently and uniformly sampled from a restricted noise vocabulary $V_N \subseteq V$. We use a restricted V_N because real-world noise typically occupies only a small subset of the vocabulary, such as digits and letters in hash-like strings. We defer details on the choice of V_N to § 4.3. For each document with n tokens, we inject noise using one of two methods. **Inserting:** sample $\frac{n\alpha}{1-\alpha}$ insertion positions with replacements, allowing consecutive noise tokens. **Overwriting:** replace each token independently with a noise token with probability α.

3.2 Model architecture details

Dense Models We train decoder-only transformer models [24] using the standard auto-regressive next-token prediction objective. Our dense architecture follows the Llama 3 family of models [2]: we use pre-normalization transformers, we do not use QK-layernorm [7], we do not use biases, we do not tie the input and output embedding weights, and we use rotary positional embeddings [25] and group query attention [26]. To vary model size, we adjust the number of layers and the model dimension [1]. We always jointly scale up the model dimension and number of query heads. Reported parameter counts include all parameters, including input and output embeddings.

MoE Models For Mixture of Experts [MoE; 27, 28] experiments, we use dropless MoEs [29, 30] with 16 feed-forward network (FFN) experts and token-choice top-2 routing[1]. For fair comparison with dense models, we match the active parameters of the top-2 MoE models by scaling the FFN dimension by 0.5. All other architectural components remain the same as the dense setting.

3.3 Training details

We use AdamWc[8] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1e-8$, global norm gradient clipping of 1.0, and weight decay $1\mathrm{e}-4$. We apply a learning rate (LR) schedule with a 2000-step linear warm-up followed by cosine decay, with a minimum LR ratio of 0.1. Following prior work studying pretraining instabilities [1], we similarly used a reduced training horizon of $2\mathrm{e}4$ total steps to keep the experiments compute tractable. We use a peak LR of $lr = 1.85e-2$ and a batch size of $2.6\mathrm{e}5$ tokens, and we train for less than one epoch in all experiments. To enable hyperparameter transfer across model sizes, we adopt Maximal Update Parametrization [31] with a base width of 256. All experiments are trained on 16 to 64 H100 GPUs, which costs >100,000 GPU hours in total.

3.4 Measuring training stability

Training outcomes can vary with random seed[2]. We therefore repeat each experiment across 20 seeds and report the fraction of runs that diverge. We classify a run as diverged if its loss exceeds the minimum observed so far by more than 0.5 nats/token for 500 consecutive steps, treating loss spikes that quickly recover as stable. For every experimental setting, we also run the corresponding clean-data experiment across 20 seeds and verify that none exhibit training instability. Thus, any divergence observed after noise injection can be attributed to the noisy data rather than to an inherently unstable training recipe.

4 Results and analysis

We first establish that noisy data causes probabilistic loss divergence (§ 4.1), then examine how the divergence probability depends on model scale and noise ratio (§ 4.2), noise structure (§ 4.3), and dense versus MoE architectures (§ 4.4). Next, we propose diagnostics based on activation and parameter statistics to separate noisy-data-induced divergences from high-LR-induced divergences (§ 4.5), one of the most common reasons for pretraining loss divergence. Finally, we propose and evaluate different interventions for stabilizing noisy data runs (§ 4.6).

In several experiments, we use tiny models ($<$ 1B parameters) with higher noise ratios and more constrained noise as a compute-efficient proxy for larger models under realistic noise levels. In § 4.2 and § 4.5, we show that this proxy exhibits similar training dynamics to those of larger models. Prior work on pretraining stability adopts a similar strategy for the same reason [1].

4.1 Noisy data causes probabilistic loss divergence

We first show that noisy data can indeed induce pretraining loss divergence. Importantly, this divergence is *probabilistic*. In Figure 1, all four runs use the same 1.3B dense model architecture, hyperparameters, and 15% noise setting, differing only in random seed. Two runs diverge, while the other two remain stable, including one that exhibits a fast loss spike before quickly recovering. In

[1] see Appendix A for more details.

[2] In addition to initialization, GPU nondeterminism may also contribute.

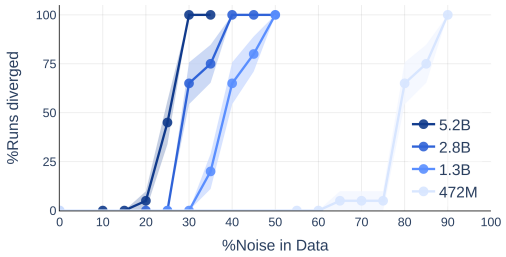

Figure 2: **Larger models diverge at lower noise ratios.** We jointly scale model depth and width. Increasing either model size or the noise ratio increases divergence probability.

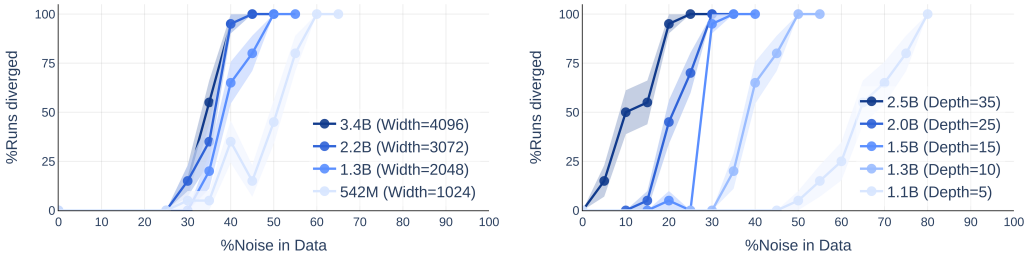

Figure 3: **Depth increases noise sensitivity more strongly than width. Left:** Scaling width at a fixed depth. **Right:** Scaling depth at a fixed width. Deeper models diverge far more frequently in the configurations studied.

the divergent runs, clean and noisy losses rise sharply at approximately the same time, showing that divergence also degrades predictions on clean tokens. See Appendix E for the legend definitions and further discussion of these loss curves. Thus, a single seed can make the same noisy-data setting appear either stable or unstable. Throughout the paper, we therefore measure the fraction of 20 seeds that diverge and study how this estimated divergence probability varies with different data and model configurations.

4.2 Larger and deeper models are more sensitive to noise

We first measure how divergence probability changes with model scale and noise ratio. For these experiments, we insert tokens sampled uniformly from the first five tokens in the tokenizer vocabulary, $|V_N| = 5$. We further examine this choice of noise vocabulary and corruption method in § 4.3.

In Figure 2, we jointly increase depth and width while keeping their ratio $\frac{\text{model_dim}}{\text{num_layers}} = 204.8$ fixed. We study four dense models ranging from 472M parameters (1024 dimensions, 5 layers) to 5.2B parameters (4096 dimensions, 20 layers). At a fixed noise ratio, larger models diverge more frequently. Increasing the noise ratio also increases divergence rates across all model sizes. To separate the effects of width and depth, the left panel of Figure 3 fixes depth at 10 layers and scales width from 1024 to 4096, increasing model size from 542M to 3.4B parameters. The right panel fixes width at 2048 and scales depth from 5 to 35 layers, increasing model size from 1.1B to 2.5B parameters. Increasing depth makes the model diverge more often under noisy data than increasing width. Most notably, the deepest 2.5B-parameter model diverges in 15% of runs even at a noise ratio of just 5%. These results show that larger models begin to diverge at lower noise ratios, so a data mixture that appears stable at a small scale can become unstable as the model grows.

We also use this scaling behavior throughout the paper to make our experiments computationally tractable. Since larger models are more susceptible to noise, we can study instability in much smaller models by deliberately pushing the noise ratio and noise structure conditions beyond what we would typically see in practice. We use these settings as a proxy for the instability that may arise in larger models at more realistic noise levels and noise settings. This lets us investigate the behavior without having to run every experiment at production scale.

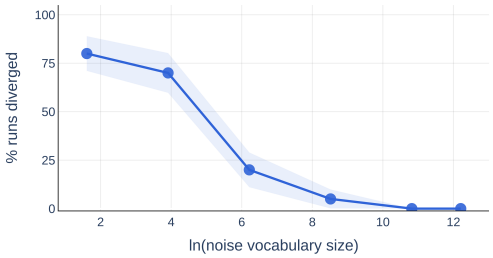

Figure 4: **Smaller noise vocabularies are more destabilizing.** Reducing the noise vocabulary size increases divergence probability. Shaded regions denote standard error.

