Outliers with Opposing Signals Have an Outsized Effect on Neural Network Optimization

Elan Rosenfeld
Carnegie Mellon University
elan@cmu.edu

Andrej Risteski
Carnegie Mellon University
aristesk@andrew.cmu.edu

Abstract

We identify a new phenomenon in neural network optimization which arises from the interaction of depth and a particular heavy-tailed structure in natural data. Our result offers intuitive explanations for several previously reported observations about network training dynamics and demonstrates how a small number training points can have an unusually large effect on a network's optimization trajectory and predictions. Experimentally, we demonstrate the significant influence of paired groups of outliers in the training data with strong *opposing signals*: consistent, large magnitude features which dominate the network output and occur in both groups with similar frequency. Due to these outliers, early optimization enters a narrow valley which carefully balances the opposing groups; subsequent sharpening causes their loss to rise rapidly, oscillating between high on one group and then the other, until the overall loss spikes. We complement these experiments with a theoretical analysis of a two-layer linear network on a simple model of opposing signals. Our finding enables new qualitative predictions of behavior during and after training which we confirm experimentally. It also provides a new lens through which to study how specific data influence the learned parameters. The full version of this work can be found at https://arxiv.org/abs/2311.04163.

1 Introduction

There is a steadily growing list of intriguing properties of neural network (NN) optimization which are not readily explained by prior tools from optimization. Likewise, there exist varying degrees of understanding of the mechanistic causes for each—but the evidence is not always entirely convincing, and there is little theoretical understanding. What these observations imply about the effect of training data on the behavior of the network at the end of training is also unclear, but what is apparent is that any commonality will be a valuable tool for further investigation.

In this work, we identify a phenomenon in NN optimization which offers a new perspective on many of these prior observations and which we hope will contribute to a deeper understanding of how they may be connected. While we do not give a complete explanation, we present strong qualitative and quantitative evidence which suggests a more coherent picture of their origin, offering a single high-level idea on the influence of specific training samples which naturally fits into several existing narratives. Specifically, we demonstrate the prevalence of paired groups of outliers which have a significant influence on a network's optimization dynamics. These groups are characterized by the inclusion of one or more (relatively) large magnitude features that dominate the network's output at random initialization and throughout most of training. In addition to their magnitude, the other distinctive property of these features is that they provide large, consistent, and *opposing* gradients, in that they (mostly) cannot be used to reduce loss on one group without increasing loss on the other; because of this structure, we refer to them as *Opposing Signals*. Some of these features are genuine signal. Others are irrelevant to the target task but, crucially, they do meaningfully correlate with

ATTRIB: Workshop on Attributing Model Behavior at Scale at NeurIPS 2023.

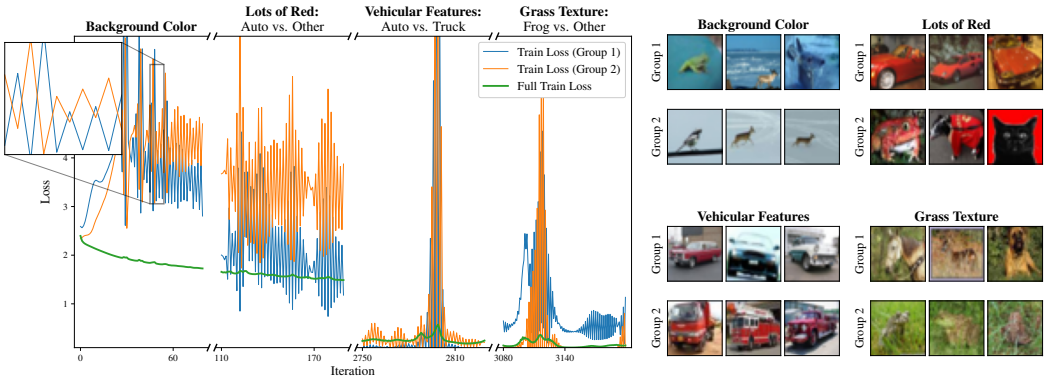


Figure 1: **Training dynamics of neural networks are heavily influenced by outliers with opposing signals.** We plot the overall loss of a ResNet-18 trained with GD on CIFAR-10, plus the losses of a small but representative subset of outlier groups. These groups have consistent *opposing signals* (e.g., wheels and headlights sometimes means `car` and sometimes `truck`). Throughout training, losses on these groups oscillate with growing amplitude—this oscillation has an obvious correspondence to the short term spikes in overall training loss and, in particular, appear to be the direct cause of the "edge-of-stability" phenomenon.

it—for example, a bright blue sky background does not determine the label of a CIFAR image, but it does most often occur in images of planes.

Opposing signals are most easily understood with an example, which we will give along with a brief outline of their effect on training dynamics; a more detailed description is presented in Section 3. Fig. 1 depicts the training loss of a ResNet-18 trained with full-batch gradient descent (GD) on CIFAR-10, along with a few dominant outlier groups and their respective losses. In the early stages of training, the network enters a narrow valley in weight space which carefully balances the pairs' opposing gradients; subsequent sharpening of the loss landscape [19, 5] causes the network to oscillate with growing magnitude along particular axes, upsetting this balance. Returning to our example of a sky background, one step results in the class `plane` being assigned greater probability for all images with sky, and the next will reverse that effect. In essence, the "sky $=$ `plane`" subnetwork grows and shrinks. The direct result of this oscillation is that the network's loss on images of planes with a sky background will alternate between sharply increasing and decreasing with growing amplitude, with the exact opposite occurring for images of *non*-planes with sky. As these pairs represent a small fraction of the data, this behavior is not immediately apparent from the overall training loss—but eventually, it progresses far enough that the overall loss spikes. As there is an obvious direct correspondence between these two events throughout, we conjecture that opposing signals are the direct cause of the *edge of stability* phenomenon [5].

We repeat this experiment across a range of vision architectures: though the precise groups and order of appearance change, the pattern occurs consistently. We also verify this behavior for transformers on next-token prediction and small ReLU MLPs on simple 1D functions; we give some examples of opposing signals in text in Appendix B. However, we rely on images for exposition because it offers the clearest intuition. To isolate this effect, most of our experiments use GD—but we observe similar patterns during SGD, which we present in Section 4. Though we do not give a *precise* description of the effect of these outliers, we present our current best understanding, with supporting experiments, of the underlying mechanism behind our findings—in particular, we argue that it is a consequence of depth and steepest descent methods. We complement this discussion with a toy example and an analysis of a two-layer linear net on a simple model. Notably, though rudimentary, our explanation enables concrete predictions of NN behavior—as well as qualitative descriptions of how a small subsets of data influence its predictions throughout training—which we confirm experimentally. More generally, it provides a new lens through which to study how specific data influence optimization, which we confirm applies to stochastic optimization in a comparison between Adam and SGD.

Connections to attribution methods. We believe our result offers new ways to understand the effectiveness of various model attribution or interpretability algorithms and that it can enable more informed development of future methods. For example, opposing signals suggest one possible reason

why methods which analyze the influence of *individual points* (or small subsets) during NN training can successfully learn meaningful global attribution structure [46, 15, 13]. Furthermore, these signals' existence implies that deep networks learn features from training data in unexpected ways: in addition to the obvious scenario where a feature is downweighted because it is not useful at all, our analysis suggests that networks will also learn to downweight features if they are *very useful in different ways for different subpopulations,* but only later in training. Thus, instead of removing a datapoint, *adding* one with a carefully designed opposing feature may have a similar effect. Finally, we wonder whether the enormous influence of these large magnitude features might help explain the surprising accuracy of linear influence estimation methods [22, 15], as the effect of adding or removing a single point might move a distance along the "axis" for its dominating features (as depicted in Fig. 2), allowing a simple method based on Taylor expansions to make accurate predictions.

