Grokking or Glitching? How Low-Precision Drives Slingshot Loss Spikes

Liu Hanqing LIU.HANQING.PHY@GMAIL.COM
Tsinghua University & The University of Tokyo

Jianjun Cao CAOJJ25@MAILS.TSINGHUA.EDU.CN
Tsinghua University

Yuanze Li LIYUANZE23@MAILS.TSINGHUA.EDU.CN
Tsinghua University

Zijian Zhou ZHOUZJ23@MAILS.TSINGHUA.EDU.CN
Tsinghua University

Abstract

Deep neural networks exhibit periodic loss spikes during unregularized long-term training, a phenomenon known as the "Slingshot Mechanism" [40]. Existing work usually attributes this to intrinsic optimization dynamics, but its triggering mechanism remains unclear. This paper proves that this phenomenon is a result of floating-point arithmetic precision limits. We show that finite-precision errors in cross-entropy computation can break the zero-sum constraint of gradients across classes and introduce a systematic drift in the parameter update of the classifier layer. This drift forms a positive feedback loop with the feature mean, which we call *Numerical Feature Inflation* ($\mathcal{NFI}$). Our results reinterpret Slingshot as a numerical dynamic of finite-precision training and provide a testable explanation for the emergence of periodic loss spikes in late-stage unregularized training.

1. Introduction

Loss spikes are a persistent puzzle in neural network training. One representative example is the *Slingshot Mechanism*, first observed in the study of grokking [30] under no explicit regularization. In such settings, training is often accompanied by periodic instabilities: the norm of the last-layer parameters grows rapidly, and is followed by an abrupt training loss spike.

Existing work has mainly interpreted Slingshot as an intrinsic optimization phenomenon. For example, Thilak et al. [40] related it to the Edge of Stability [5]. Nanda et al. [25] suggested that the effect may arise from the interaction between gradients of different scales and adaptive optimizer dynamics. *In contrast*, we show that Slingshot is not primarily caused by the optimization dynamics. Instead, it is triggered by finite-precision arithmetic in the computation of the loss.

After long training, the model enters a high-confidence regime: it reaches perfect training accuracy, and for each training sample, the correct-class logit becomes much larger than the other logits. Cross-entropy loss converts logits into probabilities through the softmax function. Prieto et al. [32] showed that when the gap between the largest logit z_m and the other logits exceeds a threshold determined by floating-point precision, absorption error occurs. The computed value is no longer exact. As a result, during backpropagation, the gradient of the loss with respect to the correct-class logit, $\partial L/\partial z_m$, is rounded exactly to zero. They call this phenomenon *Softmax Collapse* (SC).

We show that SC has a second effect that directly drives Slingshot. In exact arithmetic, the gradients on the rows $\boldsymbol{W}_k$ of the final classifier satisfy a zero-sum constraint across classes. Therefore, the global classifier mean $\boldsymbol{W}_G = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{W}_k$ remains unchanged under gradient updates. However, SC breaks this zero-sum constraint. We prove that this introduces a drift of $\boldsymbol{W}_G$ in the direction of $-\boldsymbol{\mu}_G$, where $\boldsymbol{\mu}_G = \frac{1}{B}\sum_{k,i} \boldsymbol{h}_{k,i}$ is the mean of the penultimate-layer features in a batch. We further prove that, once $\boldsymbol{W}_G$ becomes nonzero, the feature mean $\boldsymbol{\mu}_G$ also receives a drift in the direction of $-\boldsymbol{W}_G$. These two effects form a positive feedback loop: $\boldsymbol{W}_G$ and $\boldsymbol{\mu}_G$ become anti-parallel, and their norms grow exponentially. We call this process *Numerical Feature Inflation* ($\mathcal{NFI}$).

$\mathcal{NFI}$ explains how Slingshot loss spikes are triggered. As $\boldsymbol{W}_G$ and $\boldsymbol{\mu}_G$ grow, the sample-level margin $z_m - \max_{k\neq m} z_k$ can become fragile for some samples and eventually fall below the SC threshold. The correct-class gradient then reappears abruptly. Before this moment, the correct-class gradient has been zero and the incorrect-class gradients have been extremely small, so Adam can amplify the effective learning rate to a large value. When the correct-class gradient suddenly changes from zero to a finite value, this large effective learning rate produces a large parameter update, causing the loss spike.

Our contributions are summarized as follows:

(a) We provide a theoretical explanation of the Slingshot Mechanism based on finite-precision cross-entropy computation.

(b) We identify $\mathcal{NFI}$ as the mechanism that triggers Slingshot loss spikes.

(c) We propose and validate practical interventions, which suppress $\mathcal{NFI}$-induced instability.

2. The Mechanics of *Numerical Feature Inflation*

In this section, we derive the theoretical mechanism behind Slingshot. As shown in Figure 1(*a*)subfigure, our results indicate that the Slingshot mechanism is fundamentally a numerical precision artifact. Even when model parameters are stored in `float32`, casting the output logits to `float64` solely during the loss computation is sufficient to eliminate the Slingshot effect. As analyzed by Prieto et al. [32], this instability arises from absorption errors in floating-point arithmetic.

2.1. Preliminaries

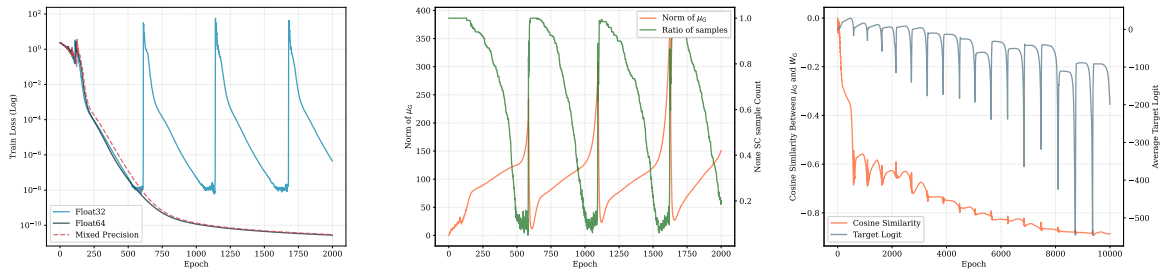
According to the IEEE 754 standard [1], a floating-point number consists of 1 sign bit s, E exponent bits, and p mantissa bits.

Definition 1 (Absorption Error) *Consider the addition of two non-zero floating-point numbers a and b (where $|a| \geq |b|$). The operation requires exponent alignment. If the ratio satisfies $\frac{|b|}{|a|} < 2^{-(p-1)}$, the smaller value b cannot be represented within the mantissa precision after alignment. This results in the absorption error, defined as $a + b = a$.*

For `float32`, the mantissa precision is $p = 24$ bits. Consequently, the critical threshold is $2^{-(24-1)} = 2^{-23} \approx 1.19 \times 10^{-7}$.

Absorption errors occur in Softmax Cross-Entropy loss calculation. PyTorch [29] implements the Softmax CE loss using the Log-Sum-Exp trick for numerical stability. The denominator is stored as:

$$Z = \log\left(\sum_k \exp(z_k)\right) = z_m + \log\left(\sum_k \exp(z_k - z_m)\right) \tag{1}$$



(*a*) (*b*) (*c*)

Figure 1: **Precision-induced $\mathcal{NFI}$ dynamics.** (a) Slingshot loss spikes vanish when training is performed in float64 or when only the logits/loss computation is cast to float64. (b) Before most samples enter Softmax Collapse, the global feature mean grows slowly. Once most samples collapse, $\|\boldsymbol{\mu}_G\|$ enters a rapid-growth phase. (c) $\boldsymbol{W}_G$ and $\boldsymbol{\mu}_G$ become anti-parallel, while the target logit decreases sharply.

where z_k represents the output logit for class k, and $z_m = \max z_k$. If the margin between the maximum logit and others satisfies $z_m - \max_{k \neq m} z_k > (p-1)\ln 2$, the second term vanishes due to absorption error, resulting in $Z = z_m$. In the late stages of training, the maximum logit z_m typically corresponds to the correct class logit z_r.

Definition 2 (Softmax Collapse [32]) *Under absorption error (where $Z = z_r$), the gradient for the logit of correct class r becomes strictly zero:*

$$g_r = \hat{y}_r - y_r = e^{z_r - Z} - 1 = e^{z_r - z_r} - 1 = 0 \tag{2}$$

At this point, the loss also becomes $-\log(\hat{y}_r) = 0$. However, for incorrect classes $k \neq r$, the gradients remain small but non-zero ($g_k = e^{z_k - z_r} \neq 0$).

Since Softmax Collapse (SC) arises in the terminal phase of training, its impact is inextricably linked to the geometric structure of the feature space in this regime. We assume the model has converged to the Neural Collapse (NC) state, which characterizes the geometry of the classifier weight matrix $\boldsymbol{W}$ and its input activations (features) $\boldsymbol{h}$.

Definition 3 (Neural Collapse [28]) *Consider a classification task with K classes. The Neural Collapse state is defined by the following conditions: **NC1:** Variability Collapse. Intra-class feature variability vanishes. For any sample i of class k, the feature vector $\boldsymbol{h}_{k,i}$ converges to the class mean $\boldsymbol{\mu}_k$; **NC2:** Simplex ETF. The centered class means $\boldsymbol{\mu}_k^* = \boldsymbol{\mu}_k - \boldsymbol{\mu}_G$ (where $\boldsymbol{\mu}_G$ is the global mean) form a Simplex Equiangular Tight Frame (ETF); **NC3:** Self-Duality. The classifier weight row vectors $\boldsymbol{W}_k$ align with the centered class means $\boldsymbol{\mu}_k^*$.*

Prieto et al. [32] identified SC, and hypothesized that it merely halts generalization. However, our investigation reveals a critical secondary effect: the interaction between SC and NC induces *Numerical Feature Inflation*, which directly triggers the Slingshot Mechanism.

2.2. Numerical Feature Inflation

In this section, we formalize the mechanism of Numerical Feature Inflation ($\mathcal{NFI}$). The interaction between SC and NC creates a deterministic feedback loop that drives feature inflation.

As established before, floating-point arithmetic introduces absorption errors. We first quantify the breaking of the zero-sum constraint.

Theorem 4 *Consider a network with CE loss. Assume the model is in an approximate NC state and satisfies the SC condition. During Gradient Descent with learning rate η, the expected update to global mean of the classifier weights $\boldsymbol{W}_G = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{W}_k$ on a class-balanced batch $\mathcal{B}$ is:*

$$\mathbb{E}_{\mathcal{B}}[\Delta \boldsymbol{W}_G] = -\frac{\eta\epsilon}{K}\boldsymbol{\mu}_G \tag{3}$$

where $\epsilon = \mathbb{E}[\sum_{k\neq r}\hat{y}_k]$ represents the expected residual probability mass on incorrect classes.

Proof Sketch. Consider an input $\boldsymbol{x}$ with feature $\boldsymbol{h}$ and label y_r. Ideally, the gradient of the loss $\mathcal{L}$ with respect to $\boldsymbol{W}_k$ satisfies a strict zero-sum constraint:

$$\sum_{k=1}^{K}\nabla_{\boldsymbol{W}_k}\mathcal{L} = \sum_{k=1}^{K}(\hat{y}_k - y_k)\boldsymbol{h} = \left(\underbrace{\sum_{k}\hat{y}_k}_{1} - \underbrace{\sum_{k}y_k}_{1}\right)\boldsymbol{h} = 0 \tag{4}$$

However, under SC, the gradient on correct class vanishes mathematically. The sum becomes: $\sum_{k=1}^{K}\nabla_{\boldsymbol{W}_k}\mathcal{L} \xrightarrow{SC} \sum_{k\neq r}\hat{y}_k\boldsymbol{h} = \epsilon\boldsymbol{h}$. The update rule $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta\nabla\boldsymbol{W}$ thus imparts a net drift to $\boldsymbol{W}_G$ in the direction of $-\boldsymbol{\mu}_G$. ■

This drift necessitates a redefinition of the geometric alignment.

Definition 5 (NC3′) *When $\boldsymbol{W}_G \neq \boldsymbol{0}$, the original weights $\boldsymbol{W}_k$ no longer form an ETF. Instead, the centered weights $\boldsymbol{W}_k^* = \boldsymbol{W}_k - \boldsymbol{W}_G$ satisfy the self-duality property, aligning with $\boldsymbol{\mu}_k^*$ and forming an ETF.*

This weight drift alters the gradient of feature layer.