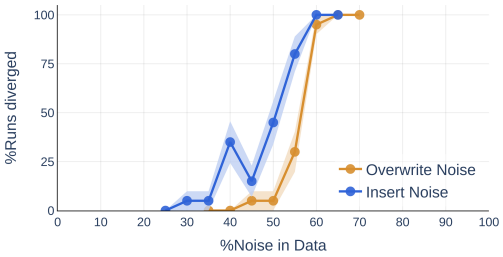

Figure 5: **Insertion is more destabilizing than overwriting.** For a fixed noise vocabulary of five tokens, inserting noise causes more divergence than overwriting clean tokens.

4.3 Effect of noise structure on divergence

While the previous scaling experiments hold noise structure fixed, we now examine how different noise types and corruption methods affect training loss divergence. Unless otherwise stated, all experiments in this subsection use a noise ratio of $\alpha = 55\%$ on a 540M-parameter dense model, following the architecture described in Table 2 and Table 3.

Smaller noise vocabularies increase divergence probability. In Figure 4, we vary the noise vocabulary size $k = |V_n|$ by restricting noise tokens to the first $k \in \{5e0, 5e1, 5e2, 5e3, 5e4\}$ tokens in the tokenizer vocabulary, and also using the full vocabulary of $2e5$. These experiments use insertion noise. We defer discussion of choices for the k tokens, as well as insertion versus overwriting noise, to later in this subsection. We observe that sampling random noise within a small vocabulary is more destabilizing than sampling from a broad vocabulary, even at the same overall noise ratio.

Inserting noise is more destabilizing than overwriting tokens. In Figure 5, we fix the noise vocabulary to the first five tokenizer tokens and compare both corruption methods across noise ratios. Insertion yields a higher divergence probability than overwriting. Together with the vocabulary-size result, this shows that the number of noisy tokens alone does not characterize the risk: both the noise vocabulary size and how noise enters the sequence matter.

Average token frequency does not explain much of the divergence variation. One may suspect that inserting common versus rare tokens as noise would affect training dynamics differently. To test this, we fix the noise vocabulary size to $|V_N| = 5$ and vary its *content* by choosing different sets of five unique tokens $\{t_1, \ldots, t_5\} \subset V$. Their average frequency in the clean corpus D_c spans several orders of magnitude, from tokens that never appear in D_c to very common tokens appearing approximately 12 million times on average. In Figure 6, the x-axis is the average clean-corpus frequency, $\frac{1}{5}\sum_{i=1}^{5} \text{frequency}(t_i, D_c)$, and the y-axis is the percentage of runs that diverge. We find that the actual noisy token content has minimal effect on divergence probability, with a near-zero Pearson correlation of 0.125.

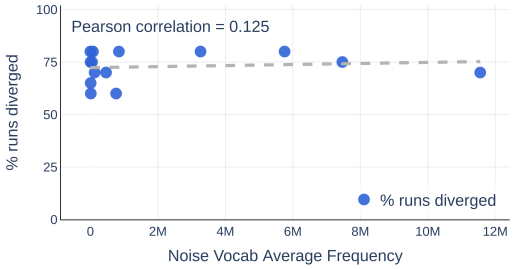

Figure 6: **Average noise-token frequency is weakly associated with divergence probability.** With $|V_N| = 5$, we vary the noise vocabulary and plot divergence rate against its average token frequency in the clean corpus. Pearson correlation is 0.125.

4.4 Dense and MoE models have similar noise sensitivity

We next investigate whether noise-induced divergence differs between dense and MoE models, as MoE architectures are increasingly common in modern LLMs. Prior work shows that MoE architectures can be less stable to train than dense models and often require additional stabilization

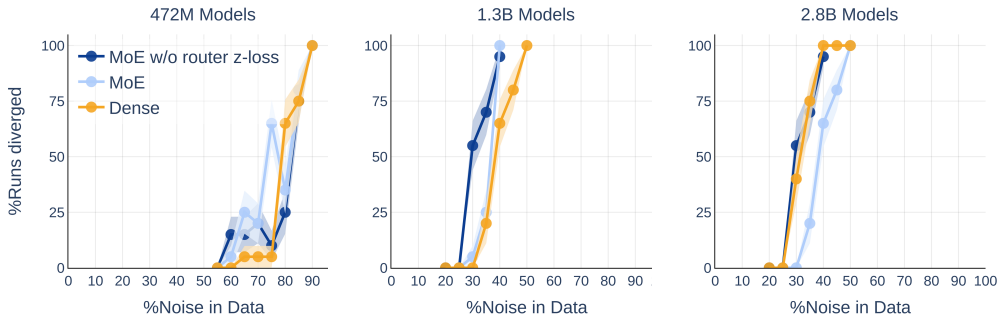

Figure 7: Divergence probability for dense and active-parameter-matched MoE models at three scales. MoE models diverge at similar rates to their dense counterparts across the tested scales.

techniques [32, 28]. With noisy data, a natural concern is whether expert specialization amplifies this instability. In particular, one can question if some experts may disproportionately receive noisy tokens, effectively behaving like a more unstable sub-network trained on highly noisy data.

We compare dense models with active-parameter–matched MoE models at three scales (472M, 1.3B, and 2.8B parameters). We evaluate MoEs both with and without router z-loss regularization [32], which is known to improve the router's training stability. As shown in Figure 7, divergence rates are similar across architectures at all three scales. This indicates that MoEs are not more sensitive than dense models to noisy data.

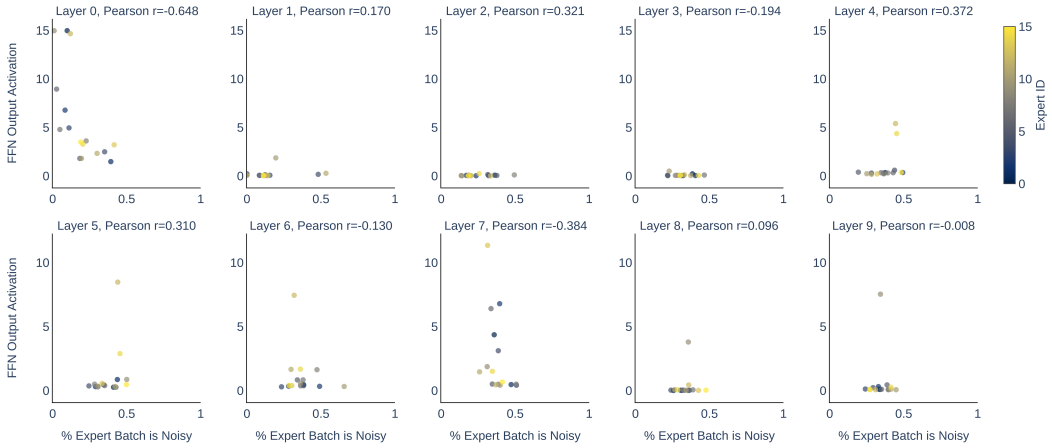

Figure 8: Each subplot corresponds to one of the ten layers in a 1.3B active-parameter MoE run, with each point representing one of the 16 expert2. The x-axis is the fraction of the expert batch that corresponds to noisy tokens, and the y-axis is the absolute mean of the FFN output activation. The average Pearson correlation across layers is -0.009, indicating little correlation between the two.

To understand this behavior, we analyze how noisy tokens are routed to experts in 1.3B active-parameter MoEs trained with 35% noise. We visualize token routing for a representative run that diverged[3]. Each subplot in Figure 8 corresponds to one of the ten MoE layers, with each point representing one of the 16 experts. The x-axis reports the fraction of noisy tokens in the expert's batch, and the y-axis reports the absolute mean of the expert FFN output activation. We examine activation magnitude because activation growth is often associated with training instability. We observe that, although experts receive varying proportions of noisy tokens, this fraction is uncorrelated with activation magnitude, with an average Pearson correlation of -0.009 across layers. This reinforces the conclusion that the MoE routing mechanism does not introduce additional sensitivity to noisy data.

[3]See leftmost panel in Figure 25 for the corresponding loss curve. We also refer readers to Appendix D for analysis on additional diverged and stable runs.