We also see possible connections between opposing signals and other existing phenomena in NN optimization, including *grokking* [39], *catapulting* [25, 47], *simplicity bias* [48], and Sharpness-Aware Minimization [9]. We discuss these connections along with further related work in Appendix A.

2 Setup and Experimental Methodology

Though their influence on aggregate metrics is non-obvious, identifying opposing signals is straightforward. When training a network with GD, we track its loss on each individual training point. For a given iteration, we identify the training points whose loss exhibited the most positive and most negative change in the preceding step (there is large overlap between these sets in successive steps). This set will sometimes contain multiple opposing signals, which we distinguish via visual inspection. We emphasize that it would not be correct to describe this process as cherry-picking: these signals consistently obey the maxim "I know it when I see it". To demonstrate this fact, Appendix J contains the pre-inspection samples for a ResNet-18, VGG-11, and a Vision Transformer at several training steps and for multiple seeds; we believe the implied groupings are immediate.

Given how these samples were selected, several other characterizations seem relevant. For instance, maximal one-step loss change is often a reasonable proxy for maximum gradient norm; we could also consider the largest eigenvalue of the loss of the *individual point*, or how much curvature it has in the direction of the overall loss's top eigenvector. In Fig. 27 in the Appendix we track these metrics for several opposing group pairs and find that they are consistently much larger than that of random samples from the training set.

The meaning of "outlier". For visual clarity, Fig. 1 includes only the pair which is most dominant in its respective training phase. But we note that this pattern occurs simultaneously for many different opposing signals and at multiple scales throughout training. Thus, what qualifies as an outlier may vary with different stages of training. Further, the samples we identify lie in the heavy tail of inputs with respect to the model's internal representations (and the label), not the input space itself. This means that the concept of an outlier is also a property of the architecture.

3 Understanding the Effect of Opposing Signals

Our eventual goal will be to derive actionable insights from this finding. To do this, it is necessary to gain a better understanding of *how* these opposing signals lead to the observed behavior. In this section we give a simplified "mental picture" which serves our current understanding this process: first a general discussion of why opposing signals are so influential, followed by a more mechanistic description with a toy example. This explanation is intentionally high-level, but we will eventually see how it gives concrete predictions of specific behaviors, which we then verify on real networks.

3.1 Why These Outliers are so Influential

Consider a NN being trained to take inputs x and minimize loss in predicting a target y. Most information in x will be unneeded—particularly with depth and high-dimensional inputs, only a small fraction will be propagated to the last linear layer [14]. At initialization, the network's features

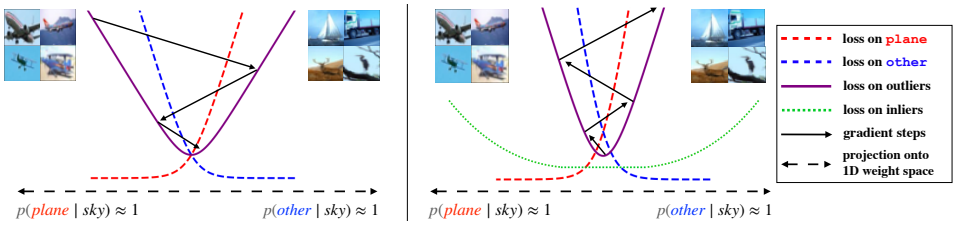


Figure 2: **A toy example illustrating the effect of opposing signals.** We project the loss to the hypothetical weight-space dimension "sky = `plane`", which has a steep gradient. **Left:** Early optimization approaches the minimum, balancing the two opposing losses. Progress continues through this valley, further aligning subnetworks and growing the linear head. **Right**: The valley sharpens and the iterates diverge. Because these images are a small minority and the others are less sensitive to this axis, the train loss is not noticeably affected at first. Eventually either (a) the outlier gradients' growth forces the network to downweight "sky", flattening the valley and returning to the first phase; or (b) the iterates diverge enough that the weights "catapult" to a different basin.

will typically be dominated by noise.[1] Training then modifies the weights to amplify meaningful signal while downweighting this noise, by aligning adjacent layers' singular values [43, 31] such that sensitivity to the relevant signal grows. This sensitivity can be measured, for example, by the spectral norm of the input-output Jacobian, which grows during training [27].

Observe that as the signal-extracting components of the network align, the network's sensitivity to changes in the *way* it processes inputs grows as well. Hypothetically, a small weight perturbation could massively amplify the effect of noise by redirecting it to the aligning subnetwork, or by slightly changing how the last layer uses it. The increase of this sensitivity represents precisely the growth of loss Hessian spectrum, with the strength of this effect increasing with depth [8, 31]. Crucially, this process also depends on the magnitude and structure of the input variation. It is clear that noise with greater variance means greater potential corruption of signal. But we also observe that *genuine signals which oppose each other* will cause more consistent sharpening than traditionally envisioned "random noise". To see why, note that it will always be beneficial to reduce the influence of noise which is independent of the target—in contrast, any feature which is not "correct" but is also not independent of the target may be retained or even further amplified, since it could be the most immediate way of minimizing loss before better features are learned. As a concrete example, observe that a randomly initialized network will be very far from any features required for the subtle task of distinguishing birds from planes. But it *will* be able to capture the presence of sky, which is very useful for separating planes from other classes. Thus, any method which attempts to minimize loss as fast as possible (e.g., steepest descent) will first learn to use these dominant features, discouraging the network from downweighting them. This description is somewhat abstract—to gain a more mechanistic understanding, we illustrate the precise dynamics on a toy example.

3.2 Illustrating with a Hypothetical Example of Gradient Descent

Consider the global loss landscape of a neural network: this is the function which describes how the loss changes as we move through parameter space. Suppose we identify a direction in this space which corresponds to the network's use of the "sky" feature to predict `plane` versus some other class. That is, we will imagine that whenever the input image includes a bright blue background, moving the parameters in one direction increases the logit of the `plane` class and decreases the others, and vice-versa. We will also decompose this loss—we consider separately the loss on images of planes and the loss on all other images.

Fig. 2 depicts this heavily simplified scenario. Early in training, optimizing this network with GD will rapidly move towards the minimum. In particular, until the network learns to use more relevant signal, the direction of steepest descent will lead to a network which predicts the likelihood $p(\text{class} \mid \text{sky})$ whenever sky is present. Eventually, the gradient will no longer be dominated by this direction and will instead point "through the valley" [52]. However, until the network separates out the "sky" feature, this will simultaneously cause the sky feature to grow in magnitude. It will also cause an increase in the

[1] In this discussion we use the term "noise" informally. We refer not necessarily to pure randomness, but more generally to input variation which is least useful in predicting the target.