Proposition 6 *Assume the model satisfies NC1, NC2, and NC3′, and that $\boldsymbol{W}_G$ is orthogonal to the classification subspace ($\boldsymbol{W}_G \perp$ span$\{\boldsymbol{W}_k^*\}$). Under the SC regime, the gradient of the loss with respect to the feature vector $\boldsymbol{h}$ contains a non-zero component parallel to $\boldsymbol{W}_G$:*

$$Proj_{\boldsymbol{W}_G}(\nabla_{\boldsymbol{h}}\mathcal{L}) = \epsilon\boldsymbol{W}_G \tag{5}$$

Proof Sketch. The gradient is $\nabla_{\boldsymbol{h}}\mathcal{L} = \sum_k(\hat{y}_k - y_k)\boldsymbol{W}_k = \sum_k(\hat{y}_k - y_k)(\boldsymbol{W}_G + \boldsymbol{W}_k^*)$. Since $\boldsymbol{W}_G \perp$ span$\{\boldsymbol{W}_k^*\}$, the projection onto $\boldsymbol{W}_G$ is proportional to $\boldsymbol{W}_G$. ■

Combining Theorem 4 and Proposition 6 reveals the feedback mechanism. The mutual reinforcement between the weight drift $\boldsymbol{W}_G$ and the feature drift $\boldsymbol{\mu}_G$ creates a coupled dynamic system:

Theorem 7 (Numerical Feature Inflation) *In a ReLU network with CE loss, the occurrence of Softmax Collapse induces a positive feedback loop. If the change of ϵ is negligible compared to the*

parameter dynamics (i.e., $|\dot{\epsilon}| \ll \frac{d}{dt}\|\boldsymbol{W}_G\|$), The norms of the global weight mean $\boldsymbol{W}_G$ and feature mean $\boldsymbol{\mu}_G$ exhibit exponential growth after long-term training:

$$\lim_{t\to\infty} \|\boldsymbol{W}_G^{(t)}\| \propto \left(1 + \frac{\eta\epsilon}{\sqrt{K}}\right)^t \tag{6}$$

$$\lim_{t\to\infty} \|\boldsymbol{\mu}_G^{(t)}\| \propto \left(1 + \frac{\eta\epsilon}{\sqrt{K}}\right)^t \tag{7}$$

where $\boldsymbol{W}_G$ progressively aligns anti-parallel to $\boldsymbol{\mu}_G$:

$$\lim_{t\to\infty} \cos(\boldsymbol{W}_G^{(t)}, \boldsymbol{\mu}_G^{(t)}) \to -1 \tag{8}$$

See Appendix D for the detailed proof.

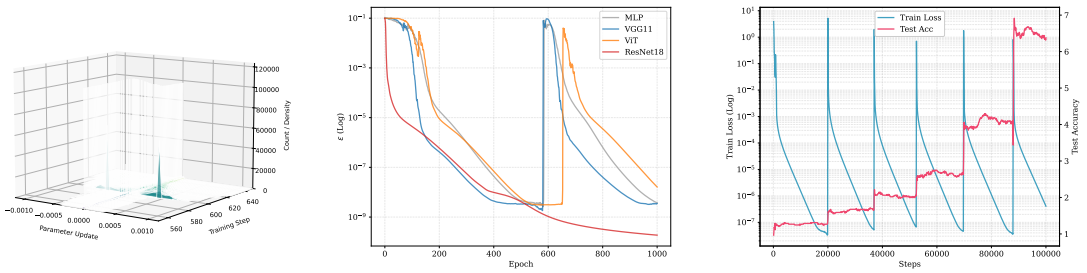
Slingshot Mechanism The exponential growth of $\boldsymbol{W}_G$ and $\boldsymbol{\mu}_G$ makes the sample-level absorption condition fragile rather than immediately causing a loss spike. Once some outlier samples fall below the absorption threshold, their correct-class gradients reappear, and Adam's amplified effective learning rate produces an excessively large update. This drives the loss back to the random-guessing level and yields the Slingshot spike. See details in Appendix B.1.

3. Empirical Results and Analysis

Experiment Setup. Our experiments focus on 3 different datasets: (1) **Modular Arithmetic:** For the modular division task with prime $p = 97$, we employ a 2-layer decoder-only Transformer and a 6-layer fully connected MLP. (2) **Image Classification:** We utilize the CIFAR-10 dataset. To ensure a comprehensive evaluation, We train a 6-layer MLP, VGG11, ResNet18, and ViT. (3) **Language Modeling:** We train a nanoGPT model with 110M parameters on the FineWeb dataset. Detailed hyperparameters are provided in Appendix E.

Gradient Re-emergence and Loss Spikes. Following Section 2.2, we estimate how a Slingshot loss spike is triggered. For a scalar toy model with global learning rate $\eta = 10^{-3}$ and $(\beta_1, \beta_2) = (0.9, 0.95)$, the pre-spike average gradient is about 3×10^{-9}, while the re-emerged gradient is larger than $\exp(-(p-1)\ln 2) = 1.19 \times 10^{-7}$. Following the Adam update rule, this gives the first moment $m_t = 1.46 \times 10^{-8}$ and the second moment $\sqrt{v_t} = 2.7 \times 10^{-8}$, so the Adam update at the spike step is approximately $\eta \frac{m_t}{\sqrt{v_t}+\varepsilon} = 4 \times 10^{-4}$. This matches Figure 2(*a*)subfigure: the classifier-layer update magnitude increases from about 10^{-5} before the spike to two sharp modes near $\pm 4 \times 10^{-4}$ at the spike epoch. This more than 40-fold increase in update magnitude is equivalent to applying a large signed displacement to many parameters. As a result, the predictions become almost random, and the loss increases to the 10^0 scale. This produces the observed Slingshot loss spike. We also test the commonly used Adam hyperparameters in computer vision, $(\beta_1, \beta_2) = (0.9, 0.999)$, and the suggested setting by Orvieto and Gower [26], $(\beta_1, \beta_2) = (0.9, 0.9)$. In both cases, the observed update spikes occur at magnitudes consistent with the same calculation.

Zero-Sum Projection. To verify Theorem 4, we constrain the update of W to remain in the subspace satisfying the zero-sum constraint in Eq. 4. Specifically, we apply a projection to the logit gradient $g = \nabla_z L$, We replace it by $g \leftarrow g - \frac{1}{K}\sum_k^K g_k$. This projection makes the gradient update of every sample satisfies the zero-sum constraint in Eq. 4. After enforcing this constraint



(*a*) (*b*) (*c*)

Figure 2: **Experimental Results.** (a) Around a loss spike, classifier-layer updates shift from near-zero values to sharp large modes. (b) Evolution of the residual probability mass ϵ across architectures. (c) In modular division, each Slingshot loss spike is accompanied by a stepwise increase in test accuracy.

throughout training, the Slingshot spike is eliminated. This result supports the prediction that the zero-sum-breaking component is necessary for the $\mathcal{NFI}$ feedback loop.

Architectural Dependence. We further test whether Slingshot occurs across different datasets and architectures, as summarized in Table 1. We find that all tested models exhibit Slingshot spikes except ResNet18. A closer analysis shows that this exception is consistent with Theorem 7. The $\mathcal{NFI}$ mechanism requires the change of ϵ to be negligible compared with the dynamics of W and μ. This condition holds for the other architectures. As shown in Figure 2(*b*)subfigure, before the loss spike occurs, ϵ decreases slowly or stays nearly constant. Therefore, the factor $(1 + \frac{\eta\epsilon}{\sqrt{K}})^t$ can support exponential growth. In contrast, if ϵ decays too fast, for example $\epsilon(t) \propto \frac{1}{t}$ or $\epsilon(t) \propto \frac{\log t}{t}$, then the accumulated growth of $(1 + \frac{\eta\epsilon(t)}{\sqrt{K}})^t$ is polynomial or logarithmic. Therefore, $\mathcal{NFI}$ cannot develop into the exponential-growth regime, and Slingshot does not occur. ResNet18 appears to learn fast enough to escape this unstable region before the $\mathcal{NFI}$ feedback loop becomes dominant.

Relation between Slingshot and Grokking An interesting consequence of the Slingshot Mechanism is its connection to grokking. As shown in Figure 2(*c*)subfigure, in the modular division task, each loss spike is accompanied by a stepwise increase in test accuracy, which eventually reaches 100%. We interpret these periodic spikes as a form of implicit perturbation: they repeatedly move the model away from its current low-loss state and allow training to resume in a different region of the loss landscape. This may help the model reach flatter and more generalizable solutions.

4. Conclusion

In this work, we have demystified the Slingshot Mechanism, revealing it to be artifacts of floating-point arithmetic— specifically Numerical Feature Inflation ($\mathcal{NFI}$)—rather than intrinsic optimization dynamics. Our results reveal a gap between gradient-flow analysis and real optimization. Existing refinements, such as central flows [7] and neural thermodynamics [48], still mainly describe the intrinsic dynamics of the model, while numerical dynamics are often ignored. We further show that SC and $\mathcal{NFI}$ are not limited to toy models: they also appear in practical large-model training and can substantially affect the optimization trajectory. Thus, finite-precision loss computation should be treated as a first-order factor in the analysis of long-term training stability.

References

[1] Ieee standard for floating-point arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, 2019.

[2] Zhiwei Bai, Zhangchen Zhou, Jiajie Zhao, Xiaolong Li, Zhiyu Li, Feiyu Xiong, Hongkang Yang, Yaoyu Zhang, and Zhi-Qin John Xu. Adaptive preconditioners trigger loss spikes in Adam. *arXiv preprint arXiv:2506.04805*, 2025.

[3] Etienne Boursier, Scott Pesme, and Radu-Alexandru Dragomir. A theoretical framework for grokking: Interpolation followed by riemannian norm minimisation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[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] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

[6] Jeremy Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, Zachary Nado, George E. Dahl, and Justin Gilmer. Adaptive gradient methods at the edge of stability. In *NeurIPS 2023 Workshop Heavy Tails in Machine Learning*, 2023.

[7] Jeremy Cohen, Alex Damian, Ameet Talwalkar, J Zico Kolter, and Jason D. Lee. Understanding optimization in deep learning with central flows. In *The Thirteenth International Conference on Learning Representations*, 2025.

[8] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. *Advances in Neural Information Processing Systems*, 33:18387–18398, 2020.

[9] Hien Dang, Tho Tran Huu, Tan Minh Nguyen, and Nhat Ho. Neural collapse for cross-entropy class-imbalanced learning with unconstrained reLU features model. In *Forty-first International Conference on Machine Learning*, 2024.

[10] 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 *9th International Conference on Learning Representations*, 2021.

[11] Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*, 2019.

[12] Connall Garrod and Jonathan P. Keating. The persistence of neural collapse despite low-rank bias. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[13] Satvik Golechha. Progress measures for grokking on real-world tasks. *arXiv preprint arXiv:2405.12755*, 2024.

[14] Andrey Gromov. Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*, 2023.

[15] Ting Han, Linara Adilova, Henning Petzka, Jens Kleesiek, and Michael Kamp. Flatness is necessary, neural collapse is not: Rethinking generalization via grokking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

[17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The Third International Conference on Learning Representations*, 2015.

[19] Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *The Twelfth International Conference on Learning Representations*, 2024.

[20] Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *The Eleventh International Conference on Learning Representations*, 2023.

[21] Jianfeng Lu and Stefan Steinerberger. Neural collapse under cross-entropy loss. *Applied and Computational Harmonic Analysis*, 59:224–241, 2022. ISSN 1063-5203. Special Issue on Harmonic Analysis and Machine Learning.

[22] Kaifeng Lyu, Jikai Jin, Zhiyuan Li, Simon Shaolei Du, Jason D. Lee, and Wei Hu. Dichotomy of early and late phase implicit biases can provably induce grokking. In *The Twelfth International Conference on Learning Representations*, 2024.

[23] Chao Ma, Lei Wu, and Weinan E. A qualitative study of the dynamic behavior for adaptive gradient algorithms. In *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145 of *Proceedings of Machine Learning Research*, pages 671–692. PMLR, 2022.

[24] 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.

[25] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

[26] Antonio Orvieto and Robert M. Gower. In search of Adam's secret sauce. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[27] Vardan Papyan. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(252):1–64, 2020a.

[28] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020b.

[29] 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. *Advances in neural information processing systems*, 32, 2019.

[30] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[31] Hari K Prakash and Charles H Martin. Grokking and generalization collapse: Insights from HTSR theory. *arXiv preprint arXiv:2506.04434*, 2025.

[32] Lucas Prieto, Melih Barsbey, Pedro A. M. Mediano, and Tolga Birdal. Grokking at the edge of numerical stability. In *The Thirteenth International Conference on Learning Representations*, 2025.

[33] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[34] Keitaro Sakamoto and Issei Sato. Explaining grokking and information bottleneck through neural collapse emergence. *arXiv preprint arXiv:2509.20829*, 2025.

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

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations*, 2015.

[37] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

[38] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

[39] Felix Stollenwerk, Anna Lokrantz, and Niclas Hertzberg. Output embedding centering for stable llm pretraining. *arXiv preprint arXiv:2601.02031*, 2026.

[40] Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. *arXiv preprint arXiv:2206.04817*, 2022.

[41] Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua M. Susskind. The slingshot effect: A late-stage optimization anomaly in adaptive gradient methods. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

[42] Yuandong Tian. Provable scaling laws of feature emergence from learning dynamics of grokking. In *The Fourteenth International Conference on Learning Representations*, 2026.

[43] 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.

[44] Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie E Everett, Alexander A Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-Dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities. In *The Twelfth International Conference on Learning Representations*, 2024.

[45] Shuo Xie and Zhiyuan Li. Implicit bias of adamw: ℓ_∞-norm constrained optimization. In *Forty-first International Conference on Machine Learning*, 2024.

[46] Zhiwei Xu, Zhiyu Ni, Yixin Wang, and Wei Hu. Let me grok for you: Accelerating grokking via embedding transfer from a weaker model. In *The Thirteenth International Conference on Learning Representations*, 2025.

[47] Yushun Zhang, Congliang Chen, Naichen Shi, Ruoyu Sun, and Zhi-Quan Luo. Adam can converge without any modification on update rules. In *Advances in Neural Information Processing Systems*, 2022.