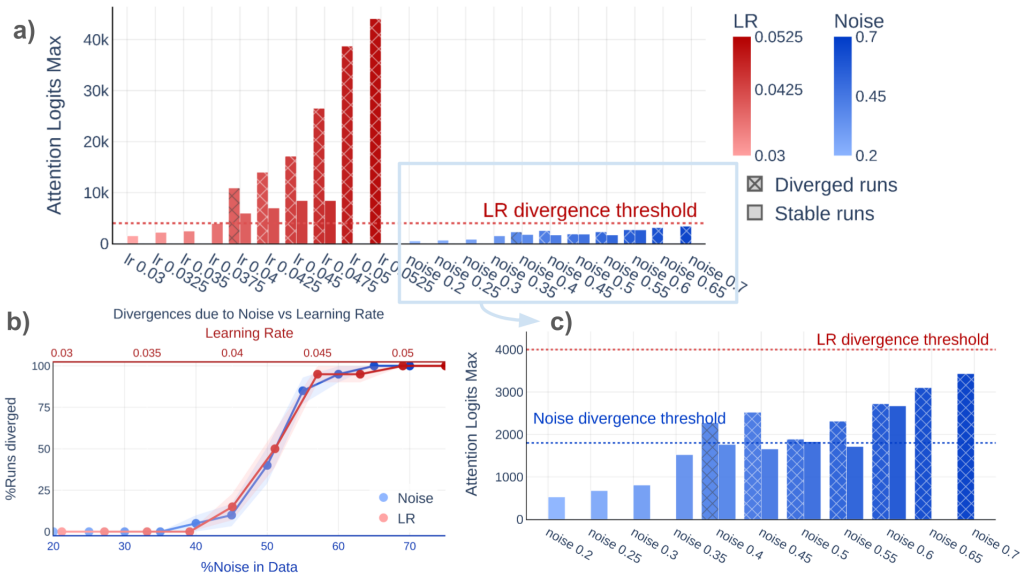

Figure 9: **Noisy data and overly high LR divergences exhibit different activation behavior.** We compare 540M dense models trained on clean data with varying high-LRs (red) versus models trained on a normal LR with varying noise ratios (blue). **(a)** Bars show the maximum attention logit at step 1000 averaged across seeds. Solid bars correspond to stable runs within the 20 seeds, while cross-hatched bars correspond to divergent runs within the 20 seeds. Missing bars indicate that no runs of that type occur for the configuration (i.e., lr=0.03 has no cross-hatched bars, meaning none of the 20 seeds for this configuration have diverged). Wortsman et al. identify divergence threshold for high-LR runs, shown by the dotted red line at 4000. **(b)** For comparability, LR and noise ratio ranges are selected to match divergence probabilities across the two settings. **(c)** A zoom-in on noisy data runs from (a) reveals a different and lower divergence threshold for noisy data settings at approximately 1800 (blue dotted lines).

4.5 Diagnosing noisy-data versus high-LR divergence

So far, we have established that noisy data can indeed cause pretraining loss divergence. However, when a training run diverges, how can we determine whether noisy data is the cause? Another common source of pretraining instability is an excessively high learning rate (LR). We therefore investigate whether noisy-data divergence can be distinguished from high-LR divergence. This distinction matters because the two failure modes require different interventions: while reducing the LR directly addresses high-LR divergence, it may suppress noisy-data divergence without resolving its root cause.

4.5.1 Examining maximum attention logits

In a transformer's self-attention layer, attention logits are computed as $z_{ij} = \frac{\langle q_i, k_j \rangle}{\sqrt{d_h}}$. Dehghani et al. [7] show that an excessively large maximum attention logit causes the softmax to collapse toward one-hot weights, destabilizing training. Building on this, Wortsman et al. [1] find that overly high LRs drive such logit growth and lead to loss divergence, and that the divergence can be predicted early in training when the maximum attention logit exceeds a certain model-size-invariant threshold.

We compare two sets of 540M dense-model experiments in Figure 9. High LRs runs (red) use LR $\in [0.03, 0.0525]$ on clean data while Noisy data runs (blue) use noise ratios $\alpha \in [20\%, 70\%]$ at a normal LR of 1.85×10^{-2}, with 20 seeds per configuration in both settings. Panel (b) shows that the selected ranges produce comparable divergence probabilities, allowing us to compare internal statistics at similar levels of instability. Panels (a) and (c) report maximum attention logits at step 1000 averaged separately over diverged seeds (cross-hatched bars) and stable seeds (solid bars). Missing bars indicate that no runs of that type occur for the configuration (i.e., the LR=0.03 configuration does not have a cross-hatched bar, meaning none of the 20 seeds for this configuration have diverged).

We chose to examine the activations at step 1000 because the earliest divergence occurs shortly after this point. See Appendix B for other model sizes, including MoE models. We make two observations:

High LR divergences produce significantly larger maximum attention logits. In Figure 9(a), we first reproduce the findings of Wortsman et al. [1]: across model sizes (see Appendix B, including MoE architectures), high LR configurations lead to a non-zero divergence probability once the maximum attention logit exceeds a threshold of 4000[4]. This threshold should be interpreted at the *configuration level*: if the maximum attention logits exceed this threshold value, then the configuration is likely unstable, i.e., at least one of the 20 seeds in the configuration diverges. Within an unstable configuration, stable runs may or may not exceed the threshold, but diverged runs do so more consistently. Beyond reproducing this result, we find that high-LR runs have significantly larger maximum attention logits than noisy data runs; even at the highest noise ratios where noisy runs diverge in 100% of seeds, the noisy data runs' maximum attention logits never exceed the high LR divergence threshold of 4000.

Noisy data divergences occur at their own threshold, around 1800. In Figure 9(c), we see that within the noisy data setting, configurations whose maximum attention logits exceed ~ 1800 exhibit a non-zero probability of divergence. As shown in Appendix B, the same numerical thresholds, ~ 4000 for high-LR runs and ~ 1800 for noisy data runs, also apply across different model sizes, including MoE models.

Hence, the same external symptom, which is similar loss divergence rate, can come from different internal dynamics. This provides a useful diagnostic signal: when a run diverges, its activation patterns can help distinguish between high-LR and data-induced divergences. In our experiments, a maximum attention logit exceeding approximately 4000 early in training is characteristic of high-LR divergence, whereas runs that diverge without reaching this regime are more consistent with noisy-data-induced divergence.

4.5.2 Examining global parameter norms

Beyond differences in maximum attention logits, we observe additional distinctions between divergences induced by high LRs and those caused by noisy data. As shown in Figure 10, even at comparable divergence rates, runs that diverge due to noisy data exhibit smaller global parameter RMS norms than those diverging due to high LRs. Moreover, increasing the LR leads to larger global parameter norms, whereas increasing the noise ratio does not. See Appendix B for analysis on other activation statistics.

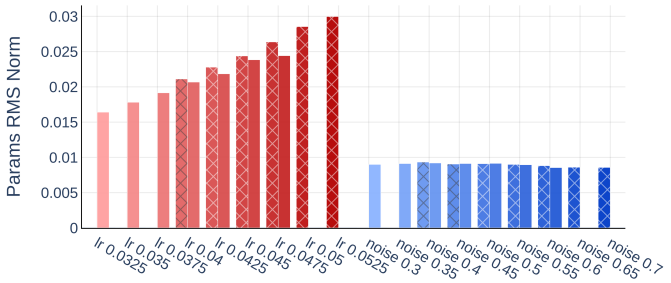

Figure 10: Global parameter RMS norms for 540M dense models trained on clean data with varying high LRs (red) versus with a fixed small LR on varying noisy data (blue). Divergences induced by noisy data exhibit significantly smaller parameter norms than those induced by high LRs. Increasing the LR also leads to growth in global parameter norms, whereas increasing the noise ratio does not.

Together, the maximum attention-logit and parameter-norm results provide diagnostic evidence. If a run diverges with abnormally large magnitude global parameter norm together with rapidly growing attention logits, that points toward a LR or optimization issue. On the other hand, if the global parameter norms and activation patterns more closely resemble those observed in our noisy-data runs, the data itself becomes a stronger candidate for the underlying cause.

[4]In Wortsman et al. [1], the reported threshold is 10^4 at step 2000. The lower threshold here is expected, as we measure activations earlier due to a shorter warm-up and fewer total training steps.

4.6 Interventions for noisy-data divergence

The diagnostic results identify attention-logit growth as a useful signal of noisy-data induced instabilities. To motivate an intervention, we decompose the attention logit as $z_{ij} = \|q_i\|\|k_j\| \cos\theta_{ij}/\sqrt{d_h}$. The analysis in Figure 23 associates the observed logit growth with increasing query and key norms rather than increasing cosine similarity. This motivates QK-layernorm [7, 33–35]as a way to control logit growth. We compare this stabilization approach with directly removing noise from the training data.