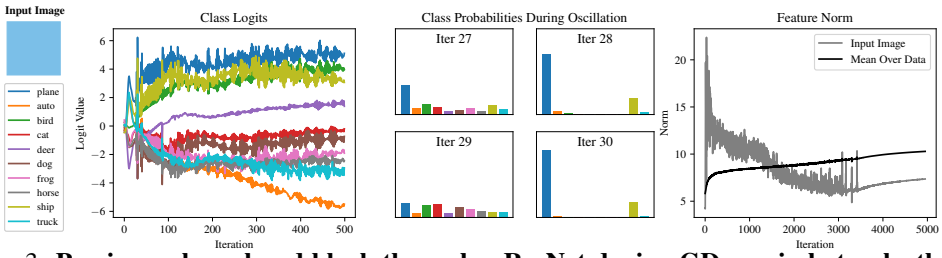


Figure 3: **Passing a sky-colored block through a ResNet during GD precisely tracks the predictions of our toy example. Left:** In the first phase, the network rapidly learns to use the sky, amplifying the feature and sharpening the loss. **Middle:** During oscillation, gradient steps alternate along the axis "sky = `plane`" (and a bit `ship`). **Right:** Oscillation originally amplifies the sky input; the network then slowly downweights this feature and learns to use other signal.

potential influence of this feature were the linear head to be selectively perturbed. Both these factors cause further sharpening. Continued optimization will oscillate across the minimum with growing magnitude, but this growth may not be immediately apparent. Returning to the loss decomposition, we see that in addition to increasing in average magnitude, these oscillations will cause the losses to grow and *alternate*, with one group having high loss and then the other. Eventually the outliers' loss increases sufficiently and the overall loss spikes, either flattening the valley and returning to the first phase, or "catapulting" to a different basin [50, 25, 47]. This is the phenomenon depicted in Fig. 1.

Though this explanation lacks precise details, it does enable concrete predictions of network behavior during training. Fig. 3 tracks the predictions of a ResNet-18 on a synthetic image with all bright blue pixels. We see exactly the described behavior—initial convergence to the minimum along with rapid growth in feature norm, followed by oscillation in class probabilities. Over time, the network learns to use other signal and downweights the sky feature. We reproduce this figure for other inputs and for a VGG-11-BN [45] in the appendix, with similar findings.

3.3 Theoretical Analysis of Opposing Signals in a Simple Model

To demonstrate this effect more concretely, we study misspecified linear regression on inputs $x \in \mathbb{R}^d$ with a two-layer linear network. Though this model is quite simplified, it enables preliminary insight into the most important factors for these dynamics to occur. However, the concept of a "partially useful" signal seems to require a somewhat more complex model to properly capture (e.g., multinomial logistic regression), so we view this analysis only as an early investigation.

We present our results in Appendix H. We study the trajectory of gradient flow, proving first an initial decrease in sharpness due to the model downweighting the noise, followed by a continuous, steady growth in sharpness as the signal is amplified. Though we do not prove it, under gradient descent with large enough step size the sharpness will cross the stability threshold and the network will begin to *reintroduce* the noise variable. We also plot simulations demonstrating the proven behavior, as well as a comparison to almost identical behavior on a small MLP trained on a 5k subset of CIFAR-10. We make several further experimental observations which seem relevant for interpreting the effect of these outliers and what that may imply more generally about the influence of specific training points on NN optimization. We briefly list them here, with a more in-depth discussion in Appendix D: (i) sharpness often occurs overwhelmingly in the first few layers; (ii) batchnorm may smooth training, even if not the loss itself; (iii) for both GD and SGD, approximately half of training points go up in loss on each step; and (iv) different losses and label smoothing have predictable effects on sharpening.

4 The Interplay of Opposing Signals and Stochasticity

Full-batch GD is not used in practice when training NNs. It is therefore pertinent to ask what these findings imply about stochastic optimization. We begin by verifying that this pattern persists during SGD. Fig. 4 displays the losses for four opposing group pairs of a VGG-11-BN trained on CIFAR-10 with SGD batch size 128. We observe that the paired groups do exhibit clear opposing oscillatory patterns, but they do not alternate with every step, nor do they always move in opposite directions. We reproduce this plot with a VGG-11 without BN in Fig. 29 in the Appendix. Having verified that this oscillation on opposing signals still occurs in the stochastic setting, we conjecture that current best

practices for neural network optimization owe at least some of their success to gracefully handling such imbalances. We investigate this possibility concretely for the Adam optimizer [21].

4.1 How Adam Handles Gradients with Opposing Signals

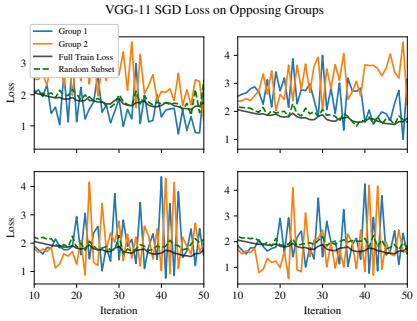


Figure 4: We plot the losses of paired outlier groups, along with the full train loss for comparison. Modulo batch randomness, the outliers' loss follow the same oscillatory pattern with large magnitude. See appendix for the same without BN.

To better understand their differences, Fig. 5 visualizes the parameter iterates of Adam and SGD with momentum on a ReLU MLP trained on a 5k subset of CIFAR-10, alongside those of GD and SGD. The top figure is the projection of these parameters onto the top eigenvector of the loss Hessian of the network trained with GD, evaluated at the first step where the sharpness crosses $2/\eta$. We observe that SGD tracks a similar path to GD, though adding momentum mitigates the oscillation somewhat. In contrast, the network optimized with Adam markedly departs from this pattern, smoothly oscillating along one side without crossing the middle. We identify several components of Adam which potentially contribute to this effect. For space constraints we list them here and expand upon each item in detail in Appendix F: (i) normalization means Adam takes very small steps near the minimum; (ii) the "trust region" means there is not too much dependence on imbalanced opposing signals, and since it is not a descent method it isn't dominated by the direction "down the valley"; and (iii) Adam's gradient implicitly incorporates *dampening* in addition to the usual momentum term, which we argue is actually an important distinction.

To test whether our findings translate to practical gains, we design a variant of SGD which incorporates these insights. First, we use dampening $\tau = 0.9$ in addition to momentum. Second, we choose a global threshold: if the gradient magnitude for a given parameter is above this threshold, we take a fixed step size. Otherwise, we take a gradient step as normal. In Appendix G we show that this approach matches Adam when training a ResNet-110 on CIFAR-10 with learning rates across several orders of magnitude. We also compare the two methods for the initial phase of training of GPT-2 on the OpenWebText dataset—not only do they perform the same, their loss similarity suggests that their exact trajectory may be very similar.

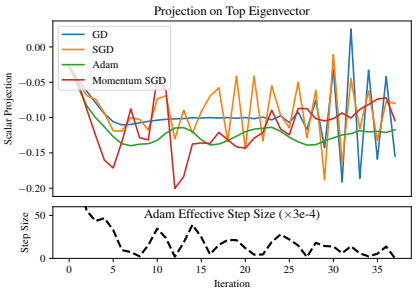


Figure 5: **Projected iterates of an MLP on CIFAR-10. Top:** SGD closely tracks GD, bouncing across the valley; momentum somewhat mitigates the sharp jumps. Adam smoothly oscillates along one side. **Bottom:** Adam's effective step size drops sharply when moving too close or far from the valley floor.