[48] Liu Ziyin, Yizhou Xu, and Isaac L. Chuang. Neural thermodynamics: Entropic forces in deep and universal representation learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Appendix Contents

Appendix A. Related Work

Grokking and Numerical Dynamics. Weight decay has traditionally been credited as the primary driver of grokking [3, 20, 22, 42]. However, growing empirical evidence shows that grokking can also occur without explicit regularization, although many of these demonstrations are conducted under mean-squared error (MSE) loss [13, 14, 19, 31]. In contrast, achieving grokking with CE loss and no weight decay appears theoretically impractical: Lyu et al. [22] estimate that it would require approximately 10^{100} steps under Gradient Descent. Nevertheless, Thilak et al. [40] observed grokking under CE loss with Adam, where generalization is accompanied by recurrent Slingshot loss spikes. Given the theoretical difficulty of unregularized CE grokking, recent work has pivoted to examining the role of numerical precision. Nanda et al. [25] first linked the Slingshot Mechanism to numerical instability, noting that lower precision exacerbates these events. Xu et al. [46] subsequently found that employing `float64` precision could mitigate the Slingshot effect. More recently, Prieto et al. [32] conducted a detailed analysis of these instabilities, identifying "Softmax Collapse" (SC) as a critical phenomenon. They showed that in PyTorch's CE loss implementation, absorption errors occur when the correct logit z_m exceeds incorrect logits by a threshold determined

by the mantissa precision (for `float32`, $z_m - \max_{k \neq m} z_k > 23 \ln 2 \approx 16$). Under SC, the gradient on the correct class strictly rounds to zero, halting test accuracy improvement. However, Prieto et al. stopped short of establishing a causal link between SC and the Slingshot mechanism itself, hypothesizing instead that Slingshots might serve as an intrinsic optimization response to avoid SC.

Neural Collapse. Recent studies indicate that neural networks tend to converge to a state known as Neural Collapse (NC) after prolonged training [28]. Theoretical works have established that under CE loss without weight decay, the NC state represents a global optimum [12, 21]. Furthermore, Dang et al. [9] demonstrated that with ReLU activation, the class means of feature vectors tend to become mutually orthogonal. Sakamoto & Sato [34] established a connection between grokking and NC. While generalization does not strictly require NC, Han et al. [15] suggest that models exhibit a strong propensity toward NC in the absence of explicit regularization.

The Instability of Adam and Loss Spikes. The convergence properties of Adam [18] have been extensively studied. Reddi et al. [33] highlighted the non-convergence issues of Adam. Shazeer & Stern [35] showed that a slow decay rate of the second moment accumulator v_t can cause larger-than-desired updates and training instability. Zhang et al. [47] showed that appropriate hyperparameter selection can ensure convergence. Cohen et al. [6, 7] investigated the EOS phenomenon and loss spikes in adaptive optimization. Molybog et al. [24] attributed loss spikes to time-domain gradient correlations that cause the Adam update ratio to shift into a bimodal distribution. In this regime, anomalous data batches trigger a chain reaction that forces the update vector to depart from its suppressed state, causing the training loss to explode. Subsequently, Bai et al. [2] provided a stability-based explanation, observing that when gradients remain small for extended periods (flat landscapes), the adaptive term v_t decays, causing the effective learning rate $\eta/(\sqrt{v_t} + \varepsilon)$ to explode. In standard settings ($\eta = 10^{-3}, \varepsilon = 10^{-8}$), this effective step size can be amplified by orders of magnitude (e.g., 10^5), making the model highly susceptible to drastic updates upon the re-emergence of gradient signals. However, these analyses typically focus on MSE loss where the Hessian is non-vanishing. In contrast, Ma et al. [23] theoretically argued that under CE loss, the Hessian eigenvalues vanish in the late training phase, implying that the optimization should stabilize and be immune to such spikes.

Appendix B. Additional Discussion

B.1. The Mechanism of the Slingshot Mechanism

The exponential growth of the global classifier mean and the global feature mean does not by itself immediately produce a loss spike. Instead, it drives the system into a regime where the sample-level absorption condition becomes fragile.

In an ideal NC state, all features $\boldsymbol{h}_{k,i}$ of class k are close to their class mean $\boldsymbol{\mu}_k$. In this case, within the subspace $\text{span}\{\boldsymbol{\mu}_k^*\}$ orthogonal to $\boldsymbol{\mu}_G$, the remaining gradients from incorrect classes still push the features of different classes away from each other. Therefore, a Slingshot spike does not occur immediately. However, this approximation no longer holds once the exponential growth of the global feature mean becomes dominant. As shown in Proposition 6, the growth along the $\boldsymbol{\mu}_G/\boldsymbol{W}_G$ direction is proportional to the residual probability mass on incorrect classes, denoted by ϵ. Thus, samples closer to the decision boundary, which have larger residual probability mass, move faster along the $\boldsymbol{\mu}_G$ direction. After this exponential growth continues for some time, the intra-class variance can become comparable to the inter-class variance. At this stage, the gradient updates may

still increase the distance between two class means $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$, but they can also reduce the margin between some outlier samples $\boldsymbol{h}_{p,o}$ and the decision boundary. Once this margin falls below the absorption threshold $z_m - \max_{k \neq m} z_k > (p-1)\ln 2$, the correct-class gradient signal reappears. Before this re-emergence, the correct-class gradient is zero and the incorrect-class gradients are extremely small. As a result, the effective learning rate of Adam is amplified toward its theoretical maximum $\eta/\varepsilon_{\text{Adam}}$. When the correct-class gradient of some samples suddenly changes from zero to a finite value, Adam cannot immediately reduce the effective learning rate because of its moment estimates. This produces an excessively large update step. In our experiments, the update magnitude at the spike epoch is typically about 10^2 times larger than that in the previous epoch. Such a large update substantially changes the model parameters and drives the loss back to the random-guessing level. This produces the loss spikes known as the *Slingshot Mechanism*.

B.2. Mitigation Study

Following our identification of the $\mathcal{NFI}$ mechanism, we conducted a series of ablation studies to evaluate whether specific architectural or algorithmic modifications could mitigate the slingshot effect. We omit discussion of explicit regularization terms like weight decay or z-loss (constraining the log softmax normalizer Z), given that their ability to mitigate Slingshot has been demonstrated in existing works [4, 40, 44].

B.2.1. MIXED PRECISION.

See Figure 1(*a*)subfigure, upgrading the Softmax computation to `float64` precision is sufficient to eliminate the slingshot. The absorption error threshold for double precision is $\frac{|b|}{|a|} < 2^{-52} \approx 2.22 \times 10^{-16}$. Achieving a training loss magnitude low enough to trigger this threshold is generally infeasible within standard training durations.

B.2.2. OPTIMIZER.

While the Slingshot mechanism is typically associated with adaptive optimizers, we demonstrate that the onset of the instability is independent of adaptivity and is driven solely by the magnitude of the step size. To verify this, we switched the optimizer from Adam to vanilla GD immediately after the training loss reached strict zero. We manually set the learning rate to $\eta = 10^5$ to emulate the massive effective step size ($\approx \eta_{\text{Adam}}/\varepsilon_{\text{Adam}}$). Crucially, we observed that GD triggers a Slingshot spike at approximately the same epoch as the baseline Adam run. We note, however, a key distinction in the aftermath: while Adam is able to adapt its moments to recover from the spike (forming the periodic pattern), the high-learning-rate GD fails to reconverge, leading to permanent divergence.

B.2.3. INCREASING $\varepsilon_{\text{ADAM}}$.

Thilak et al. [40] reported that increasing the Adam optimizer's ε parameter to 10^{-5} prevents slingshots. Our analysis confirms that this intervention works by lowering the upper bound of the effective learning rate ($\eta/\varepsilon_{\text{Adam}}$), thereby delaying the amplification of the re-emerging gradient signals.

B.2.4. LAYER NORMALIZATION (LN).

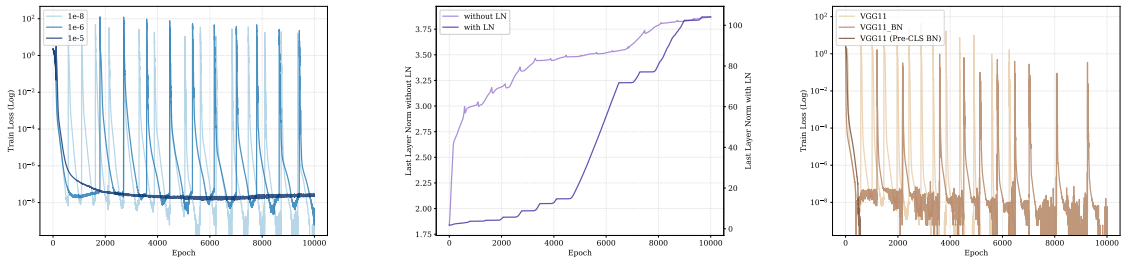
Applying LN to features immediately before the classifier fails to mitigate the effect. Since LN normalizes samples individually, it does not prevent the collective alignment of features towards a common direction (i.e., $\mathcal{NFI}$ persists). In fact, see Figure 3(b)subfigure, our experiments indicate that LN *accelerates* the onset of slingshots. LN constrains the feature norm, and smaller feature norms mean that even minor reductions in angular separation can violate the absorption threshold. Furthermore, LN decouples the dynamics of direction and magnitude. In our unnormalized MLP experiments, the last layer norm grows continuously, consistent with Soudry et al. [38]. However, with LN, we observe the stepwise growth noted by Thilak et al.: the model alternates between optimizing angular alignment to reduce loss and, when angles stabilize, increasing the scalar scale of the last layer.

B.2.5. BATCH NORMALIZATION (BN).

As shown in Figure 3(c)subfigure, BN applied directly before the classifier successfully eliminates slingshots. By centering the batch features (subtracting the global mean $\boldsymbol{\mu}_G$), BN removes the principal drift component. We note that Thilak et al. observed that BN failed to prevent slingshots. This discrepancy arises from their architecture: they applied BN within the convolutional layers of VGG11, followed by another ReLU layers which re-introduced rank compression [8]. We verify that BN is effective when applied immediately before the final linear layer.

B.2.6. LABEL SMOOTHING (LS).

Label Smoothing sets the target probability to $y < 1$, preventing the infinite growth of correct-class logits. While this successfully eliminates the precision-induced $\mathcal{NFI}$ spikes, our experiments reveal that LS introduces a new class of instabilities independent of numerical precision. When



(a) (b) (c)

Figure 3: **Mitigation Study. (a) Adam's ε.** Increasing the optimizer's ε parameter mitigates instability: while $\varepsilon = 10^{-6}$ reduces the frequency of spikes, setting $\varepsilon = 10^{-5}$ completely eliminates them. **(b) Layer Norm.** Applying LN changes the evolution of the last layer norm from a continuous trajectory to a distinct stepwise pattern. Notably, LN significantly increases the magnitude of the last layer norm. **(c) Batch Norm.** Standard VGG11 with BN within convolutional layers still exhibits Slingshot spikes. However, applying BN before the final classifier eliminates the instability, stabilizing the training at the zero-loss SC state.

the target probability is not 1 (and incorrect targets are not 0), the theoretical guarantee by Ma et al. [23]—that Hessian eigenvalues vanish late in training—no longer holds. Instead, the maximum Hessian eigenvalue remains finite. This pushes the optimization into the EOS regime, where loss spikes arise from the intrinsic sharpness of the loss landscape (see Figure 4(*b*)subfigure). A detailed discussion distinguishing Slingshot from EOS is provided in Appendix B.4, and why LS leads to finite Hessian eigenvalues is elaborated in Appendix D.4.3.

B.2.7. DATASET SIZE

To rigorously evaluate whether the Slingshot mechanism is an artifact confined to the small-data regime, we conducted experiments on CIFAR-10 subsets of varying magnitudes: $N \in \{200, 2000, 20000, 50000\}$, where $N = 50000$ represents the full training set. To disentangle the effect of dataset size from the noise induced by stochastic sampling (as discussed in Appendix B.5.2), we employed *Full Batch Adam* for all configurations.

We observed distinct Slingshot loss spikes across all dataset sizes, including the full CIFAR-10 set. This empirical evidence demonstrates that the occurrence of Slingshots is independent of the dataset size. It confirms that as long as the model possesses sufficient capacity to drive the residual probability mass ϵ below the floating-point absorption threshold, the $\mathcal{NFI}$ feedback loop will be triggered, provided that the optimization trajectory is not continuously disrupted by mini-batch noise.

B.2.8. LEARNING RATE

We conducted a sensitivity analysis by sweeping the global learning rate η across the range $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. Counter-intuitively, we observe that *smaller* learning rates are significantly more prone to triggering Slingshot instabilities.

This phenomenon implies that a larger learning rate provides a crucial form of implicit regularization. Large step sizes introduce stochastic noise that prevents the optimizer from settling into sharp, rugged basins of attraction, effectively smoothing the optimization trajectory. In contrast, a small learning rate allows the model to converge faithfully to the nearest local minimum. In the complex loss landscape of deep networks, these "nearest" solutions are frequently "bad" local minima characterized by extreme sharpness.