We train 540M dense models with noise ratios ranging from 30% to 70%. Without QK-layernorm, no runs diverge at 30% noise and all 20 seeds diverge at 70%, with intermediate divergence rates between these settings (Figure 9b). To assess the performance cost of noise separately from divergence, Figure 11 reports loss on clean tokens only, averaged only over stable runs. In the figure, we only show noise ratio $\alpha = 0\%$, 35%, and 55% for clarity, but the same trend holds across noise ratios. We make three observations:

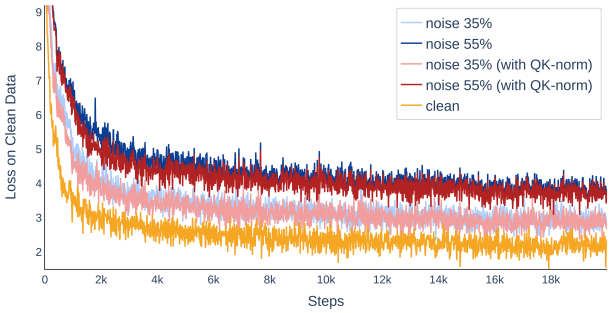

Figure 11: Average loss for stable runs of a 540M-parameter dense model, evaluated on clean tokens only. Results are averaged over seeds that do not diverge.

Cleaning the data gives the best performance. Even among runs that remain stable, increasing the noise ratio degrades performance on clean tokens. Training on clean data yields the lowest loss. Thus, removing noise is the best solution when possible.

QK-layernorm stabilizes training when cleaning is infeasible. It substantially reduces maximum attention logits across noisy settings (see Figure 24) and eliminates all divergence at every tested noise ratio. Even at 70% noise, where all baseline runs diverge, all 20 seeds remain stable with QK-ayernorm. Among stable runs, QK-layernorm also yields slightly lower training loss than training without it.

However, when data cleaning is possible, QK-LayerNorm should not be applied blindly. There are three reasons for this. First, cleaning the data yields the best model performance. Second, QK-LayerNorm suppresses the growing activations associated with noisy data, potentially masking a useful signal of poor data quality. Third, prior work has identified undesirable side effects of QK-LayerNorm, including degraded long-context capabilities [36, 37]. Therefore, data cleaning should remain the preferred intervention when feasible, while QK-LayerNorm offers an effective alternative when cleaning is not possible.

5 Conclusion

We present a large-scale, controlled study demonstrating that uniform random noise can cause LLM pretraining loss divergence and how it does so. We show that divergence becomes more likely with higher noise ratios and larger models, and that noise structure also plays an important role in training stability. We further find that noise-induced divergence exhibits distinct activation and parameter patterns from high-learning-rate divergence, providing diagnostic signals for identifying the likely cause of training failures. Finally, we show that data cleaning is the preferred intervention, yielding both stable training and the best model performance. However, when data cleaning is not feasible, we show applying QK-LayerNorm offers an effective alternative for stabilizing training, although it does not recover the performance of clean-data training. Together, our findings establish data quality as an important factor in LLM pretraining stability, beyond its effects on model performance, and provide practical guidance for diagnosing and mitigating training divergence.

Acknowledgments

Authors thank Abhishek Kadian and Punit Singh Koura for helpful pointers regarding data loader implementations. The authors are grateful for Meng Zhang, Hanwen Zha, Vinay Rao, and Haroun Habeeb for insightful research discussions. QZ would also like to thank the interns in the legacy GenAI team for brainstorming together, exchanging ideas, and providing feedback, including Wannan (Winnie) Yang, Rachit Bansal, Julian Coda-Forno, Sai Surya Duvvuri, Sriyash Poddar, Arushi Rai, Rishabh Tiwari, and Devvrit Khatri. QZ thanks Chris Lu for helpful discussions on the project.

References

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

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[3] Aaron Adcock, Aayushi Srivastava, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pande, Abhinav Pandey, Abhinav Sharma, Abhishek Kadian, Abhishek Kumawat, Adam Kelsey, et al. The llama 4 herd: Architecture, training, evaluation, and deployment notes. *arXiv preprint arXiv:2601.11659*, 2026.

[4] 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. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

[5] 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.

[6] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

[7] 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*, pages 7480–7512. PMLR, 2023.

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

[9] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[10] Maxim Fishman, Brian Chmiel, Ron Banner, and Daniel Soudry. Scaling fp8 training to trillion-token llms. *arXiv preprint arXiv:2409.12517*, 2024.

[11] Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, et al. Fp8-lm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*, 2023.

[12] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, 2022.

[13] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.

[14] Jinghan Ru, Yuxin Xie, Xianwei Zhuang, Yuguo Yin, Zhihui Guo, Zhiming Liu, Qianli Ren, and Yuexian Zou. Do we really have to filter out random noise in pre-training data for language models? *arXiv preprint arXiv:2502.06604*, 2025.

[15] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.

[16] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *International conference on machine learning*, pages 1062–1070. PMLR, 2019.

[17] Aritra Ghosh, Himanshu Kumar, and P Shanti Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[18] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[19] Hao Chen, Jindong Wang, Ankit Shah, Ran Tao, Hongxin Wei, Xing Xie, Masashi Sugiyama, and Bhiksha Raj. Understanding and mitigating the label noise in pre-training on downstream tasks. *arXiv preprint arXiv:2309.17002*, 2023.

[20] Mo Zhou, Tianyi Liu, Yan Li, Dachao Lin, Enlu Zhou, and Tuo Zhao. Toward understanding the importance of noise in training neural networks. In *International Conference on Machine Learning*, pages 7594–7602. PMLR, 2019.

[21] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[22] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.

[23] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

[24] 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.

[25] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[26] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[27] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[28] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23 (120):1–39, 2022.

[29] Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

[30] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[31] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

[32] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

[33] 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.

[34] Gemma Team, 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, Robert 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, Alex Feng, 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-Plucińska, 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, Noveen Sachdeva, 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, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL `https://arxiv.org/abs/2503.19786`.

[35] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

[36] Bowen Yang, Bharat Venkitesh, Dwarak Talupuru, Hangyu Lin, David Cairuz, Phil Blunsom, and Acyr Locatelli. Rope to nope and back again: A new hybrid attention strategy. *arXiv preprint arXiv:2501.18795*, 2025.

[37] Amanda Bertsch, Luca Soldaini, Matthew R Gormley, Graham Neubig, Hannaneh Hajishirzi, Kyle Lo, and Dirk Groeneveld. Cracks in the foundation: Seemingly minor architectural choices impact long context extension. *arXiv preprint arXiv:2608.10296*, 2026.

[38] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

A Model architecture details

Table 2 reports the default model configuration that is shared across all experiments and model sizes.

FFN Dim	Model Dim $\times$ 4
Sequence Length	8192
Head Dim	128
Key/Value Heads	8
Activation Function	SwiGLU
Tokenizer Vocabulary Size	200,000
Positional Embeddings	RoPE ($\theta = 500,000$)

Table 2: Summary of architecture hyperparameters shared across all models trained in this paper.

Table 3 provides the additional architectural variations used in all experiments. For MoE models, the " # Parameters" column reports the number of active parameters.

	# Parameters	Model Dim	# Layers	# Query Heads
Scaling Depth	1.1B	2048	5	16
	1.3B	2048	10	16
	1.5B	2048	15	16
	2.0B	2048	25	16
	2.5B	2048	35	16
Scaling Width	540M	1024	10	8
	1.3B	2048	10	16
	2.2B	3072	10	24
	3.4B	4096	10	32
Scaling Both	480M	1024	5	8
	1.3B	2048	10	16
	2.8B	3072	15	24
	5.2B	4096	20	32

Table 3: Model configurations for experiments scaling depth, width, and scaling both depth and width.

A.1 MoE architecture details

For Mixture of Experts [MoE; 27, 28] experiments, we use dropless MoEs[29, 30] with 16 feed-forward network (FFN) experts and token-choice top-2 routing. Given a token representation x_t, the router produces logits $r_{t,e}$ over experts $e \in \{1, \ldots, 16\}$. We select the top-2 experts

$$S_t = \mathrm{TopK}(\mathrm{Router}(x_t), k = 2),$$

We compute the mixture weights $p_{t,e}$ by applying a softmax over the selected experts S_t:

$$p_{t,e} = \mathrm{softmax}(r_{t,e}) \quad \text{for } e \in S_t$$

The final MoE layer output is the top-2 experts weighted by the mixture weights

$$y_t = \sum_{e \in S_t} p_{t,e}\, \mathrm{FFN}_e(x_t).$$

To maintain a balanced load across experts, we adopt loss-free balancing [38, 30] with an expert-bias coefficient of $1\mathrm{e}{-3}$. For some experiments, we additionally apply the router z-loss [32], using an auxiliary loss coefficient of $1\mathrm{e}{-3}$.