5 Conclusion

The existence of outliers with such a significant yet non-obvious influence on neural network training raises as many questions as it answers. This work presents an initial investigation into their effect on various aspects of optimization and the decisions of the model which results from it, but there is still much more to understand. Though it is clear they have a large influence on training, less obvious is whether reducing their influence is *necessary* for improved optimization, and if doing so will result in models which depend more broadly on the training set or just depend heavily on a different set of points. At the same time, there is evidence that the behavior these outliers induce may serve as an important method of exploration and/or regularization. If so, another key question is whether these two effects can be decoupled—or if the incredible generalization ability of neural networks somehow relies on the fact that their training is so disproportionately affected by a small subset of the data.

References

[1] Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In *International Conference on Machine Learning*, pages 948–1024. PMLR, 2022.

[2] Boaz Barak, Benjamin L. Edelman, Surbhi Goel, Sham M. Kakade, eran malach, and Cyril Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=8XWP2ewX-im.

[3] Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. In *International Conference on Machine Learning*, pages 1817–1853. PMLR, 2022.

[4] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, et al. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*, 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. URL https://openreview.net/forum?id=jh-rTtvkGeM.

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

[7] Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022. URL https://openreview.net/forum?id=enoU_Kp7Dz.

[8] Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[9] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

[10] Stanislav Fort and Surya Ganguli. Emergent properties of the local geometry of neural loss landscapes. *arXiv preprint arXiv:1910.05929*, 2019.

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[12] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19b.html.

[13] Kelvin Guu, Albert Webson, Ellie Pavlick, Lucas Dixon, Ian Tenney, and Tolga Bolukbasi. Simfluence: Modeling the influence of individual training examples by simulating training runs. *arXiv preprint arXiv:2303.08114*, 2023.

[14] Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021.

[15] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9525–9587. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/ilyas22a.html.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[17] Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[18] Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amost Storkey. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkgEaj05t7.

[19] Stanisław Jastrzębski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho*, and Krzysztof Geras*. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

[20] Stanisław Jastrzębski, Devansh Arpit, Oliver Astrand, Giancarlo B Kerg, Huan Wang, Caiming Xiong, Richard Socher, Kyunghyun Cho, and Krzysztof J Geras. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2021. URL https://proceedings.mlr.press/v139/jastrzebski21a.html.

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

[22] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[23] Dmitry Kopitkov and Vadim Indelman. Neural spectrum alignment: Empirical study. In *Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part II 29*, pages 168–179. Springer, 2020.

[24] Itai Kreisler, Mor Shpigel Nacson, Daniel Soudry, and Yair Carmon. Gradient descent monotonically decreases the sharpness of gradient flow solutions in scalar networks and beyond. In *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kreisler23a.html.

[25] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

[26] Xinyan Li, Qilong Gu, Yingxue Zhou, Tiancong Chen, and Arindam Banerjee. Hessian based analysis of sgd for deep nets: Dynamics and generalization. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 190–198. SIAM, 2020.

[27] Chao Ma and Lexing Ying. On linear stability of sgd and input-smoothness of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 16805–16817. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8c26d2fad09dc76f3ff36b6ea752b0e1-Paper.pdf.

[28] Chao Ma, Daniel Kunin, Lei Wu, and Lexing Ying. Beyond the quadratic approximation: the multiscale structure of neural network loss landscapes. *arXiv preprint arXiv:2204.11326*, 2022.

[29] Lachlan Ewen MacDonald, Jack Valmadre, and Simon Lucey. On progressive sharpening, flat minima and generalisation. *arXiv preprint arXiv:2305.14683*, 2023.

[30] William Merrill, Nikolaos Tsilivis, and Aman Shukla. A tale of two circuits: Grokking as competition of sparse and dense subnetworks. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023. URL https://openreview.net/forum?id=8GZxtu46Kx.

[31] Rotem Mulayoff and Tomer Michaeli. Unique properties of flat minima in deep networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/mulayoff20a.html.

[32] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. *arXiv preprint arXiv:1905.11604*, 2019.

[33] 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. URL https://openreview.net/forum?id=9XFSbDPmdW.

[34] Pascal Notsawo Jr, Hattie Zhou, Mohammad Pezeshki, Irina Rish, Guillaume Dumas, et al. Predicting grokking long before it happens: A look into the loss landscape of models which grok. *arXiv preprint arXiv:2306.13253*, 2023.

[35] Yan Pan and Yuanzhi Li. Toward understanding why adam converges faster than sgd for transformers. *arXiv preprint arXiv:2306.00204*, 2023.

[36] Vardan Papyan. The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

[37] Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/papyan19a.html.

[38] Vardan Papyan. Traces of class/cross-class structure pervade deep learning spectra. *The Journal of Machine Learning Research*, 21(1):10197–10260, 2020.

[39] 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.

[40] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

[41] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[42] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[43] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[44] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[46] Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*, 2020.

[47] 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.

[48] Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rye4g3AqFm.

[49] Zixuan Wang, Zhouzi Li, and Jian Li. Analyzing sharpness along GD trajectory: Progressive sharpening and edge of stability. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=thgItcQrJ4y.

[50] Lei Wu, Chao Ma, and Weinan E. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/6651526b6fb8f29a00507de6a49ce30f-Paper.pdf.

[51] Lei Wu, Mingze Wang, and Weijie J Su. The alignment property of SGD noise and how it helps select flat minima: A stability analysis. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=rUc8peDIM45.

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

[53] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

[54] Xingyu Zhu, Zixuan Wang, Xiang Wang, Mo Zhou, and Rong Ge. Understanding edge-of-stability training dynamics with a minimalist example. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=p7EagBsMAEO.

[55] Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 7654–7663. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/zhu19e.html.

A Related Work and Discussion

Characterizing the NN loss landscape. Earlier studies of the loss landscape extensively explored the spectrum of the Hessian of the loss, with a common observation being a hierarchical structure with a small group of very large outlier eigenvalues [40, 41, 36, 37, 10, 12, 26, 38, 23]. Later efforts focused on concretely linking these observations to corresponding behavior, often with an emphasis on SGD's bias towards particular solutions [50, 17, 19] and what this may imply about its resulting generalization [18, 55, 51]. Our method for identifying these paired groups, along with Fig. 27, indicates that this outlier spectrum is precisely the directions with opposing signals in the gradient, and that this pattern may be key to better understanding the generalization ability of NNs trained with SGD.

Progressive sharpening and the edge of stability. Shifting away from the overall structure, more recent focus has been specifically on top eigenvalue(s), where it was empirically observed that their magnitude (the loss "sharpness") grows when training with SGD [18, 19] and GD [23, 5] (so-called "progressive sharpening"). This leads to rapid oscillation in weight space [52, 18, 5, 6]. Cohen et al. [5] also found that for GD this coincides with a consistent yet non-monotonic decrease in training loss over long timescales, which they named the "edge of stability"; moreover, they noted that this behavior runs contrary to our traditional understanding of NN convergence. Many works have since investigated the possible origins of this phenomenon [54, 24]; several are deeply related to our findings. Ma et al. [28] connect this behavior to the existence of multiple "scales" of losses; the outliers we identify corroborate this point. Damian et al. [7] prove that GD implicitly regularizes the sharpness—we identify a conceptually distinct source of such regularization, as described in Section 3. Arora et al. [1] show under some conditions that the GD trajectory follows a minimum-loss manifold towards lower curvature regions. This is consistent with our findings, and we believe this manifold to be precisely the path which evenly balances the opposing gradients. Wang et al. [49] provide another thorough analysis of NN training dynamics at the edge of stability; their demonstrated phases closely align with our own. They further observe that this sharpening coincides with a growth in the norm of the last layer, which was also noted by MacDonald et al. [29]. Our proposed explanation for the effect of opposing signals offers some insight into this relationship, but further investigation is needed.