B.2.9. BIAS

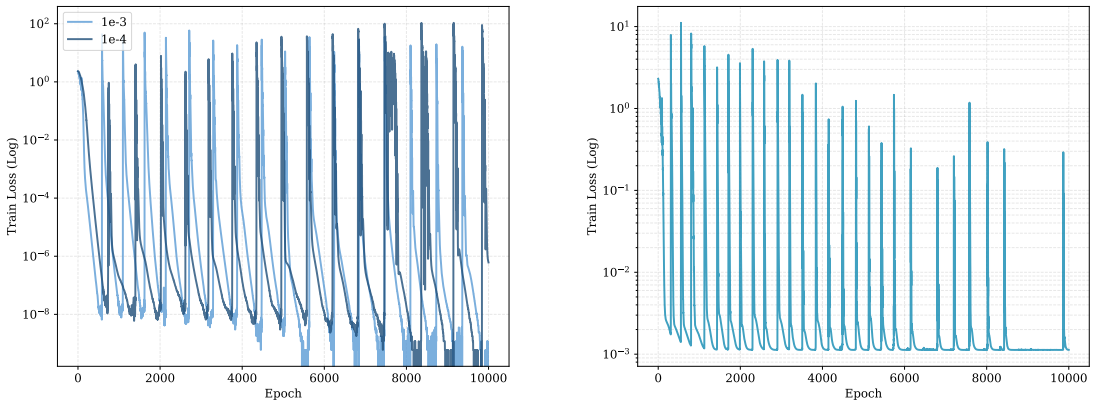
We find that including a bias term in the classification layer accelerates the occurrence of Slingshots. This instability stems from a significant scale discrepancy between the updates of weights and biases. The gradient with respect to the weight $\boldsymbol{W}_k$ is scaled by the feature vector $\boldsymbol{h}$:

$$\nabla_{\boldsymbol{W}_k} \mathcal{L} = (\hat{y}_k - y_k)\boldsymbol{h} \tag{9}$$

In contrast, the gradient with respect to the bias b_k is:

$$\nabla b_k \mathcal{L} = (\hat{y}_k - y_k) \tag{10}$$

In our experiments, the feature norm $\|\boldsymbol{h}\|$ grows to approximately 10^2. This implies that the weight matrix $\boldsymbol{W}$ updates two orders of magnitude faster than the bias b. As a result, the bias term effectively "lags behind" the rapidly evolving weights. This dynamic mismatch destabilizes the logit



(*a*) (*b*)

Figure 4: **(a)** Training Loss with different learning rates. **(b)** Train Loss with label smoothing.

computation $z_k = \boldsymbol{W}_k^T \boldsymbol{h} + b_k$, making the system more prone to violating the absorption threshold and triggering spikes.

B.2.10. NUMERICAL UNDERFLOW

We demonstrate that the Slingshot mechanism is unrelated to numerical underflow (often referred to as precision overflow or range truncation).

Floating-point arithmetic is subject to two distinct types of precision loss: *absorption error* (due to limited mantissa bits) and *underflow* (due to limited exponent bits). Underflow occurs when a value is smaller than the minimum representable positive number, causing it to be truncated to zero. Specifically, for the Softmax function, the gradient with respect to the minimum logit z_{min} vanishes if:

$$\exp(z_{min} - z_{max}) < 2^{-(p-1+2^{E-1}-2)} \tag{11}$$

where p denotes the number of mantissa bits and E the number of exponent bits. For standard `float32` precision, this cutoff occurs when the logit difference satisfies $z_{max} - z_{min} > 149 \ln 2 \approx 103.28$.

To rule out underflow as a cause, we conducted an experiment where we explicitly clamped the logits such that the margin satisfies $z_{min} \geq z_{max} - 100$ at all times. This constraint guarantees that gradients remain strictly within the representable range of `float32`. We observed that Slingshot spikes persist despite this intervention, confirming that the phenomenon is driven purely by absorption errors in the mantissa, independent of exponent-based underflow.

B.3. $\mathcal{NFI}$ in Real-World Tasks

So far, we have mainly studied the full-batch setting. In real-world training, however, mini-batches are used for both efficiency and generalization. Mini-batch stochasticity makes it unlikely that all samples fall below the Softmax Collapse threshold at the same time, except in highly structured and nearly noiseless tasks such as modular arithmetic. Therefore, a visible Slingshot loss spike may not appear. Nevertheless, we find that $\mathcal{NFI}$ can still occur.

B.3.1. MINI BATCH.

For VGG11 trained with batch size 256, approximately half of samples still enter Softmax Collapse after 10^6 steps training, with exactly zero loss. Samples that do not collapse keep nonzero correct-class gradients. These gradients continue to separate features in the classification subspace, keep Adam's moment estimates active, and prevent the effective learning rate from suddenly becoming too large. Hence, no Slingshot spike is observed. However, collapsed samples still induce the $\mathcal{NFI}$ drift, leading to rapid late-stage growth of $\boldsymbol{\mu}_G$ and the logits. Consistent with this interpretation, subtracting $\boldsymbol{W}_G$ or applying BatchNorm before the classifier, which removes $\boldsymbol{\mu}_G$ during training, substantially slows down the late-stage parameter growth.

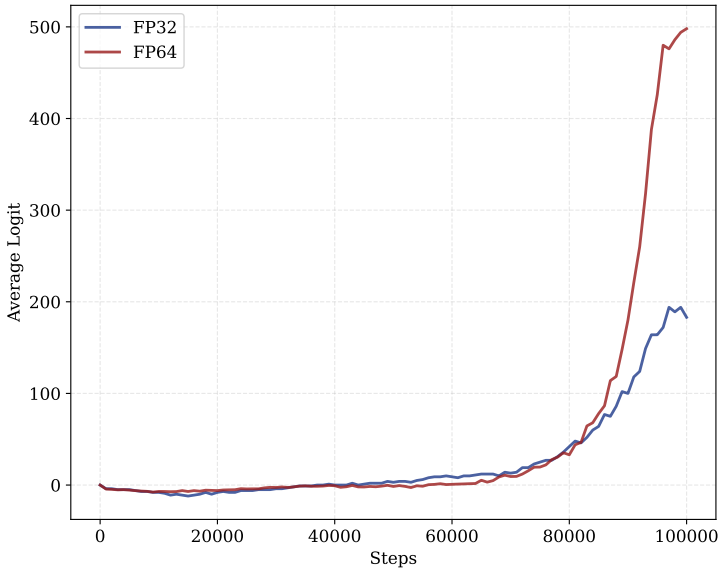


Figure 5: The average logit of different precisions in LLM training.

B.3.2. LARGE LANGUAGE MODELS.

We also examine the language-modeling setting. Softmax Collapse also appears during GPT training: in each step with about 1.3×10^5 tokens, roughly 4000 tokens have exactly zero loss. These tokens correspond to highly predictable contexts. On FineWeb, the top-10 tokens with the highest Softmax Collapse frequency are ".", "org", "example", ",", "to", "t", "last", " ", " you", and "of". Several of them, such as "example", "org", and "last", are not simply the most frequent tokens. Instead, they are highly predictable because of dataset-specific templates, such as "example.org" or form fields containing "first name" and "last name".

Long language-model training also shows abnormal logit growth. To test the effect of numerical precision, we replace the loss computation with `float64` and compare it with standard `bf16` training (calculate the loss in `float32`), the result is shown in Figure 5. Interestingly, unlike the MLP, CNN, and ViT experiments, higher precision does not reduce logit growth in this setting. After 10^5 steps, the mean logit is 183 under `float32` training, but increases to 498 when the loss is computed in `float64`.

We believe this difference comes from the special structure of natural language data. In the previous classification tasks, classes are balanced. In language modeling, token frequencies follow Zipf's law, so the output embedding vectors $\boldsymbol{W}_k$ are inherently imbalanced toward frequent tokens. This imbalance can create a large global mean $\boldsymbol{W}_G$ even without numerical collapse [11]. As a result, the features and output embeddings can reinforce each other in the same direction: frequent-token embeddings guide their corresponding features to grow along aligned directions. In contrast, $\mathcal{NFI}$ creates an anti-parallel interaction between $\boldsymbol{W}_G$ and $\boldsymbol{\mu}_G$. Therefore, in this setting, low precision can partially suppress the faster logit divergence caused by the frequency-induced $\boldsymbol{W}_G$.
In both cases, the key factor is the global mean component of the output embedding. Removing the last-layer mean $\boldsymbol{W}_G$ eliminates this mutual reinforcement and stabilizes the logits. We provide further discussion of logit divergence in language models in Appendix B.6.

B.4. Is Slingshot an Edge of Stability Phenomenon?

While both the Slingshot mechanism and the Edge of Stability (EOS) phenomenon manifest as frequent oscillations and spikes in the training loss trajectory, our theoretical analysis and empirical evidence demonstrate that they are governed by fundamentally different mechanisms.
In the context of convex optimization, the condition for Gradient Descent (GD) to converge stably on a function f is that the learning rate must satisfy $\eta < 2/\lambda$, where λ represents the Lipschitz constant of the gradient ∇f. For twice-differentiable functions, λ corresponds to the upper bound of the maximum eigenvalue (spectral norm) of the Hessian matrix $\nabla^2 f(x)$.
Cohen et al. [5] identified the phenomenon of "Progressive Sharpening" in neural network training, where the maximum eigenvalue of the Hessian, λ_{max}, steadily increases until it reaches the stability threshold $2/\eta$. Upon breaching this limit, the optimization enters an unstable regime characterized by oscillations, which effectively regulate λ_{max} to hover around the critical value $2/\eta$. This self-stabilizing behavior near the limit of divergence is termed the "Edge of Stability". The visual signature of EOS in the training loss curve is typically a series of high-frequency oscillations.
For adaptive optimizers, the stability condition extends to the maximum eigenvalue of the *Preconditioned Hessian* $\mathbf{PH}(w)$, where $\mathbf{P} = \text{diag}(1/(\sqrt{v} + \epsilon))$ acts as the preconditioner [6, 7]. In the presence of momentum (e.g., in Adam), this stability threshold is further modified by a coefficient $\kappa(\beta_1, \beta_2)$ dependent on the hyperparameters [2, 6].
Although Slingshot events also manifest as loss spikes, their underlying principle is distinct from EOS for several reasons:

B.4.1. VANISHING HESSIAN UNDER CROSS-ENTROPY.

First, in the specific setting of CE loss, the progressive sharpening required for EOS does not occur in the late stages of training. Instead, λ_{max} tends toward zero.

Theorem 8 *Consider a neural network with parameters $\theta \in \mathbb{R}^d$ and output logits $z(\theta, x) \in \mathbb{R}^K$. For a classification task with Cross-Entropy loss $\mathcal{L}$, if the model converges to an interpolation solution (i.e., the predicted probability vector $\hat{y}$ converges to the one-hot label y), then the maximum eigenvalue of the Hessian matrix $\lambda_{\max}(H_\theta)$ converges to 0, provided that the derivatives of the network are locally bounded.*

The proof is provided in Section D.4. Consequently, the optimization operates far below the stability threshold $2/\eta$ during the late training phase, implying that the instability does not arise from the curvature of the loss landscape.

B.4.2. NUMERICAL ARTIFACT VS. INTRINSIC PROPERTY.

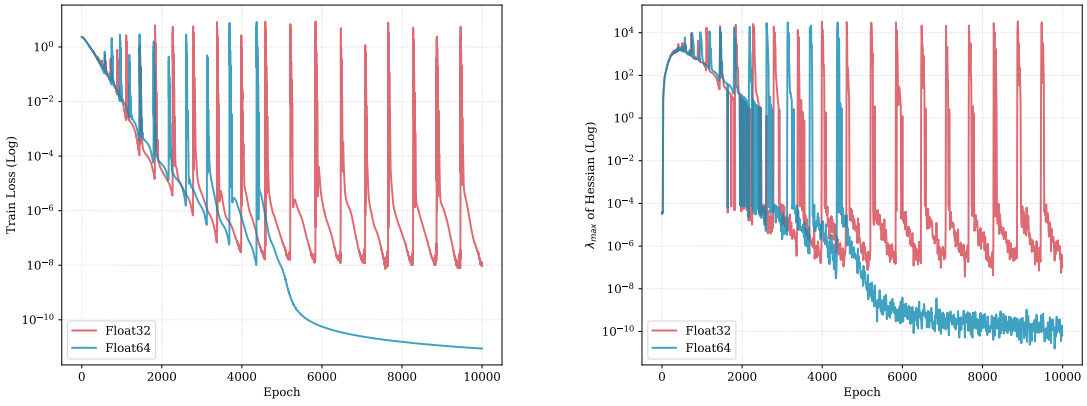
Second, EOS is an intrinsic property of the optimization landscape that persists regardless of numerical precision. In stark contrast, the Slingshot effect is a numerical artifact. As demonstrated in our main results, simply increasing the precision to `float64` eliminates Slingshot spikes entirely, whereas true EOS oscillations would persist in double precision.

B.4.3. COEXISTENCE OF PHENOMENA.

Finally, we note that these two phenomena can coexist within the same training run. We verified this by training a narrow network (hidden dimension $d = 20$). As illustrated in Figure 6, the model trained in `float32` exhibits loss spikes in both the early and late stages of training. However, when trained in `float64`, the late-stage spikes (Slingshots) disappear, while the early-stage oscillations—occurring while λ_{max} is still finite and large—persist. This confirms that the early instability is genuine EOS, while the late-stage instability is strictly a result of numerical breakdown.