For fair comparison with dense models, our top-2 MoE models are active-parameter matched by scaling the FFN dimension by 0.5. All remaining architectural components follow the same configuration as the dense models.

B Additional model parameter and activation analysis

We provide additional analyses of model parameters and activations extending § 4.5. Specifically, we study three dense models with parameter counts ranging from 540M to 2.8B, as well as an MoE model with 1.3B active parameters (3.7B total parameters). We find that the conclusions of § 4.5 hold consistently across model sizes and for both dense and MoE architectures.

Beyond differences in maximum attention logits, we observe additional distinctions between divergences induced by high LRs and those caused by noisy data. As shown in Figure 12, even at comparable divergence rates, runs that diverge due to noisy data exhibit smaller parameter norms than those diverging due to high LRs. Moreover, increasing the LR leads to larger parameter norms, whereas increasing the noise ratio does not.

B.1 540M dense model

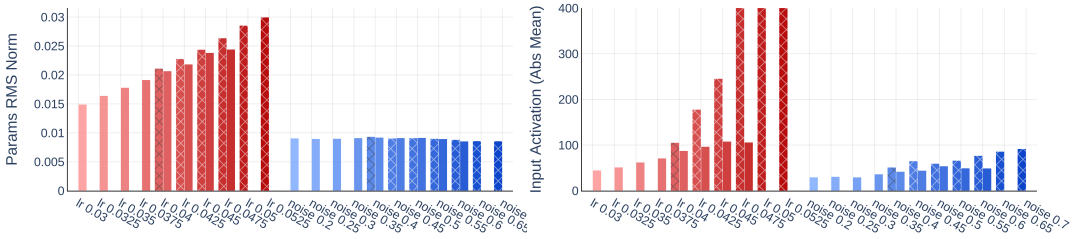

Figure 12: 540M dense model run statistics. **Left:** Parameter RMS norm. **Right:** Absolute mean of activation input to transformer layers.

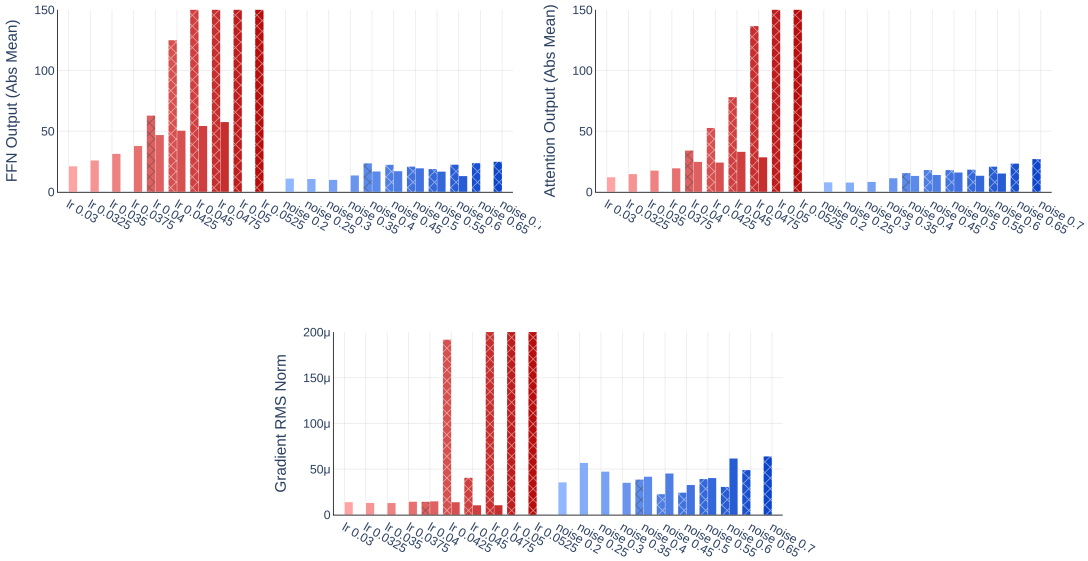

Figure 13: 540M dense model run statistics. **Top left**: Absolute mean of pre-residual FFN output. **Top right**: Absolute mean of pre-residual attention output. **Bottom**: Gradient RMS norm.

B.2 1.5B dense model

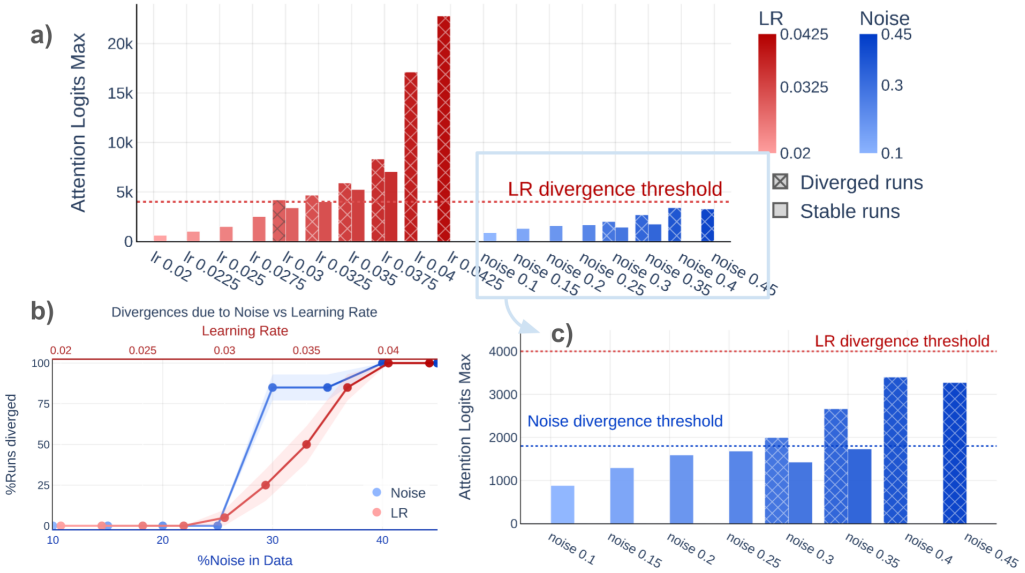

Figure 14: Examining the maximum attention logits for 1.5B Dense models .**(a)** Wortsman et al. identify a model-size-invariant divergence threshold for high LR runs, shown by the dotted red line at 4000. **(b)** For comparability, LR and noise ratio ranges are selected to match divergence probabilities across the two settings. **(c)** A zoom-in on noisy data runs from (a) reveals a different and lower divergence threshold for noisy data settings at approximately 1800 (see the blue dotted lines). This noisy data divergence threshold also holds across different model sizes and MoE architectures.

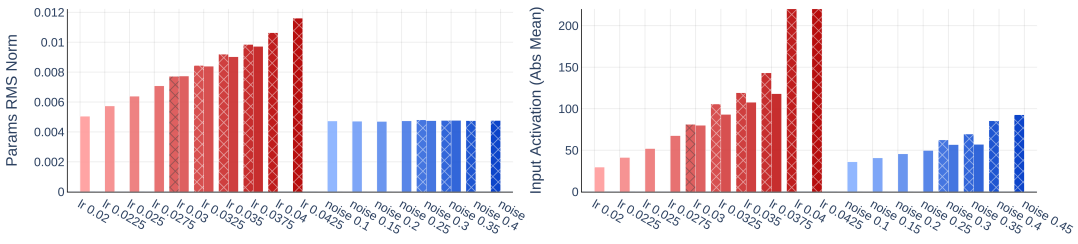

Figure 15: 1.5B dense model run statistics. **Left:** Parameter RMS norm. **Right:** Absolute mean of activation input to transformer layers.