Generalization, grokking, slingshotting, and subnetworks. We see ties between opposing signals and many other phenomena in NN optimization, but a few specific concepts stand out to us as most clearly related. At least in images, the features which provide opposing signals match the traditional picture of "spurious correlations" surprisingly closely—we conjecture that the factors affecting whether a network maintains balance or diverges along a direction also determine whether it continues to use a "spurious" feature or is forced to find an alternative way to minimize loss. Indeed, the exact phenomenon of a network "slingshotting" to a new region has been directly observed, along with a corresponding change in generalization [50, 25, 20, 47]. "Grokking" [39], whereby a network learns to generalize long after memorizing the training set, is another closely related area of study. Several works have shown that grokking is a "hidden" phenomenon, with gradual amplification of generalizing subnetworks [2, 33, 30]; it has even been noted to co-occur with weight oscillation [34]. We observe that the opposing signal gradients dominate the overall gradient for most of training—and our experiments on SGD in Section 4 and Appendix G give some further evidence that NN optimization occurs on two different scales, obscured by the network's behavior on outliers. We note that this also relates to the Lottery Ticket Hypothesis [11]: we think it is not unlikely that a pruned winning ticket is one which has mostly eliminated the influence of opposing signals, allowing it to more easily traverse the loss landscape. In fact, said ticket may already be following this trajectory during standard optimization—but the behavior (and loss) of the remainder of the network on these opposing signals would obscure its progress.

Simplicity bias. Relatedly, Nakkiran et al. [32] observe that NNs learn functions of greater complexity throughout training. Our experiments—particularly the slow decay in the norm of the feature embedding of opposing signals—lead us to believe it would be more correct to say that they *unlearn* simple functions, which enables smaller, more complex subnetworks with better performance to take over. At first this seems at odds with the notion of "simplicity bias" [48, 44], defined broadly as a tendency of networks to rely on simple functions of their inputs. However, we are led to believe that the network *will* use the simplest features (e.g., largest norm or something similar) that it can—so long as such features allow it to achieve reasonably small training loss. Otherwise it will eventually

begin to diverge out of the valley again. We believe the existence of opposing signals in data presents new questions on the origin of this phenomenon and what elements may be important to investigate further. For example, perhaps this could enable alternative definitions of "simplicity" which more closely match the specific way the inputs are processed by the network.

Sharpness-Aware Minimization One final possible connection we think merits further inquiry is Sharpness-Aware Minimization (SAM) [9], which is known to improve generalization of neural networks for reasons still not fully understood. In particular, the better-performing variant is 1-SAM, which takes initial gradient steps on each training point in the batch individually. In light of our result, it seems apparent that several such updates will point directly along directions of steepest descent/ascent orthogonal to the valley floor (and, if not normalized, the updates may be *very* large). Thus it may be that 1-SAM is in some sense "simulating" divergence out of this valley, enabling exploration for the rest of the network in a manner that would not normally be possible until the sharpness grows large enough. The interaction between SAM and outliers with opposing signals seems to us a promising direction for meaningful insights into how to improve generalization even further.

B Examples of Opposing Signals in Text

Punctuation Ordering

```
believe that it was the right thing to do.[”] /
tunnel, because it shows when we test them.[”]
/ but I think it’s a pretty comfortable bike.[”]
/ I used to catch me a few and make pets out of
them.[”]
```

```
driven by ideological and political motives[”].
/ personal bank account was a "genuine
donation[”]. / "most consequential decision
I’ve ever been involved with[”].
```

New Line After Colon

```
According to the CBO update:[\n] / 5 reasons as
to why self diagnosis is valid:[\n] / successive
Lambda invocations. It looks more or less like
this:[\n] / data, there are three things to
do:[\n]
```

```
religion return in various ways to one core
issue:[ ] / other acts of love, both divine and
human:[ ] / integrate fighters from the Kurds’
two main political parties:[ ] / of precisely
what makes it so wonderful:[ ]
```

Figure 6: **Examples of opposing signals in text.** Found by training GPT-2 on a subset of OpenWebText. Sequences are separated by forward slashes, the token in brackets is the target and all prior tokens are (the end of the) context. **Left:** As both standards are used, it is not always clear whether punctuation will come before or after the end of a quotation (we include the period after the quote for clarity—the model does not condition on it). **Right:** Sometimes a colon occurs mid-sentence, other times it announces the start of a new line. The model must *unlearn* “: ↦ [\n]” versus “: ↦ [ ]” and instead use other contextual information (possibly by diverging out of the valley).

C Reproducing Fig. 3 in Other Settings

Though colors are straightforward, for some opposing signals such as grass texture it is not clear how to produce a synthetic image which properly captures what precisely the model is latching on to. Instead, we identify a real image which has as much grass and as little else as possible, with the understanding that the additional signal in the image could affect the results. We depict the grass image alongside the plots it produced.

C.1 ResNet-18 Trained with GD on Other Inputs

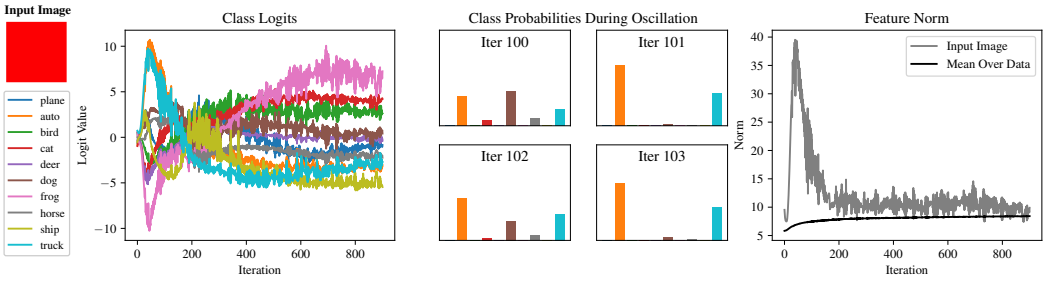


Figure 7: ResNet-18 on a red color block.

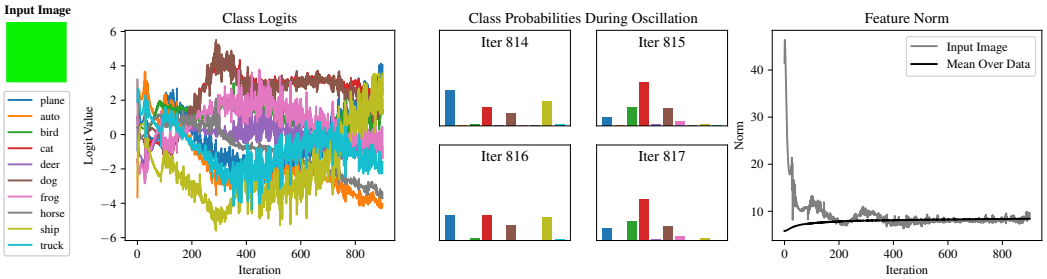


Figure 8: ResNet-18 on a green color block. As this color seems unnatural, we've included two examples of relevant images in the dataset.