B.5. Why are Slingshots Rare in Practical Settings?

The Slingshot mechanism has remained largely underexplored primarily because it rarely manifests under standard deep learning training configurations. We identify two dominant factors that naturally suppress this phenomenon in conventional settings:



(*a*) (*b*)

Figure 6: **Coexistence of EOS and Slingshot Phenomena.** **(a)** Training Loss showing early EOS oscillations versus late-stage numerical spikes. **(b)** Evolution of Maximum Hessian Eigenvalue λ_{max}.

B.5.1. WEIGHT DECAY.

Explicit regularization effectively constrains the magnitude of weights and, by extension, the growth of output logits. Xie & Li [45] theoretically demonstrated that for the AdamW optimizer with a weight decay factor λ, the parameter magnitude is bounded by $\|w\|_\infty \lesssim 1/\lambda$. Since the logits are kept within a finite, bounded range, the logit margin $z_m - \max_{k \neq m} z_k$ rarely exceeds the critical threshold required to trigger floating-point absorption errors (approx. 16 for `float32`). Without this numerical breakdown, the feedback loop cannot initiate.

B.5.2. MINI-BATCH

While we have confirmed that Slingshot events can occur on the full CIFAR-10 dataset, they are often more difficult to observe in standard training regimes due to the dynamics of mini-batch. We identify two primary mechanisms through which mini-batching suppresses or delays the onset of $\mathcal{NFI}$:

Stochastic Delay of Convergence. The onset of the $\mathcal{NFI}$ feedback loop requires the model to enter a "silent" regime where the residual probability mass ϵ drops below the floating-point absorption threshold (approx. 10^{-7} for `float32`). In full-batch gradient descent, the loss decreases monotonically and rapidly, allowing the model to hit this numerical floor quickly. However, the stochastic noise introduced by mini-batch sampling creates fluctuations in the optimization trajectory. These fluctuations significantly impede the precise, asymptotic convergence required to trigger absorption errors.

Implicit Regularization towards Flat Minima. More fundamentally, mini-batch SGD creates an implicit regularization effect that biases the optimization towards flatter minima [17, 37]. Therefore, Slingshots are most prominent when the batch size is large enough (approximating full-batch dynamics).

B.5.3. IMPLICATIONS FOR LOW-PRECISION TRAINING.

While Slingshot events are effectively mitigated in standard `float32` training, our findings suggest they may pose a significant latent risk for modern low-precision paradigms. As the field moves toward `BF16`, `FP8`, and even `FP4` training for Large Language Models, the margin required to trigger absorption errors shrinks dramatically. We posit that loss spikes induced by $\mathcal{NFI}$ could become a critical source of instability in these aggressive quantization regimes.

B.6. Discussion on output logit divergence in LLMs.

Abnormally rapid logit growth after long-term LLM training was first observed in PaLM [4] and was later termed "output logit divergence" by Wortsman et al. [44]. In their experiments, the average logit rapidly drifts toward negative infinity, which appears qualitatively similar to the drift dynamics observed in our setting (see Figure 1(*c*)subfigure). In our setting, the origin of this drift is explicit: as $\boldsymbol{W}_G$ and $\boldsymbol{\mu}_G$ become anti-parallel and grow exponentially, every logit contains a large common component

$$z_k = \boldsymbol{W}_k^\top \boldsymbol{h} = (\boldsymbol{W}_G + \boldsymbol{W}_k^*)^\top (\boldsymbol{\mu}_G + \boldsymbol{h}^*) = \boldsymbol{W}_G^\top \boldsymbol{\mu}_G + \boldsymbol{W}_G^\top \boldsymbol{h}^* + (\boldsymbol{W}_k^*)^\top \boldsymbol{\mu}_G + (\boldsymbol{W}_k^*)^\top \boldsymbol{h}^*. \quad (12)$$

The dominant term $\boldsymbol{W}_G^\top \boldsymbol{\mu}_G$ tends to $-\infty$, thereby driving all logits downward. Both Wortsman et al. and Thilak et al. [41] noted possible similarities between these phenomena, and we initially

considered that they might share a related origin. However, when we attempted to reproduce this phenomenon, we did not observe the same behavior. Following the model architectures and hyperparameters described in their paper, we trained GPT-2 models with 19M and 150M parameters. We first implemented the experiments in PyTorch, but we could not reproduce the divergence of logits toward negative infinity reported in Figure 4 of the arXiv version of their paper (Figure G.2 in the proceedings version). We then used NanoDo[1], the JAX/Flax-based small GPT-2 implementation that their paper states it is based on. After modifying the configuration to match the paper description as closely as possible, we still did not observe the reported phenomenon.

After further checking related materials, we found that, despite the paper being an ICLR Oral, we are not aware of any independent work that has successfully reproduced the same phenomenon in the three years since its publication. Stollenwerk et al. [39] report a successful reproduction, but when we ran their released source code, the observed behavior was not exactly the same. In addition, we noticed that only Google-related open-source models, such as PaLM and Gemma, discuss how to mitigate output logit divergence in their technical reports, while other open-source models do not mention this issue. This is surprising, because Wortsman et al. claim that the phenomenon can simply be reproduced in a model with only 2.4M parameters by increasing the learning rate to 0.1. Therefore, we suspect that this phenomenon may be specific to running JAX/Flax on TPUs, or that there may be additional implementation details that were not described in the paper. Unlike PyTorch, which provides relatively mature precision protection through autocast, mixed-precision training in Flax often requires manual casting when implementing neural network modules. This may introduce additional precision-related issues.

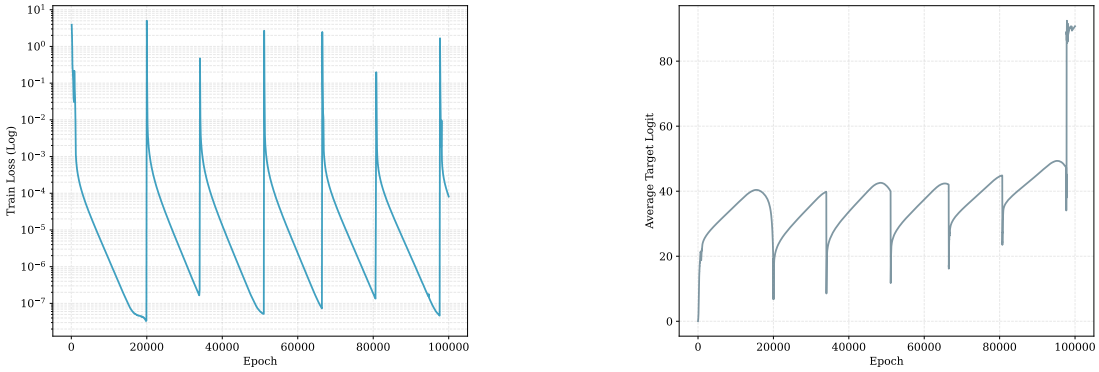
The phenomenon we observe in LLMs is instead consistent with Stollenwerk et al.: the output logits diverge toward positive infinity.

Appendix C. Additional Results

Table 1: Summary of Loss Spike observations across different model architectures and datasets.

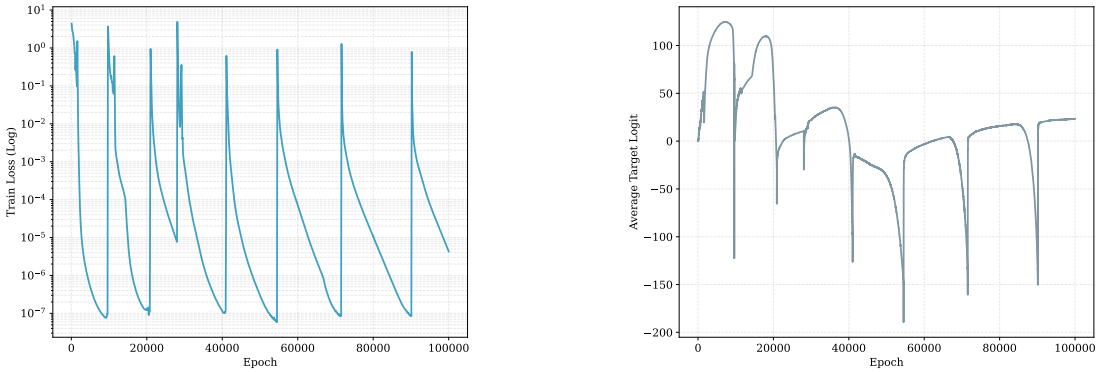
Model	**Dataset**	**Loss Spike?**	**Figure**
Transformer	Modular Division	Yes	Figure 7
MLP	Modular Division	Yes	Figure 8
MLP	CIFAR-10	Yes	Figure 9
VGG11	CIFAR-10	Yes	Figure 10
VGG11 with BN	CIFAR-10	Yes	Figure 11
ViT	CIFAR-10	Yes	Figure 12
ResNet18	CIFAR-10	**No**	Figure 13

1. https://github.com/google-deepmind/nanodo



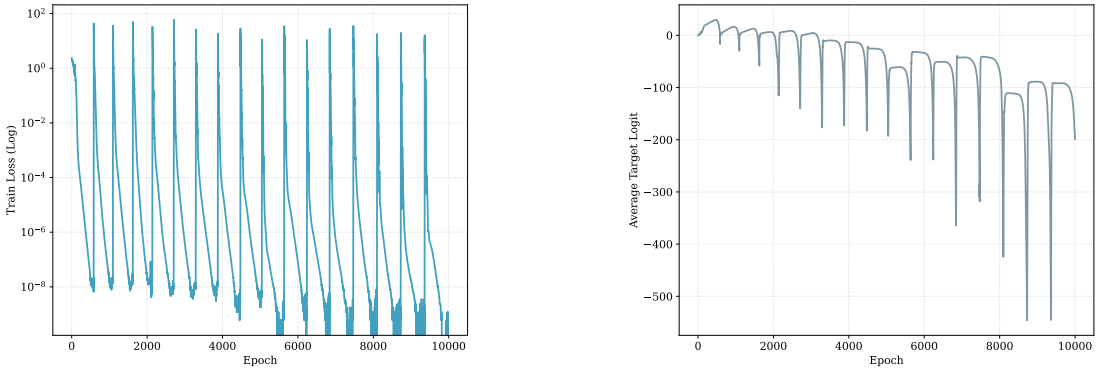
(*a*) (*b*)

Figure 7: Slingshot in Transformer on modular division.



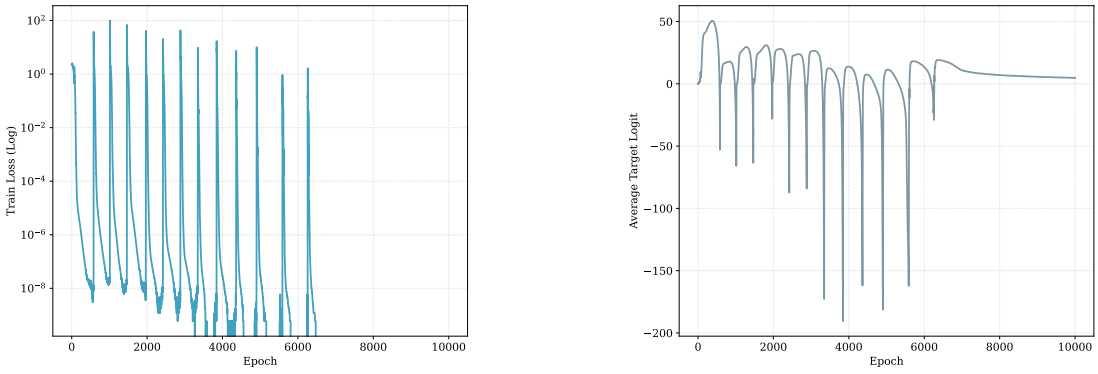
(*a*) (*b*)

Figure 8: Slingshot in MLP on modular division.



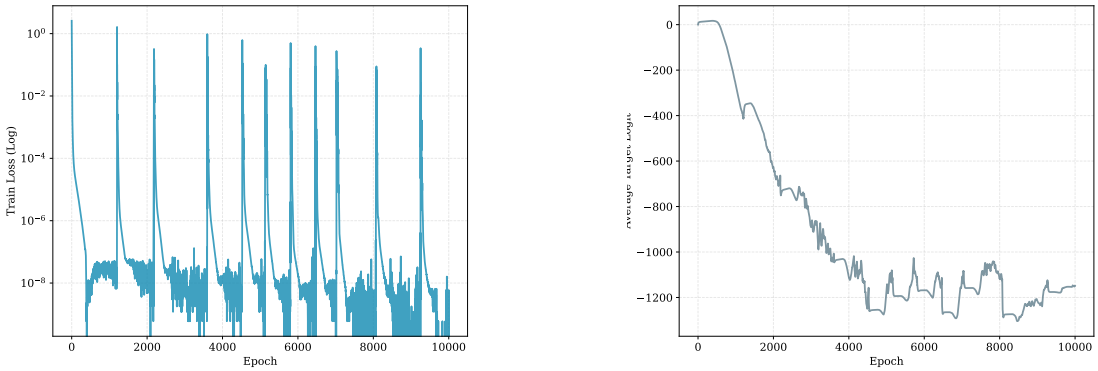
(*a*) (*b*)

Figure 9: Slingshot in MLP on CIFAR-10.