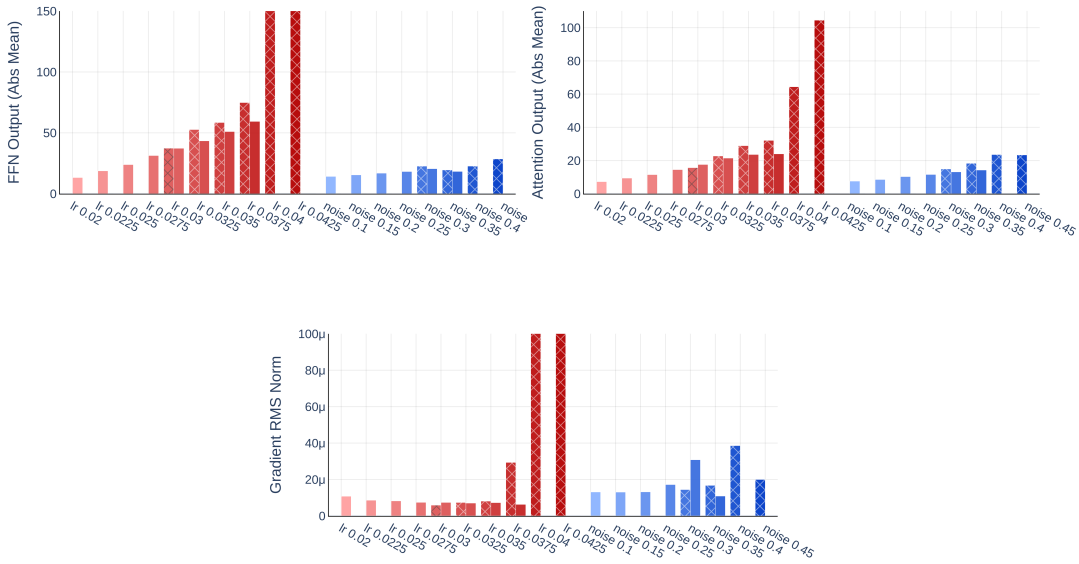

Figure 16: 1.5B dense model run statistics. **Top left**: Absolute mean of pre-residual FFN output. **Top right**: Absolute mean of pre-residual attention output. **Bottom**: grad RMS norm.

B.3 2.8B dense Model

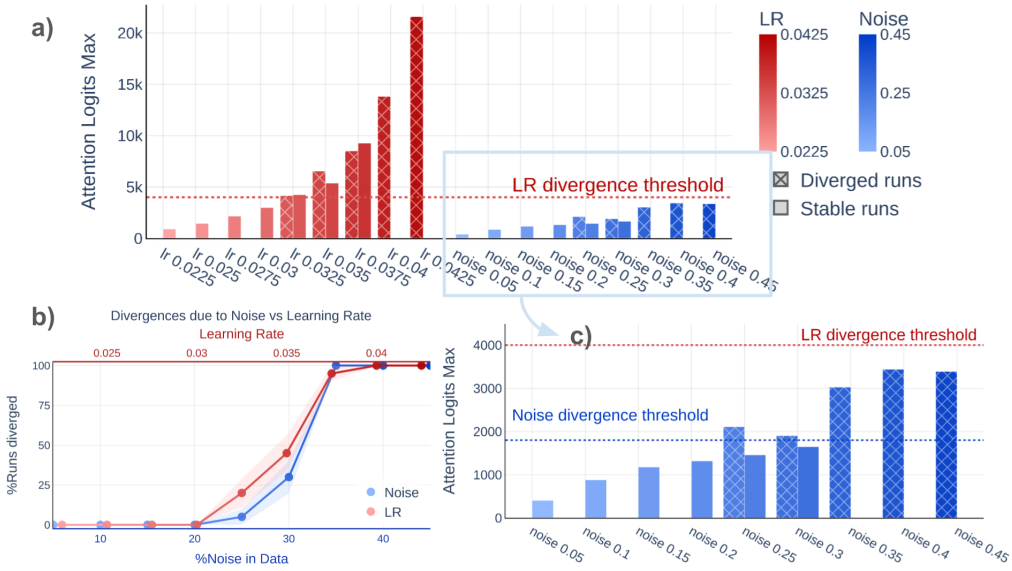

Figure 17: Examining the maximum attention logits for 2.8B dense models .**(a)** Wortsman et al. identify a model-size-invariant divergence threshold for high LR runs, shown by the dotted red line at 4000. **(b)** For comparability, LR and noise ratio ranges are selected to match divergence probabilities across the two settings. **(c)** A zoom-in on noisy data runs from (a) reveals a different and lower divergence threshold for noisy data settings at approximately 1800 (see the blue dotted lines). This noisy data divergence threshold also holds across different model sizes and MoE architectures.

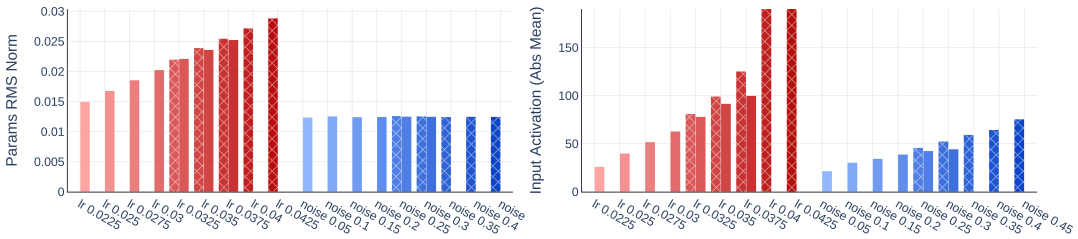

Figure 18: 2.8B dense model run statistics. **Left:** Parameter RMS norm. **Right:** Absolute mean of activation input to transformer layers.

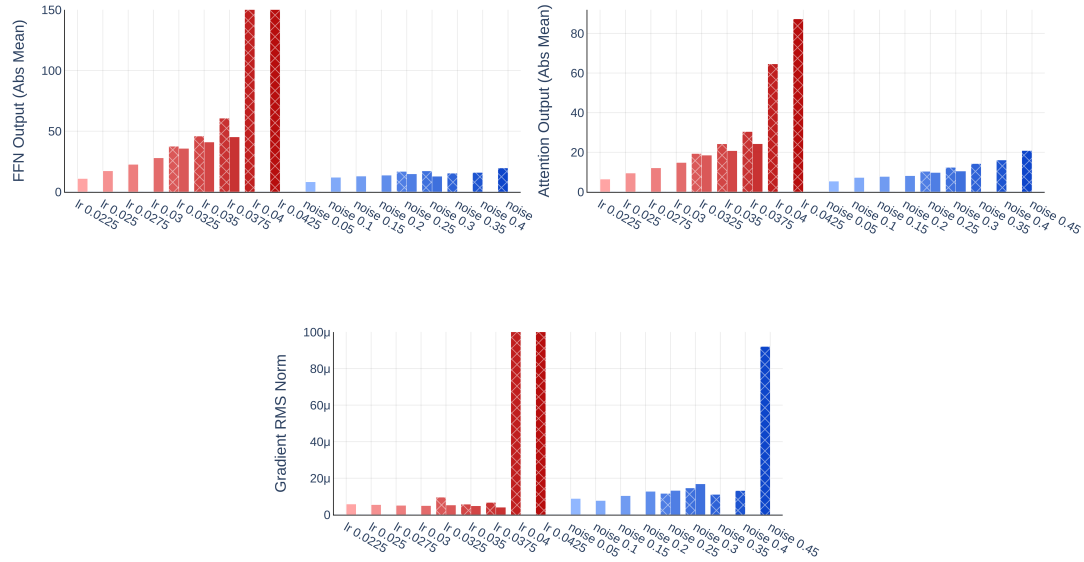

Figure 19: 2.8B dense model run statistics. **Top left**: Absolute mean of pre-residual FFN output. **Top right**: Absolute mean of pre-residual attention output. **Bottom**: Gradient RMS norm.

B.4 1.3B active parameter MoE

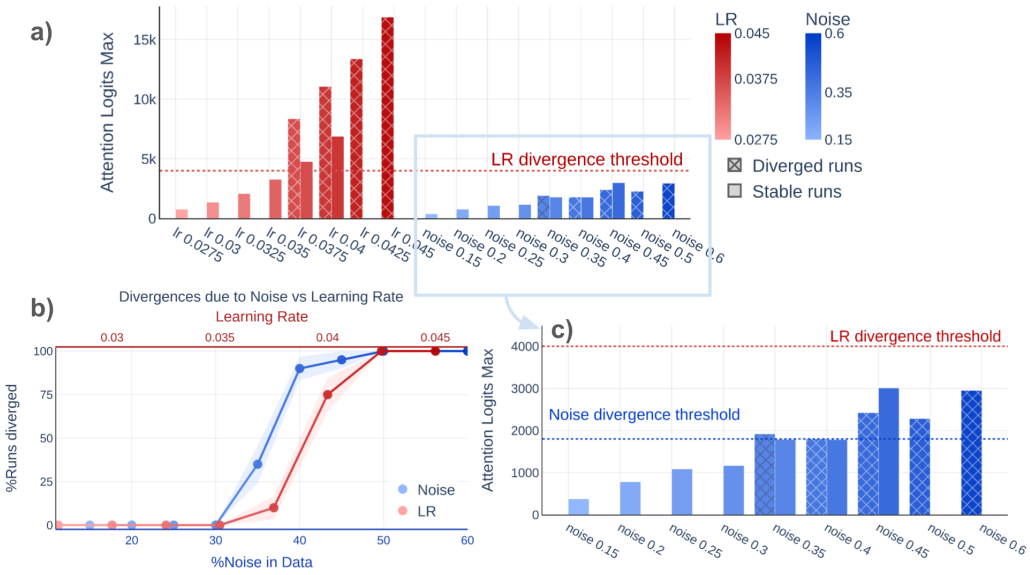

Figure 20: Examining the maximum attention logits for 1.3B active parameter MoE models .**(a)** Wortsman et al. identify a model-size-invariant divergence threshold for high LR runs, shown by the dotted red line at 4000. **(b)** For comparability, LR and noise ratio ranges are selected to match divergence probabilities across the two settings. **(c)** A zoom-in on noisy data runs from (a) reveals a different and lower divergence threshold for noisy data settings at approximately 1800 (see the blue dotted lines). This noisy data divergence threshold also holds across different model sizes and MoE architectures.