Figure 9: Examples of images with the above green color.

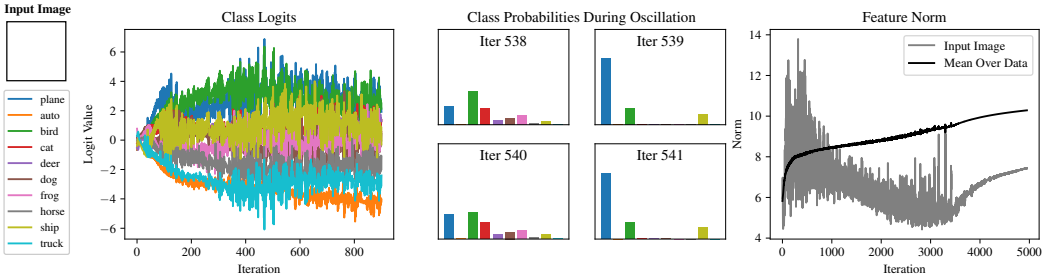


Figure 10: ResNet-18 on a white color block.

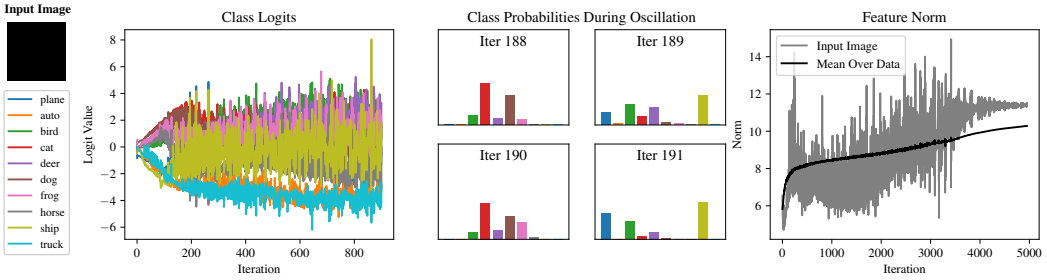


Figure 11: ResNet-18 on a black color block.

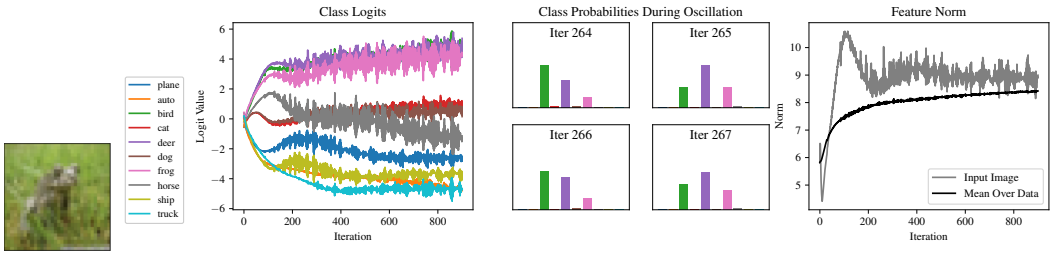


Figure 12: ResNet-18 on an image with mostly grass texture.

C.2 VGG-11-BN Trained with GD

For VGG-11, we found that the feature norm of the embedded images did not decay nearly as much over the course of training. We expect this has to do with the lack of a residual component. However, for the most part these features do still follow the pattern of a rapid increase, followed by a marked decline.

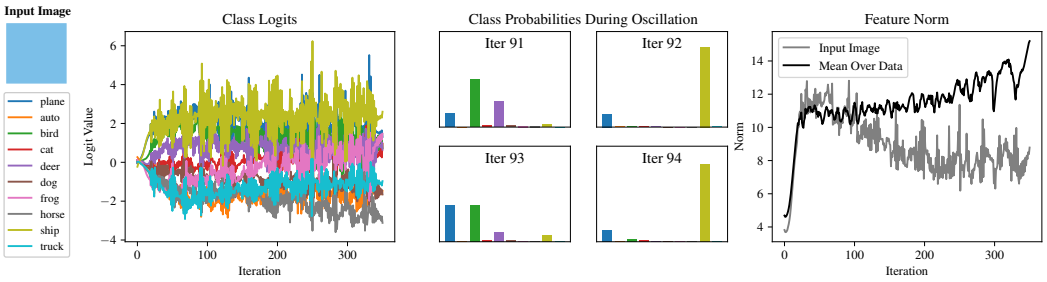


Figure 13: VGG-11-BN on a sky color block.

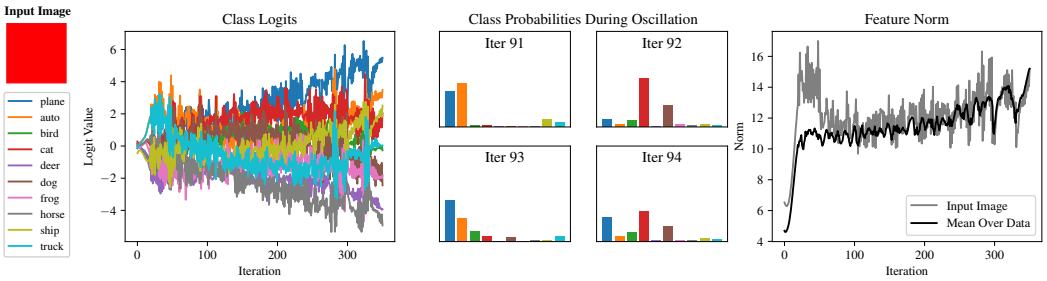


Figure 14: VGG-11-BN on a red color block.

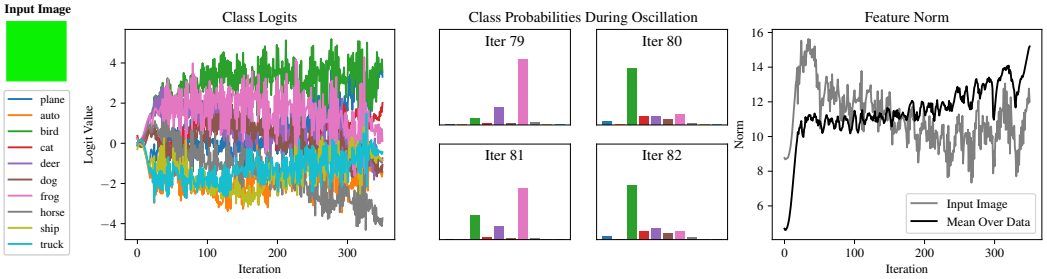


Figure 15: VGG-11-BN on a green color block. See above for two examples of relevant images in the dataset.

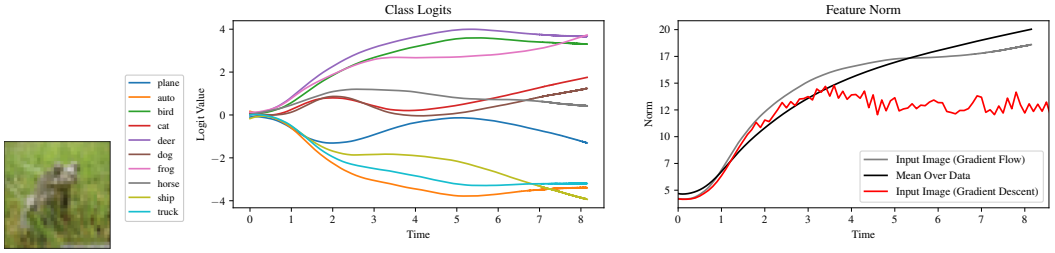


Figure 16: VGG-11-BN on an image with mostly grass texture.