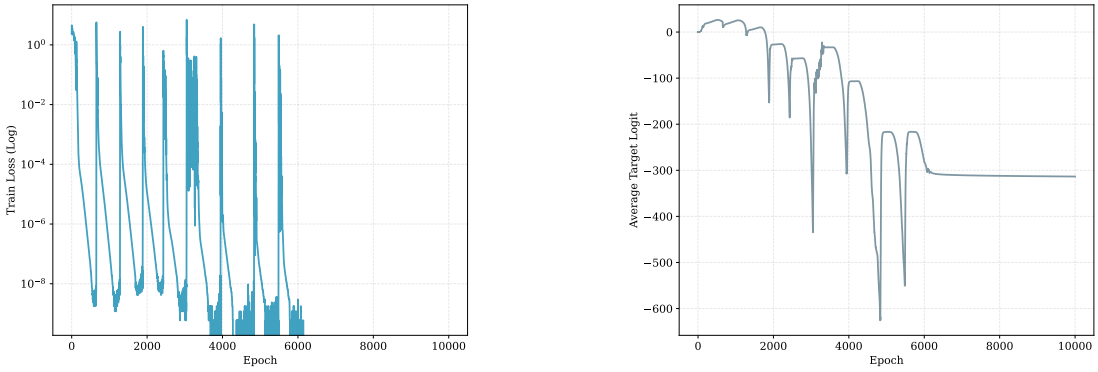
(*a*) (*b*)

Figure 10: Slingshot in VGG11 on CIFAR-10.



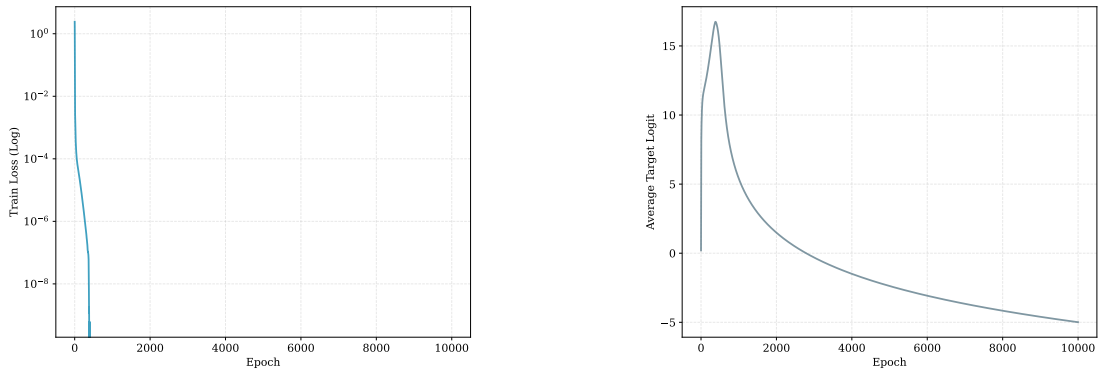
(*a*) (*b*)

Figure 11: Slingshot in VGG11 with BN on CIFAR-10.



(*a*) (*b*)

Figure 12: Slingshot in ViT on CIFAR-10.



(*a*) (*b*)

Figure 13: No Slingshot in ResNet18 on CIFAR-10.

Appendix D. Proofs

D.1. Proof of Theorem 4

In this section, we provide the detailed derivation for Theorem 4, demonstrating how floating-point absorption errors (Softmax Collapse) coupled with the geometry of Neural Collapse induce a deterministic drift in the global weight mean.

D.1.1. PRELIMINARIES AND ASSUMPTIONS

We consider a classification task with K classes, a batch size of B, and a learning rate η. Let $(\boldsymbol{x}, y)$ denote an input-label pair, and $\boldsymbol{h} \in \mathbb{R}^d$ be the feature vector of $\boldsymbol{x}$. The logits are given by $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{h}$, where $\boldsymbol{W} \in \mathbb{R}^{K \times d}$.
We assume the model is in a state of approximate Neural Collapse (NC), satisfying:

- **NC1:** Features collapse to class means: $\boldsymbol{h}_{k,i} \approx \boldsymbol{\mu}_k = \boldsymbol{\mu}_G + \boldsymbol{\mu}_k^*$.
- **NC2:** Centered class means $\boldsymbol{\mu}_k^*$ form a Simplex ETF satisfying $\langle \boldsymbol{\mu}_p^*, \boldsymbol{\mu}_q^* \rangle = -\frac{1}{K-1}\|\boldsymbol{\mu}^*\|^2$ for $p \neq q$.
- **NC3:** Classifiers align with features: $\boldsymbol{W}_k \propto \boldsymbol{\mu}_k^*$, implying $\|\boldsymbol{W}_k\| = \|\boldsymbol{W}\|$ and $\langle \boldsymbol{W}_p, \boldsymbol{W}_q \rangle = -\frac{1}{K-1}\|\boldsymbol{W}\|^2$ for centered weights.

D.1.2. BREAKDOWN OF THE ZERO-SUM CONSTRAINT

The gradient of the Cross-Entropy loss $\mathcal{L}$ with respect to the k-th classifier weight $\boldsymbol{W}_k$ for a single sample $(\boldsymbol{x}, y)$ is:

$$\nabla_{\boldsymbol{W}_k} \mathcal{L} = (\hat{y}_k - y_k)\boldsymbol{h} \tag{13}$$

where y_k is the one-hot label (1 if $k = r$, 0 otherwise) and $\hat{y}_k$ is the softmax probability.

Consider the gradient of the global weight mean $\boldsymbol{W}_G = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{W}_k$. By summing the gradients over all classes:

$$\nabla_{\boldsymbol{W}_G}\mathcal{L} = \frac{1}{K}\sum_{k=1}^{K}(\hat{y}_k - y_k)\boldsymbol{h} = \frac{1}{K}\left(\underbrace{\sum_{k=1}^{K}\hat{y}_k}_{1} - \underbrace{\sum_{k=1}^{K}y_k}_{1}\right)\boldsymbol{h} = 0 \tag{14}$$

In ideal arithmetic, $\boldsymbol{W}_G$ receives zero gradient and remains static.

D.1.3. THE SOFTMAX COLLAPSE (SC) SCENARIO

Under SC, as defined in Section 3, the floating-point absorption error occurs when calculating the loss contribution of the correct class r. Specifically, the term $\hat{y}_r - 1$ is numerically rounded to 0 because the precision limit prevents subtracting the small residual $(1 - \hat{y}_r)$ from the large mantissa of 1. Thus, the summation becomes:

$$\sum_{k=1}^{K}\nabla_{\boldsymbol{W}_k}\mathcal{L} = (\hat{y}_r - y_r)\boldsymbol{h} + \sum_{k\neq r}(\hat{y}_k - 0)\boldsymbol{h} \tag{15}$$

$$\approx 0\cdot\boldsymbol{h} + \sum_{k\neq r}\hat{y}_k\boldsymbol{h} \tag{16}$$

$$= \epsilon\boldsymbol{h} \tag{17}$$

where $\epsilon = \sum_{k\neq r}\hat{y}_k > 0$ represents the total probability mass assigned to incorrect classes.

D.1.4. QUANTIFYING THE RESIDUAL ϵ

Assuming the model is in the NC state, we can approximate $\hat{y}_k$ for incorrect classes ($k \neq r$) using the ETF geometry. The probability is given by $\hat{y}_k \approx \exp(z_k - z_r)$. Using NC3 alignment, for a sample of class r:

- Correct logit: $z_r = \langle \boldsymbol{W}_r, \boldsymbol{h}\rangle \approx \|\boldsymbol{W}\|\|\boldsymbol{\mu}^*\|$
- Incorrect logit ($k \neq r$): $z_k = \langle \boldsymbol{W}_k, \boldsymbol{h}\rangle \approx \|\boldsymbol{W}\|\|\boldsymbol{\mu}^*\|\cos\theta_{ETF} = -\frac{1}{K-1}\|\boldsymbol{W}\|\|\boldsymbol{\mu}^*\|$

Substituting these into the exponent:

$$\hat{y}_k \approx \exp\left(-\frac{1}{K-1}\|\boldsymbol{W}\|\|\boldsymbol{\mu}^*\| - \|\boldsymbol{W}\|\|\boldsymbol{\mu}^*\|\right) \tag{18}$$

$$= \exp\left(-\frac{K}{K-1}\|\boldsymbol{W}\|\|\boldsymbol{\mu}^*\|\right) \tag{19}$$

Summing over the $K-1$ incorrect classes:

$$\epsilon = \sum_{k\neq r}\hat{y}_k \approx (K-1)\exp\left(-\frac{K}{K-1}\|\boldsymbol{W}\|\|\boldsymbol{\mu}^*\|\right) \tag{20}$$

D.1.5. BATCH AGGREGATION AND DRIFT

Finally, we consider the update rule for $\boldsymbol{W}_G$ over a batch of size B with learning rate η. We assume mean reduction for the loss.

$$\Delta \boldsymbol{W}_G = -\eta \left(\frac{1}{B} \sum_{i=1}^{B} \nabla^{(i)}_{\boldsymbol{W}_G} \mathcal{L} \right) \tag{21}$$

Substituting the residual gradient $\sum_k \nabla^{(i)}_{\boldsymbol{W}_k} \mathcal{L} = \epsilon_i \boldsymbol{h}_i$, and noting that $\nabla_{\boldsymbol{W}_G} = \frac{1}{K} \sum_k \nabla_{\boldsymbol{W}_k}$:

$$\Delta \boldsymbol{W}_G = -\frac{\eta}{KB} \sum_{i=1}^{B} \epsilon_i \boldsymbol{h}_i \tag{22}$$

Assuming $\epsilon_i \approx \epsilon$ is roughly constant across the batch (due to NC), we focus on the sum of features $\sum \boldsymbol{h}_i$. Under NC1 (feature collapse) and a class balanced dataset:

$$\mathbb{E}\left[\sum_{i=1}^{B} \boldsymbol{h}_i \right] = \sum_{k=1}^{K} \frac{B}{K} \boldsymbol{\mu}_k = \sum_{k=1}^{K} \frac{B}{K} (\boldsymbol{\mu}_G + \boldsymbol{\mu}_k^*) \tag{23}$$

Since the centered means $\boldsymbol{\mu}_k^*$ form an ETF, $\sum_k \boldsymbol{\mu}_k^* = \mathbf{0}$. Thus:

$$\mathbb{E}\left[\sum_{i=1}^{B} \boldsymbol{h}_i \right] = B \boldsymbol{\mu}_G \tag{24}$$

Substituting this back into the update equation:

$$\mathbb{E}[\Delta \boldsymbol{W}_G] = -\frac{\eta \epsilon}{KB} (B \boldsymbol{\mu}_G) = -\frac{\eta \epsilon}{K} \boldsymbol{\mu}_G \tag{25}$$

Conclusion: Under Softmax Collapse, the global weight vector $\boldsymbol{W}_G$ drifts continuously in the direction opposite to the global feature mean $\boldsymbol{\mu}_G$, proving Theorem 4.

D.2. Proof of Proposition 6

In this section, we provide the derivation for Proposition 6. We analyze the gradient flow back to the feature layer under the regime of Softmax Collapse (SC) and drifted weights.

D.2.1. GRADIENT DECOMPOSITION

The gradient of the loss $\mathcal{L}$ with respect to the feature vector $\boldsymbol{h}$ is given by:

$$\nabla_{\boldsymbol{h}} \mathcal{L} = \boldsymbol{W}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) = \sum_{k=1}^{K} (\hat{y}_k - y_k) \boldsymbol{W}_k \tag{26}$$

We decompose the weight vectors into the global mean and the centered components: $\boldsymbol{W}_k = \boldsymbol{W}_G + \boldsymbol{W}_k^*$. Substituting this into the gradient:

$$\nabla_{\boldsymbol{h}} \mathcal{L} = \sum_{k=1}^{K} (\hat{y}_k - y_k)(\boldsymbol{W}_G + \boldsymbol{W}_k^*) \tag{27}$$

D.2.2. APPLYING SOFTMAX COLLAPSE

Under the SC condition, the gradient contribution from the correct class r vanishes (i.e., the term corresponding to $(\hat{y}_r - 1)$ becomes strictly 0 due to absorption). For incorrect classes $k \neq r$, the label $y_k = 0$. Thus, the summation simplifies to:

$$\nabla_{\boldsymbol{h}} \mathcal{L} \approx \sum_{k \neq r} \hat{y}_k (\boldsymbol{W}_G + \boldsymbol{W}_k^*) \tag{28}$$

We can separate this sum into two components:

$$\nabla_{\boldsymbol{h}} \mathcal{L} = \left(\sum_{k \neq r} \hat{y}_k \right) \boldsymbol{W}_G + \sum_{k \neq r} \hat{y}_k \boldsymbol{W}_k^* \tag{29}$$

Using the definition $\epsilon = \sum_{k \neq r} \hat{y}_k$, the first term simplifies to $\epsilon \boldsymbol{W}_G$.