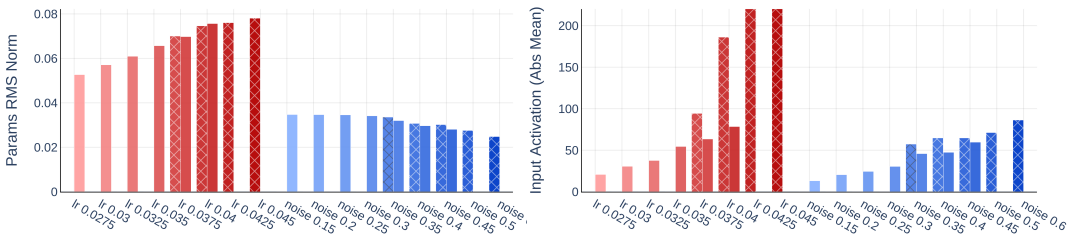

Figure 21: 1.3B MoE model run statistics. **Left:** Parameter RMS norm. **Right:** Absolute mean of activation input to transformer layers.

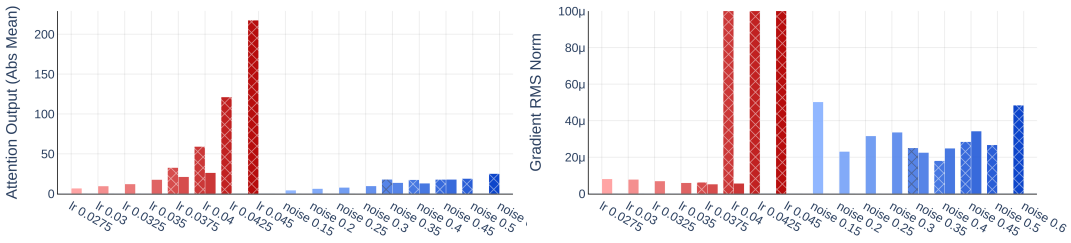

Figure 22: 1.3B MoE model run statistics. **left**: Absolute mean of pre-residual attention output. **Right**: Gradient RMS norm.

C QK-layernorm interventions

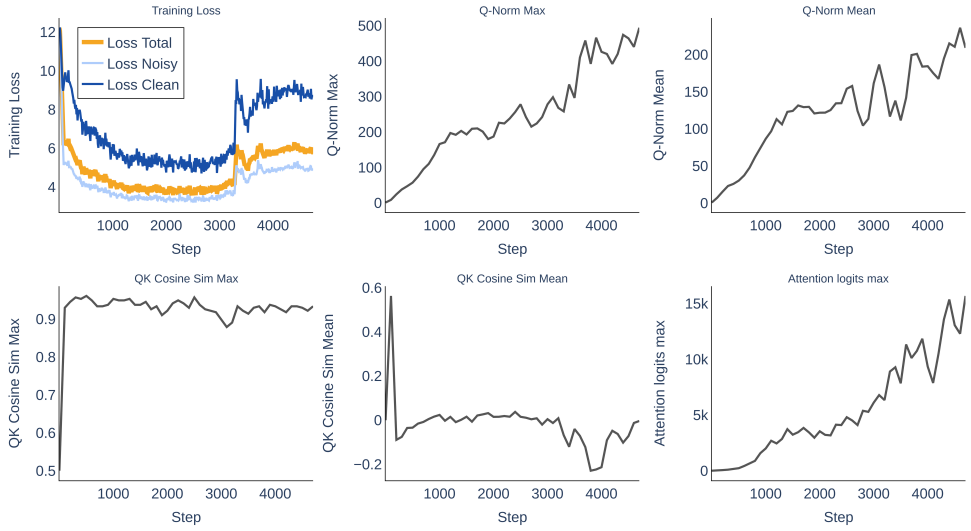

Figure 23: We show that growing Q-norms and K-norms are the cause of growing maximum attention logits, not the growing cosine similarity between the query and the key.

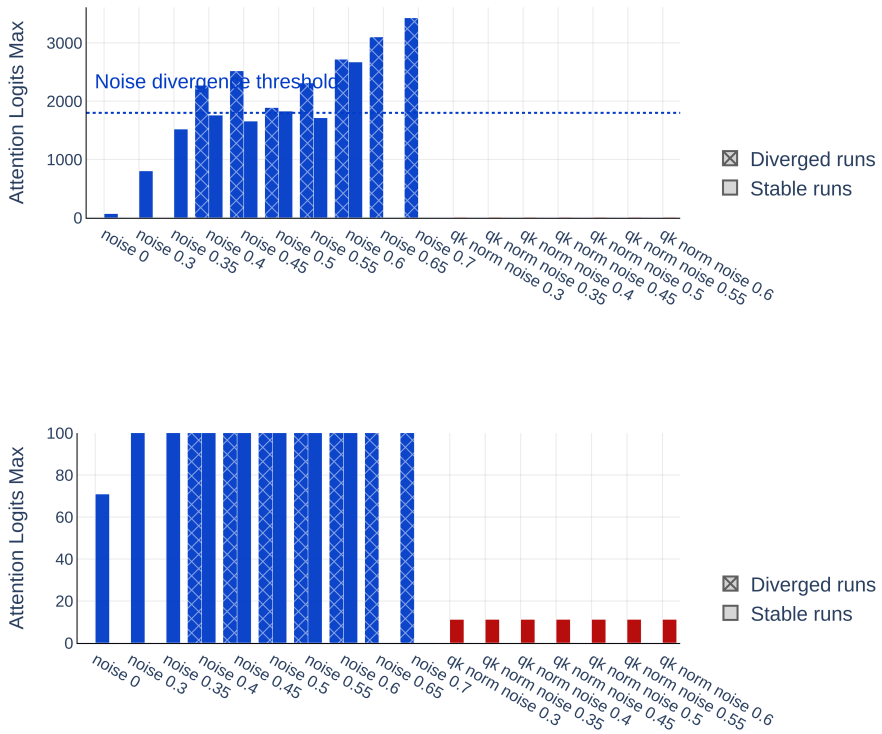

Figure 24: Maximum attention logits without (blue) versus with QK-layernorm (red). For clarity, we show a zoomed-in view of the plot on the bottom.

D MoE router analysis

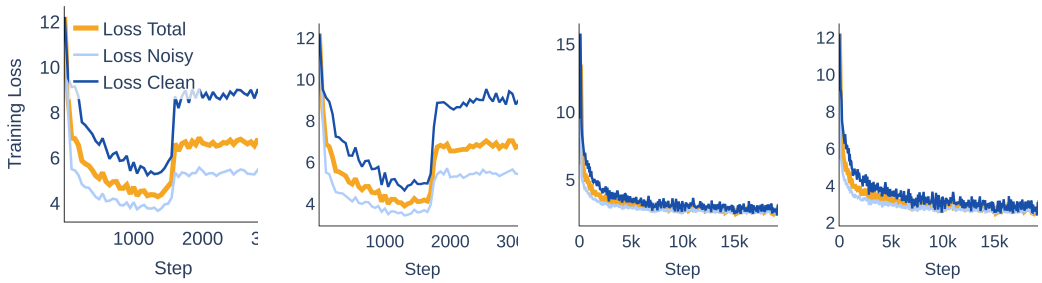

Figure 25: Training loss curves of 1.3B MoE runs with 35 % noise and applying router z-loss. The only difference among the four runs is the random seed. We see that the first two runs diverge, while the latter two runs remain stable.