C.3 VGG-11-BN with Small Learning Rate to Approximate Gradient Flow

Here we see that oscillation is a valuable regularizer, preventing the network from continuously upweighting opposing signals. As described in the main body, stepping too far in one direction causes an imbalanced gradient between the two opposing signals. Since the group which now has a larger loss is also the one which suffers from the use of the feature, the network is encouraged to downweight its influence. If we use a very small learning rate to approximate gradient flow, this regularization does not occur and the feature norms grow continuously. This leads to over-reliance on these features, suggesting that failing to downweight opposing signals is a likely cause of the poor generalization of networks trained with gradient flow.

The following plots depict a VGG-11-BN trained with learning rate .0005 to closely approximate gradient flow. We compare this to the feature norms of the same network trained with gradient descent with learning rate 0.1, which closely matches gradient flow until it becomes unstable..

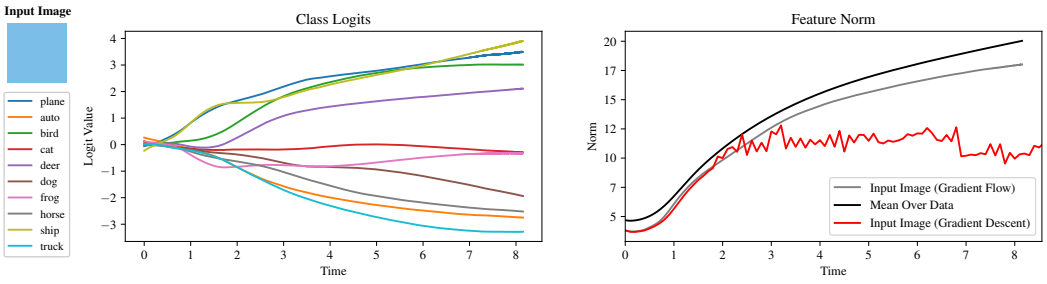


Figure 17: VGG-11-BN on a sky color block with learning rate 0.005 (approximating gradient flow) compared to 0.1.

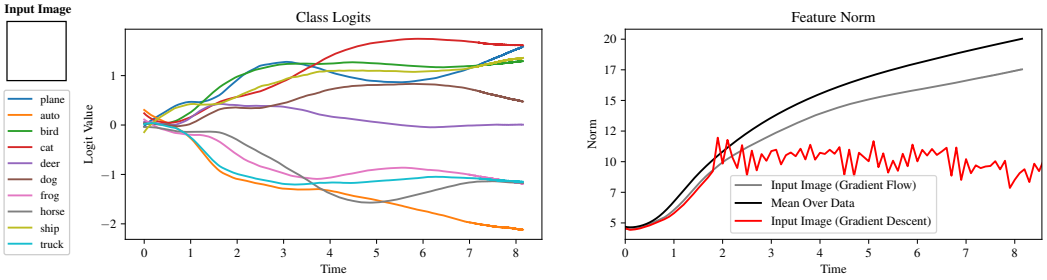


Figure 18: VGG-11-BN on a white color block with learning rate 0.005 (approximating gradient flow) compared to 0.1.

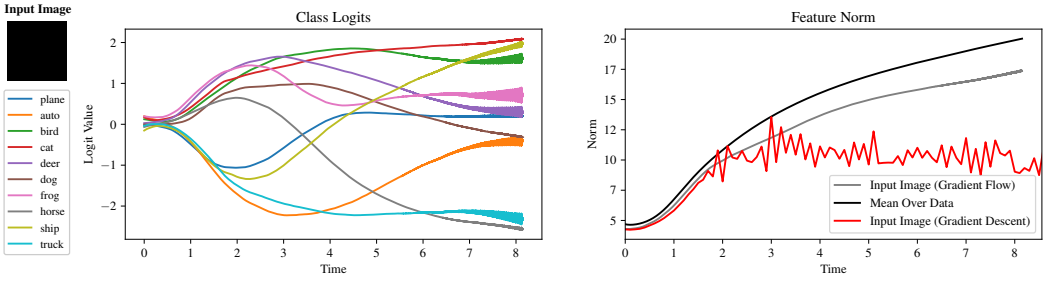


Figure 19: VGG-11-BN on a black color block with learning rate 0.005 (approximating gradient flow) compared to 0.1.

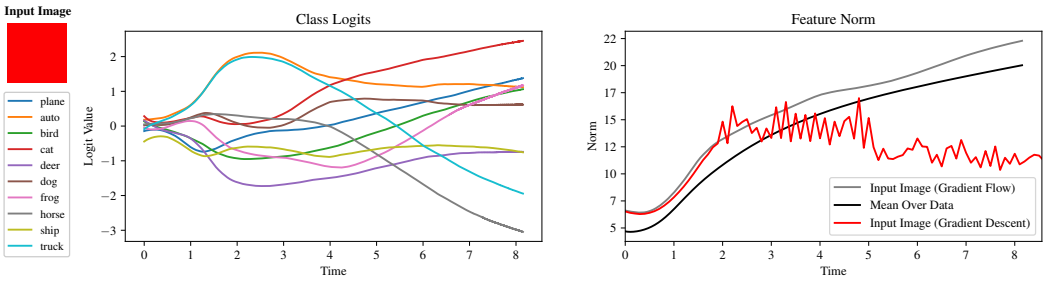


Figure 20: VGG-11-BN on a red color block with learning rate 0.005 (approximating gradient flow) compared to 0.1.

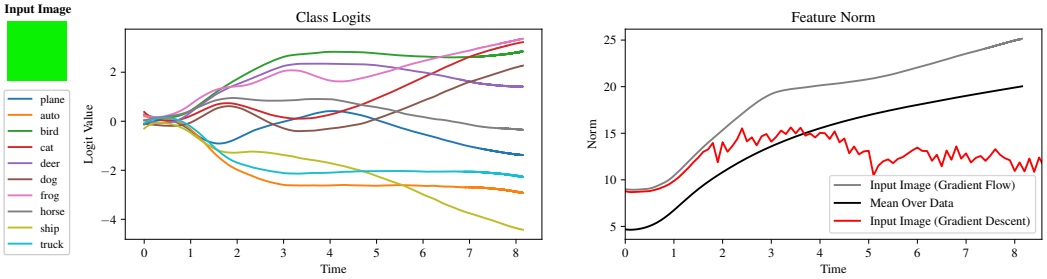


Figure 21: VGG-11-BN on a green color block with learning rate 0.0005 (approximating gradient flow) compared to 0.1.

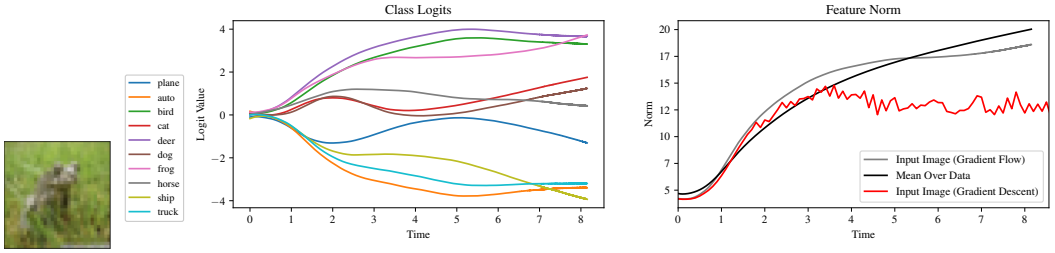


Figure 22: VGG-11-BN on an image with mostly grass texture with learning rate 0.0005 (approximating gradient flow) compared to 0.1.