D.2.3. PROJECTION ONTO THE DRIFT DIRECTION

We now calculate the projection of this gradient onto the direction of the global weight drift $\boldsymbol{W}_G$. The projection operator is defined as:

$$\text{Proj}_{\boldsymbol{W}_G}(\nabla_{\boldsymbol{h}} \mathcal{L}) = \frac{\langle \nabla_{\boldsymbol{h}} \mathcal{L}, \boldsymbol{W}_G \rangle}{\|\boldsymbol{W}_G\|^2} \boldsymbol{W}_G \tag{30}$$

First, we compute the inner product $\langle \nabla_{\boldsymbol{h}} \mathcal{L}, \boldsymbol{W}_G \rangle$:

$$\langle \nabla_{\boldsymbol{h}} \mathcal{L}, \boldsymbol{W}_G \rangle = \left\langle \epsilon \boldsymbol{W}_G + \sum_{k \neq r} \hat{y}_k \boldsymbol{W}_k^*, \boldsymbol{W}_G \right\rangle \tag{31}$$

$$= \epsilon \langle \boldsymbol{W}_G, \boldsymbol{W}_G \rangle + \sum_{k \neq r} \hat{y}_k \langle \boldsymbol{W}_k^*, \boldsymbol{W}_G \rangle \tag{32}$$

We invoke the orthogonality assumption stated in Theorem 4: the global drift is orthogonal to the centered classification subspace, i.e., $\boldsymbol{W}_G \perp \text{span}\{\boldsymbol{W}_k^*\}$. Therefore, $\langle \boldsymbol{W}_k^*, \boldsymbol{W}_G \rangle = 0$ for all k. The inner product simplifies to:

$$\langle \nabla_{\boldsymbol{h}} \mathcal{L}, \boldsymbol{W}_G \rangle = \epsilon \|\boldsymbol{W}_G\|^2 \tag{33}$$

Substituting this back into the projection equation:

$$\text{Proj}_{\boldsymbol{W}_G}(\nabla_{\boldsymbol{h}} \mathcal{L}) = \frac{\epsilon \|\boldsymbol{W}_G\|^2}{\|\boldsymbol{W}_G\|^2} \boldsymbol{W}_G = \epsilon \boldsymbol{W}_G \tag{34}$$

Conclusion: The feature vector $\boldsymbol{h}$ receives a consistent, non-zero gradient component $\epsilon \boldsymbol{W}_G$. Since $\boldsymbol{W}_G$ drifts towards $-\boldsymbol{\mu}_G$ (from Theorem 2), the update step $\boldsymbol{h} \leftarrow \boldsymbol{h} - \eta \nabla_{\boldsymbol{h}} \mathcal{L}$ effectively adds a component aligned with $+\boldsymbol{\mu}_G$, driving Numerical Feature Inflation.

D.3. Proof of Theorem 7: Numerical Feature Inflation

In this section, we provide the rigorous proof for Theorem 7. We define the alignment of features towards the global mean as the inevitable consequence of the feedback loop established in Theorems 4 and 6.

D.3.1. GEOMETRIC ORTHOGONALITY

We first establish the geometric relationship between the global mean and the classification subspace.

Lemma 9 (Orthogonality of Global Mean) *For a ReLU network in the NC state with K balanced classes, the global mean $\boldsymbol{\mu}_G = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\mu}_k$ is orthogonal to the centered class mean subspace $\mathcal{S}_{NC} = span\{\boldsymbol{\mu}_k^*\}_{k=1}^K$.*

Proof Dang et al. [9] have proved that for a ReLU network in the NC state, the uncentered class means $\boldsymbol{\mu}_k$ are mutually orthogonal ($\boldsymbol{\mu}_p^T\boldsymbol{\mu}_q = 0$ for $p \neq q$) and reside in the first orthant.
The dataset is class balanced so that $\|\boldsymbol{\mu}_k\| = R$ for all k. Since the uncentered class means $\{\boldsymbol{\mu}_k\}$ form an orthogonal basis in the feature space (scaled by R), we can compute the squared norm of the global mean:

$$\|\boldsymbol{\mu}_G\|^2 = \left\|\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\mu}_k\right\|^2 = \frac{1}{K^2}\sum_{k=1}^{K}\|\boldsymbol{\mu}_k\|^2 = \frac{1}{K^2}\cdot KR^2 = \frac{R^2}{K} \tag{35}$$

Next, we compute the projection of any class mean $\boldsymbol{\mu}_k$ onto the global mean:

$$\langle\boldsymbol{\mu}_G, \boldsymbol{\mu}_k\rangle = \frac{1}{K}\sum_{j=1}^{K}\langle\boldsymbol{\mu}_j, \boldsymbol{\mu}_k\rangle = \frac{1}{K}\|\boldsymbol{\mu}_k\|^2 = \frac{R^2}{K} \tag{36}$$

noting that cross-terms $\langle\boldsymbol{\mu}_j, \boldsymbol{\mu}_k\rangle$ vanish for $j \neq k$. The centered class means are defined as $\boldsymbol{\mu}_k^* = \boldsymbol{\mu}_k - \boldsymbol{\mu}_G$. Calculating the inner product with $\boldsymbol{\mu}_G$:

$$\langle\boldsymbol{\mu}_G, \boldsymbol{\mu}_k^*\rangle = \langle\boldsymbol{\mu}_G, \boldsymbol{\mu}_k - \boldsymbol{\mu}_G\rangle = \langle\boldsymbol{\mu}_G, \boldsymbol{\mu}_k\rangle - \|\boldsymbol{\mu}_G\|^2 = \frac{R^2}{K} - \frac{R^2}{K} = 0 \tag{37}$$

Since $\boldsymbol{\mu}_G$ is orthogonal to every basis vector $\boldsymbol{\mu}_k^*$ of the subspace $\mathcal{S}_{NC}$, it follows that $\boldsymbol{\mu}_G \perp \text{span}\{\boldsymbol{\mu}_k^*\}$. ■

D.3.2. ASYMPTOTIC ALIGNMENT VIA COUPLED DYNAMICS

We treat the evolution of the global weight mean $\boldsymbol{W}_G$ and global feature mean $\boldsymbol{\mu}_G$ as a coupled linear dynamical system in $\mathbb{R}^d$.
We derive the update rules for the vectors.

- From Theorem 4, the weight mean update is:

$$\boldsymbol{W}_G^{(t+1)} = \boldsymbol{W}_G^{(t)} - \alpha\boldsymbol{\mu}_G^{(t)} \tag{38}$$

  where $\alpha = \frac{\eta\epsilon}{K} > 0$.

- From Proposition 6, the feature gradients contain a component parallel to the current weight mean. Assuming the ETF structure leads to the cancellation of orthogonal components when averaged over a batch (i.e., $\sum_k \boldsymbol{W}_k^* \approx 0$), the update to the feature mean is dominated by:

$$\boldsymbol{\mu}_G^{(t+1)} = \boldsymbol{\mu}_G^{(t)} - \beta\boldsymbol{W}_G^{(t)} \tag{39}$$

  where $\beta = \eta\epsilon > 0$.

Let $\boldsymbol{u}^{(t)} = \begin{bmatrix} \boldsymbol{W}_G^{(t)} \\ \boldsymbol{\mu}_G^{(t)} \end{bmatrix}$ be the state vector in the product space $\mathbb{R}^{2d}$. The dynamics are governed by the block matrix $\mathbf{M}$:

$$\begin{bmatrix} \boldsymbol{W}_G^{(t+1)} \\ \boldsymbol{\mu}_G^{(t+1)} \end{bmatrix} = \begin{bmatrix} I & -\alpha I \\ -\beta I & I \end{bmatrix} \begin{bmatrix} \boldsymbol{W}_G^{(t)} \\ \boldsymbol{\mu}_G^{(t)} \end{bmatrix} \tag{40}$$

where I is the $d \times d$ identity matrix.
$\mathbf{M}$ has two eigenvalues:

$$\lambda_1 = 1 + \sqrt{\alpha\beta} = 1 + \eta\epsilon/\sqrt{K} > 1 \tag{41}$$

$$\lambda_2 = 1 - \sqrt{\alpha\beta} = 1 - \eta\epsilon/\sqrt{K} < 1 \tag{42}$$

Solving $\mathbf{M}\boldsymbol{v} = \lambda\boldsymbol{v}$:

1. For λ_1:

$$v_W - \alpha v_\mu = (1 + \sqrt{\alpha\beta})v_W \implies -\alpha v_\mu = \sqrt{\alpha\beta} v_W \implies \boldsymbol{W}_G = -\frac{1}{\sqrt{K}}\boldsymbol{\mu}_G \tag{43}$$

2. For λ_2, $\boldsymbol{W}_G = \frac{1}{\sqrt{K}}\boldsymbol{\mu}_G$.

The state vector can be expressed as a linear combination of the eigenvectors:

$$\boldsymbol{u}^{(t)} = c_1\lambda_1^t \begin{bmatrix} -\frac{1}{\sqrt{K}}\boldsymbol{\mu}_G \\ \boldsymbol{\mu}_G \end{bmatrix} + c_2\lambda_2^t \begin{bmatrix} \frac{1}{\sqrt{K}}\boldsymbol{\mu}_G \\ \boldsymbol{\mu}_G \end{bmatrix} \tag{44}$$

As $t \to \infty$, the term with λ_1 dominates since $\lambda_1 > 1$ and $\lambda_2 < 1$:

$$\lim_{t\to\infty} \boldsymbol{u}^{(t)} \propto \begin{bmatrix} -\frac{1}{\sqrt{K}}\boldsymbol{\mu}_G \\ \boldsymbol{\mu}_G \end{bmatrix} \tag{45}$$

This leads to the asymptotic relationship:

$$\lim_{t\to\infty} \boldsymbol{W}_G^{(t)} \approx -\frac{1}{\sqrt{K}}\boldsymbol{\mu}_G^{(t)} \tag{46}$$

Therefore, the vectors asymptotically align with the relationship defined by the dominant eigenvector:

$$\lim_{t\to\infty} \cos(\boldsymbol{W}_G^{(t)}, \boldsymbol{\mu}_G^{(t)}) = -1 \tag{47}$$

And the norms grow exponentially:

$$\|\boldsymbol{W}_G^{(t)}\| \propto (1 + \frac{\eta\epsilon}{\sqrt{K}})^t \tag{48}$$

$$\|\boldsymbol{\mu}_G^{(t)}\| \propto (1 + \frac{\eta\epsilon}{\sqrt{K}})^t \tag{49}$$

D.3.3. NUMERICAL FEATURE INFLATION ($\mathcal{NFI}$)

Finally, we prove the second claim: the features condense toward $\boldsymbol{\mu}_G$. We decompose the feature update $\Delta \boldsymbol{h}$ into a parallel component along $\boldsymbol{\mu}_G$ and a perpendicular component in the subspace $\mathcal{S}_{NC}$. From Proposition 6, the gradient component along the drift direction is:

$$\boldsymbol{g}_{\|} = \text{Proj}_{\boldsymbol{\mu}_G}(\nabla_{\boldsymbol{h}} \mathcal{L}) = \epsilon \boldsymbol{W}_G \tag{50}$$

Given the result of Section D.3.2, for large t, we can approximate $\boldsymbol{W}_G \approx -\|\boldsymbol{W}_G\| \frac{\boldsymbol{\mu}_G}{\|\boldsymbol{\mu}_G\|}$. The gradient update step $h \leftarrow h - \eta \nabla_h \mathcal{L}$ adds a component:

$$\Delta h_{\|} = -\eta(\epsilon \boldsymbol{W}_G) \approx \eta \epsilon \|\boldsymbol{W}_G\| \frac{\boldsymbol{\mu}_G}{\|\boldsymbol{\mu}_G\|} \tag{51}$$

For the perpendicular component, consider NC3′, the gradient is the weighted sum of centered ETF weights:

$$\boldsymbol{g}_{\perp} = \sum_{k \neq r} \hat{y}_k \boldsymbol{W}_k^* \tag{52}$$

In the ETF configuration, the vectors $\boldsymbol{W}_k^*$ sum to zero. The weighted sum $\boldsymbol{g}_{\perp}$ represents the residual interference. The magnitude of the parallel update is driven by the scalar sum of probabilities $\epsilon = \sum \hat{y}_k$, whereas the perpendicular update is a vector sum of randomly oriented Simplex vectors weighted by $\hat{y}_k \approx \epsilon/(K-1)$. Comparing the growth rates:

$$\frac{\|\Delta \boldsymbol{h}_{\|}\|}{\|\Delta \boldsymbol{h}_{\perp}\|} = \frac{\|\epsilon \boldsymbol{W}_G\|}{\|\sum_{k \neq r} \hat{y}_k \boldsymbol{W}_k^*\|} \approx \frac{\epsilon \|\boldsymbol{W}_G\|}{\sqrt{K-1} \frac{\epsilon}{K-1} \|\boldsymbol{W}^*\|} \propto \frac{\|\boldsymbol{W}_G\|}{\|\boldsymbol{W}^*\|} \tag{53}$$

As $\|\boldsymbol{W}_G\|$ grows exponentially due to the feedback loop while $\|\boldsymbol{W}^*\|$ (representing the fixed classification structure) grows linearly or remains bounded relative to the drift, the parallel component dominates. Therefore,

$$\lim_{t \to \infty} \frac{\|\boldsymbol{h}_{\perp}\|}{\|\boldsymbol{h}_{\|}\|} \to 0 \implies \lim_{t \to \infty} \cos(\boldsymbol{h}_t, \boldsymbol{\mu}_G) \to 1 \tag{54}$$

This confirms that the feature space effectively collapses into a rank-1 subspace aligned with the global mean.