Below, we show the noisy token assignment for each of the four runs in Figure 25. Although experts receive varying proportions of noisy tokens, this fraction is uncorrelated with experts' output activation magnitude, with an average Pearson correlation of near-zero across layers. This reinforces the conclusion that the MoE routing mechanism does not introduce additional sensitivity to noisy data.

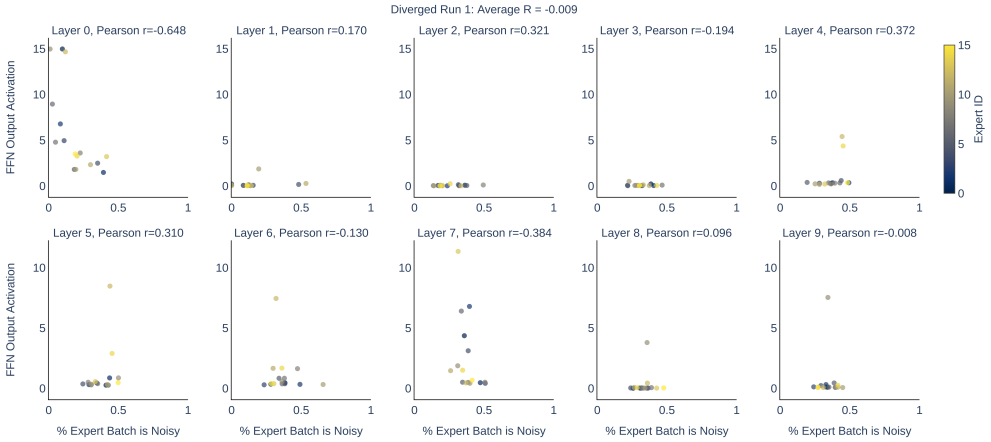

Figure 26: Noisy token assignment has very little correlation to FFN activation.

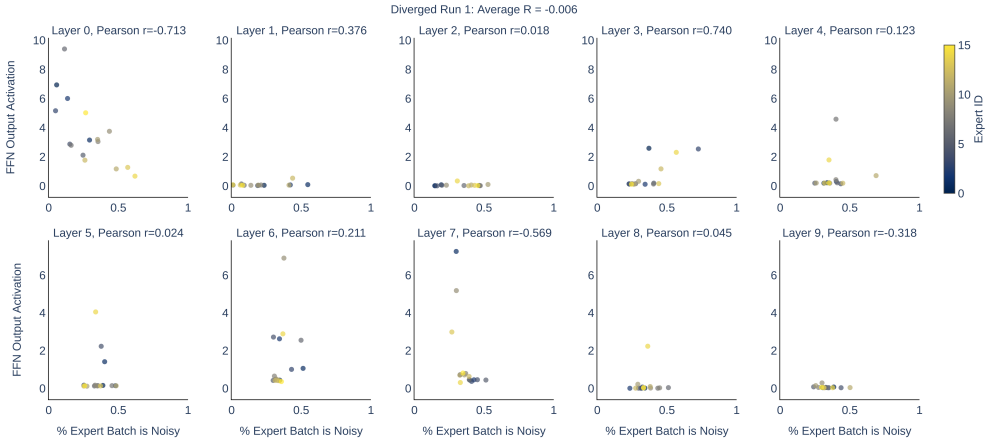

Figure 27: Noisy token assignment has very little correlation to FFN activation.

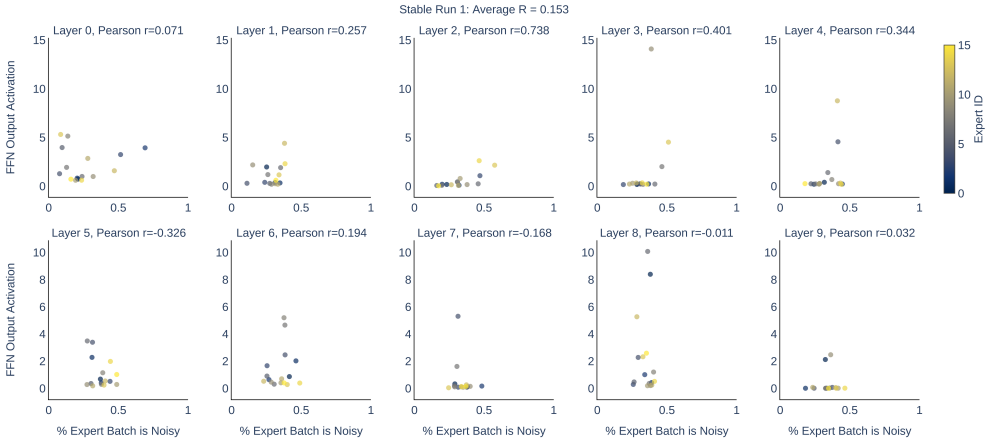

Figure 28: Noisy token assignment has very little correlation to FFN activation.

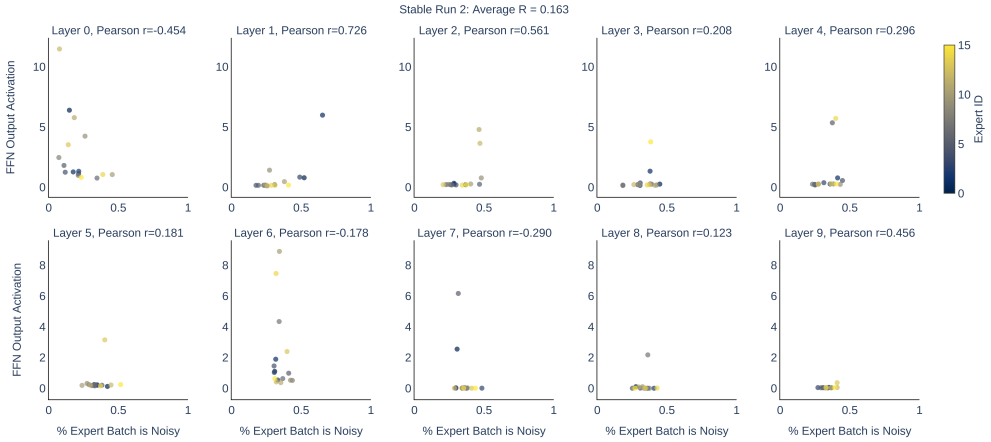

Figure 29: Noisy token assignment has very little correlation to FFN activation.

E Loss definitions and training dynamics in Figure 1

In Figure 1, "Loss Total" (orange) averages over all tokens; "Loss Clean" (dark blue) averages over tokens for which the target token is clean; "Loss Noisy" (light blue) averages over tokens for which the target token is noise. Early in training, clean loss is higher because learning the clean data distribution is harder than learning to predict uniformly over the restricted noise vocabulary. Noisy loss then plateaus near its random-guessing optimum, while clean loss continues to decrease and becomes lower.

F Limitations and future work

Our study focuses on synthetic uniform random noise, which allows us to isolate its effect on pretraining stability under controlled conditions. However, real-world pretraining data contains more diverse forms of noise, whose effects remain unexplored. Several experiments also use intentionally high noise ratios and small noise vocabularies to induce divergence reliably in smaller models. While our results suggest that larger models may exhibit similar instabilities under less extreme conditions, validating this at frontier scale with naturally occurring noise remains future work. Finally, although we identify sepcific noise sturcture causes more divergence and that noise-induced divergence are associated with distinct activation and parameter patterns, a complete mechanistic explanation remains open for future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims matches empirical results from § 4

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: We mention in the paper things that are outside the scope of the paper due to computational constraints, such as ablating other stabilization techniques and other forms of low-quality data.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes],

   Justification: We describe experimental setups in details in § 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Due to IP constraints at the organization this work was done that, we do not to provide open access.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide these detaisl in § 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: We provide standard error bars on all figures.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: We describe our compute resources in § 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: We have reviewed the ethics guidelines, and our research conforms.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A].

    Justification: Our work advances the understanding of LLM pretraining by empirically characterizing how noisy data can cause training loss divergence, and our findings help improve the reliability and efficiency of LLM training. While increased training stability can facilitate broader deployment of LLMs, the societal impacts depend on downstream applications. We do not identify specific new ethical concerns arising directly from this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: We do not release data or models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: We cite all the code, data, and models accordingly in our paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: We do not introduce any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: We used LLMs solely to polish the writing for spotting grammar mistakes and improving readability. The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.