D.4. Proof of Theorem 8

In this section, we provide the detailed proof for Theorem 8, demonstrating that the maximum eigenvalue of the Hessian matrix with respect to model parameters converges to zero as the model approaches the interpolation solution.

The Hessian matrix of the loss with respect to parameters, $H_\theta = \nabla_\theta^2 \mathcal{L}$, can be decomposed into the generalized Gauss-Newton (GGN) term and the residual term [27]:

$$H_\theta = \underbrace{J^T H_z J}_{G(\theta)} + \underbrace{\sum_{k=1}^{K} (\nabla_z \mathcal{L})_k \nabla_\theta^2 z_k}_{E(\theta)} \tag{55}$$

where $J = \nabla_\theta z \in \mathbb{R}^{K \times d}$ is the Jacobian matrix of the logits, and $H_z = \nabla_z^2 \mathcal{L} \in \mathbb{R}^{K \times K}$ is the Hessian of the loss with respect to the logits. We analyze the convergence of these two terms separately.

D.4.1. ANALYSIS OF THE GAUSS-NEWTON TERM $G(\theta)$.

For Cross-Entropy loss with Softmax activation, the Hessian with respect to logits is given explicitly by:

$$H_z = \text{diag}(\hat{y}) - \hat{y}\hat{y}^T \tag{56}$$

where $\hat{y} = \text{softmax}(z)$ is the predicted probability vector. Since H_z is positive semi-definite, its spectral norm (maximum eigenvalue) is bounded by its trace:

$$\|H_z\|_2 = \lambda_{\max}(H_z) \leq \text{tr}(H_z) = \sum_{k=1}^{K}(\hat{y}_k - \hat{y}_k^2) = 1 - \|\hat{y}\|_2^2 \tag{57}$$

In the late stage of training, assuming the model enters the interpolation regime, the prediction $\hat{y}$ converges to the distinct one-hot label vector y. Since $\|y\|_2^2 = 1$, we have:

$$\lim_{\hat{y}\to y} \|H_z\|_2 \leq \lim_{\hat{y}\to y}(1 - \|\hat{y}\|_2^2) = 0 \tag{58}$$

Using the sub-multiplicativity of the matrix norm, the norm of the GGN term is bounded by:

$$\|G(\theta)\|_2 = \|J^T H_z J\|_2 \leq \|J\|_2^2 \|H_z\|_2 \tag{59}$$

Assuming the Jacobian J is bounded in the local convergence region (i.e., $\exists M_1 > 0$ such that $\|J\|_2 \leq M_1$), it follows that:

$$\lim_{\hat{y}\to y} \|G(\theta)\|_2 = 0 \tag{60}$$

D.4.2. ANALYSIS OF THE RESIDUAL TERM $E(\theta)$.

The gradient of the loss with respect to logits is the prediction error: $\nabla_z \mathcal{L} = \hat{y} - y$. The residual term can be expanded as:

$$E(\theta) = \sum_{k=1}^{K}(\hat{y}_k - y_k)\nabla_\theta^2 z_k \tag{61}$$

Applying the triangle inequality:

$$\|E(\theta)\|_2 \leq \sum_{k=1}^{K} |\hat{y}_k - y_k| \cdot \|\nabla_\theta^2 z_k\|_2 \tag{62}$$

Assume the network function $z(\theta)$ is C^2 continuous and its second-order derivatives are locally bounded (i.e., $\exists M_2 > 0$ such that $\|\nabla_\theta^2 z_k\|_2 \leq M_2$). As $\hat{y} \to y$, the error term $|\hat{y}_k - y_k| \to 0$ for all classes k. Consequently:

$$\lim_{\hat{y}\to y} \|E(\theta)\|_2 = 0 \tag{63}$$

Conclusion. Combining the bounds for both terms:

$$\|H_\theta\|_2 \leq \|G(\theta)\|_2 + \|E(\theta)\|_2 \tag{64}$$

Since both $\|G(\theta)\|_2 \to 0$ and $\|E(\theta)\|_2 \to 0$ as the model converges to the interpolation solution, we conclude that:

$$\lim_{\hat{y}\to y} \lambda_{\max}(H_\theta) = 0 \tag{65}$$

D.4.3. LABEL SMOOTHING LEADS TO NON-VANISHING HESSIAN

In Theorem 8, we proved that under standard Cross-Entropy loss with hard targets, the maximum eigenvalue of the Hessian $\lambda_{max} \to 0$ because the predicted probability vector $\hat{y}$ converges to a one-hot vector. Here, we demonstrate that Label Smoothing fundamentally alters this asymptotic behavior.

Definition 10 (Label Smoothing) *Let y be the one-hot label for the correct class r. Label Smoothing replaces y with a soft target y^{LS}:*

$$y_k^{LS} = \begin{cases} 1-\alpha & \text{if } k = r \\ \frac{\alpha}{K-1} & \text{if } k \neq r \end{cases} \tag{66}$$

where $\alpha \in (0,1)$ is the smoothing parameter.

Proof Consider the Gauss-Newton decomposition of the Hessian as defined in Equation (55). The core component governing the scale of the Hessian eigenvalues is the Hessian of the loss with respect to logits, $H_z = \text{diag}(\hat{y}) - \hat{y}\hat{y}^T$.
Unlike standard training where logits diverge to infinity to minimize loss (driving $\hat{y} \to y$), under Label Smoothing, the global minimum is achieved at finite logit values where the predicted distribution matches the soft target exactly:

$$\lim_{t\to\infty} \hat{y} = y^{LS} \tag{67}$$

Consequently, for the correct class r, the prediction $\hat{y}_r$ converges to $1-\alpha$ rather than 1.
The trace of H_z represents the sum of eigenvalues (variances of the categorical distribution):

$$\text{tr}(H_z) = \sum_{k=1}^{K} (\hat{y}_k - \hat{y}_k^2) = 1 - \|\hat{y}\|_2^2 \tag{68}$$

Substituting the limit $\hat{y} \to y^{LS}$:

$$\lim_{\hat{y}\to y^{LS}} \text{tr}(H_z) = 1 - \left((1-\alpha)^2 + (K-1)\left(\frac{\alpha}{K-1}\right)^2 \right) \tag{69}$$

$$= 1 - \left((1-\alpha)^2 + \frac{\alpha^2}{K-1} \right) \tag{70}$$

$$= 2\alpha - \alpha^2 - \frac{\alpha^2}{K-1} \tag{71}$$

$$= \alpha \left(2 - \alpha \left(1 + \frac{1}{K-1} \right) \right) \tag{72}$$

Since $\alpha \in (0,1)$, the term $2-\alpha\left(1+\frac{1}{K-1}\right) > 0$. Therefore, the trace of H_z converges to a positive constant:

$$\lim_{\hat{y}\to y^{LS}} \text{tr}(H_z) = C_{LS} > 0 \tag{73}$$

Since H_z is positive semi-definite, its maximum eigenvalue is bounded below by the average eigenvalue:

$$\lambda_{\max}(H_z) \geq \frac{\text{tr}(H_z)}{K} = \frac{C_{LS}}{K} > 0 \tag{74}$$

Thus, the Gauss-Newton term satisfies:

$$\|G(\theta)\|_2 = \|J^T H_z J\|_2 \geq \|J\|_2^2 \cdot \frac{C_{LS}}{K} > 0 \tag{75}$$

Assuming the Jacobian J remains bounded away from zero, we conclude that:

$$\lim_{t\to\infty} \lambda_{\max}(H_\theta) \geq \lim_{t\to\infty} \|G(\theta)\|_2 > 0 \tag{76}$$

Therefore, under Label Smoothing, the maximum eigenvalue of the Hessian does not vanish, confirming that $\lambda_{\max}(H_\theta)$ converges to a positive constant rather than zero. ■

Appendix E. Experimental Details

All of the experiments were conducted on NVIDIA RTX5090 GPUs. For modular and CIFAR-10 experiments, each run takes approximately 1 hour. For LLM experiments, each run takes approximately 12 hours.

E.1. Modular Arithmetic

For the modular arithmetic experiments, we focus on the task of modular division. Given a prime $p = 97$, the model is trained to predict $c = (a \cdot b^{-1}) \pmod p$ for all pairs $a \in \{0, \ldots, p-1\}$ and $b \in \{1, \ldots, p-1\}$.

Dataset and Task. The dataset consists of $p(p-1)$ samples. Each input is formatted as a sequence of four tokens: $[a, \text{OP}, b, =]$, where OP represents the division operator. The vocabulary size is $p+2$ to account for the operator and the equal sign. We employ a 50/50 training and validation split.

Model Architectures. We evaluate two primary architectures: a decoder-only Transformer and a deep MLP.

- **Transformer** [43]: A 2-layer decoder-only Transformer. Each layer consists of a multi-head self-attention mechanism with $n_{head} = 4$ and a feed-forward network (FFN) with a hidden dimension of $4 \times d_{model}$. The model uses $d_{model} = 128$. In our primary unregularized experiments, Layer Normalization is disabled to isolate the Slingshot dynamics.
- **MLP**: A 6-layer fully connected MLP. The architecture comprises 5 hidden layers with a width of 512 and ReLU activations, followed by a final linear classification layer. The input is provided as flattened one-hot encodings of the token sequence. Bias terms are included in all linear layers.

Training and Optimization. All models are trained using the Adam optimizer with a cross-entropy loss function. No explicit regularization is applied during the baseline runs.

Table 2: Hyperparameters for Modular Arithmetic Experiments

Hyperparameter	**Value**
Prime p	97
Optimizer	Adam
Learning Rate	10^{-3}
Warmup Steps	10
β_1, β_2	0.9, 0.999
ε (Adam)	10^{-8}
Weight Decay	0.0
Batch Size	512
Total Steps	100,000
Numerical Precision	Float32
Random Seed	42

E.2. Image Classification (CIFAR-10)

For image classification tasks, we evaluate the Slingshot mechanism using the CIFAR-10 dataset.

Dataset and Preprocessing Consistent with prior work on the Slingshot mechanism, we utilize the CIFAR-10 dataset. For evaluation, we use a fixed test set of 1,000 samples. Images are converted to tensors using a standard `transforms.ToTensor()` operation without additional data augmentation to ensure the optimization dynamics remain deterministic and focused on the numerical properties of the loss landscape.

Model Architectures We conduct experiments across four distinct architectures to verify the universality of $\mathcal{NFI}$:

- **MLP**: A 6-layer fully connected MLP. The architecture comprises 5 hidden layers with a width of 512 and ReLU activations, followed by a final linear classification layer.
- **ResNet18** [16]: A modified ResNet18 architecture optimized for the 32×32 resolution of CIFAR-10. The first convolutional layer is replaced with a 3×3 kernel (stride 1, padding 1) and the initial max-pooling layer is removed to prevent excessive spatial downsampling in early layers.
- **VGG11** [36]: A standard VGG11 architecture, evaluated both with and without Batch Normalization. The classification head is simplified to a single linear layer to match the feature-classifier interface of our theoretical model.
- **ViT** [10]: A small-scale Vision Transformer with approximately 10M parameters. It utilizes a patch size of 4, an embedding dimension of 256, 12 transformer blocks, and 8 attention heads.

Training and Optimization All models are trained for 10,000 epochs using the Adam optimizer. To isolate the effects of $\mathcal{NFI}$, we set weight decay to zero and use a standard cross-entropy loss without label smoothing.

Table 3: Hyperparameters for CIFAR-10 Image Classification

Hyperparameter	**Value**
Optimizer	Adam
Learning Rate	10^{-3}
β_1, β_2	0.9, 0.95
ε (Adam)	10^{-8}
Weight Decay	0.0
Label Smoothing	0.0
Batch Size	2048 for ViT, Full Batch for others
Total Epochs	10,000
Numerical Precision	Float32
Random Seed	42

E.3. Large Language Models

For the language modeling experiments, we use a nanoGPT-style decoder-only Transformer. We follow the experimental setup of Stollenwerk et al. [39] as closely as possible, including the optimizer, architecture, precision, training schedule, and data-processing settings. We do not change the training recipe, and only add additional measurements needed for our analysis, including the fraction of tokens in Softmax Collapse, the mean and norm of output logits, the global mean of the output embedding $\boldsymbol{W}_G$, and the effect of removing $\boldsymbol{W}_G$ during evaluation or training. The main hyperparameters are summarized in Table 4.

Table 4: Hyperparameters for the language modeling experiments.

Hyperparameter	**Value**
optimizer	AdamW
β_1	0.9
β_2	0.95
$\varepsilon_{\text{Adam}}$	10^{-8}
weight decay	0.0
gradient clipping	1.0
dropout	0.0
weight tying	false
QK-layernorm	yes
bias	no
learning rate schedule	cosine decay
minimum learning rate	10^{-5}
normalization	LayerNorm
precision	BF16
positional embedding	RoPE
vocabulary size	50304
hidden activation	SwiGLU
sequence length	2048
batch size (samples)	64
batch size (tokens)	131072
training length	100000 steps $\approx$ 13.1B tokens
warmup	5000 steps $\approx$ 0.7B tokens
embedding initialization	Normal with standard deviation $1/\sqrt{d}$
weight initialization	Xavier with average of fan_in and fan_out