C.4 ResNet-18 Trained with Full-Batch Adam

Finally, we plot the same figures for a ResNet-18 trained with full-batch Adam. We see that Adam consistently and quickly reduces the norm of these features, especially for more complex features such as texture, and that it also quickly reaches a point where oscillation ends. Note when comparing to plots above that the maximum iteration on the x-axis differs.

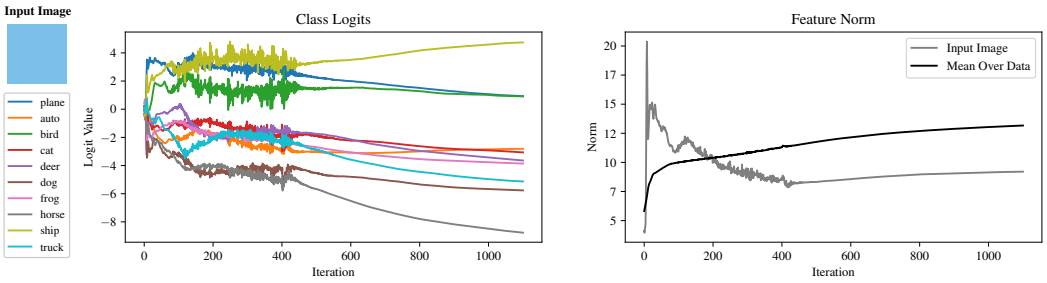


Figure 23: ResNet-18 on a sky color block trained with Adam.

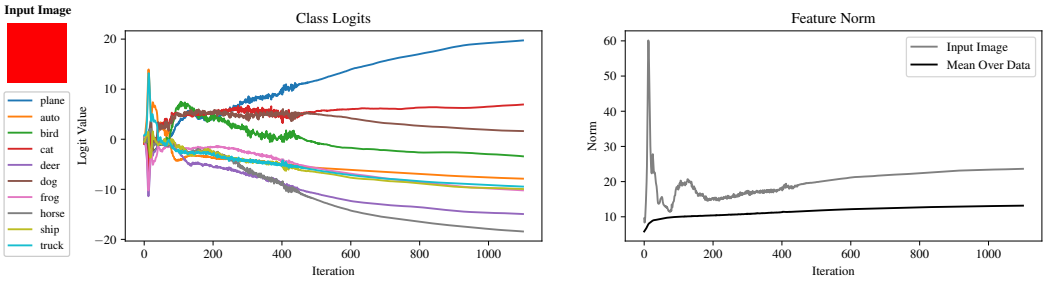


Figure 24: ResNet-18 on a red color block trained with Adam.

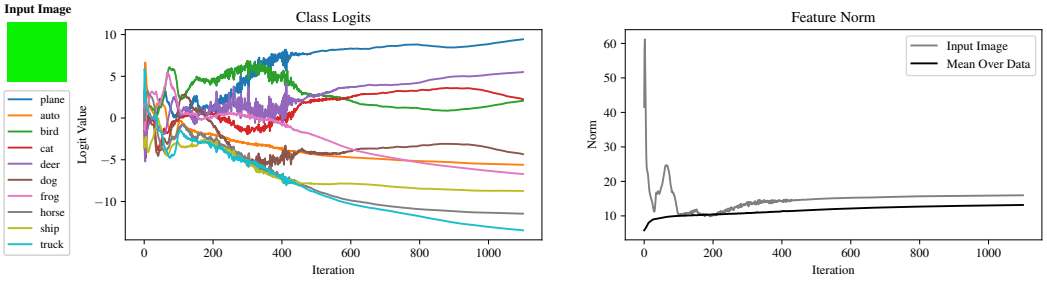


Figure 25: ResNet-18 on a green color block trained with Adam.

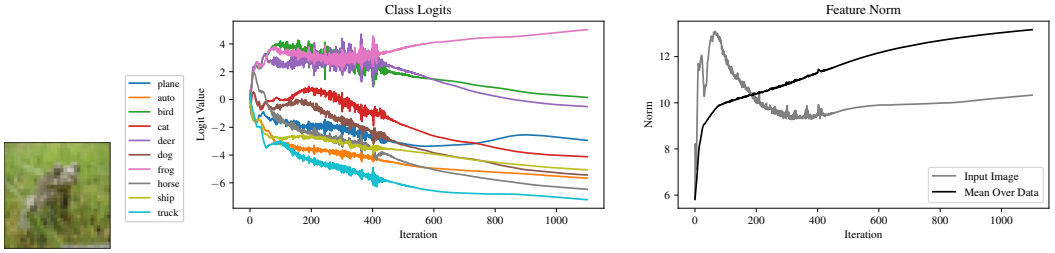


Figure 26: ResNet-18 on an image with mostly grass texture trained with Adam.

D Discussion of Additional Experimental Findings

Implications for the effect of loss smoothing or other losses, as well as various activations. We found that the sharpness in ResNets occurred overwhelmingly in the first convolutional layer, after the first few training steps where it was in the last layer. For VGG, it occurred mostly in the first few layers. In transformers, curvature typically was most concentrated in the initial embedding layer and the first few MLP projection layers. Generally, it seems that the components of the network which *interact most directly with the input* have the most significant sharpness—particularly if they also perform dimensionality reduction. This is consistent with our understanding of what causes said sharpness.

Batchnorm may smooth training, even if not the loss itself. We also found that adding an additional batchnorm (BN) layer [16] before the last ResNet layer reduced its initial sharpness in that location. Cohen et al. [5] noted that BN does not prevent networks from reaching the edge of stability and concluded, contrary to [42], that BN does not smooth the loss landscape; we **conjecture** that the effect of BN depends on the use of GD vs. SGD. Specifically, our findings hint at a possible benefit of BN which applies *only* to minibatches: reducing the influence of imbalanced opposing signals. This suggests that in fact BN may smooth the *optimization trajectory* of neural networks, rather than the loss itself (this is consistent with the distinction made by Cohen et al. [5] between regularity and smoothness). In Section 4 we demonstrate that Adam also smooths the optimization trajectory and that minor changes to emulate this effect can aid stochastic optimization.

For both GD and SGD, approximately half of training points go up in loss on each step. Though only the outliers are wildly oscillating, many more images contain some small component of the features they exemplify. Fig. 28 tracks the fraction of points which increase in loss on each step—to some extent, a small degree of oscillation appears to be happening to the entire dataset.

Different losses and label smoothing have predictable effects on sharpening. [29] note that label smoothing the cross-entropy loss reduces sharpening. This is reasonably explained by the fact that smoothing reduces the loss suffered by extreme overconfidence, and therefore the loss for a given opposing signal will not be as sharp. This also hints at why logistic loss may be more suitable for NN optimization, because it only has substantial curvature around $x = 0$ so, unlike square or exponential loss, large steps will not massively increase sharpness. We expect a similar property may contribute to the relative performance of various activations (e.g., ReLU).

E Additional Figures

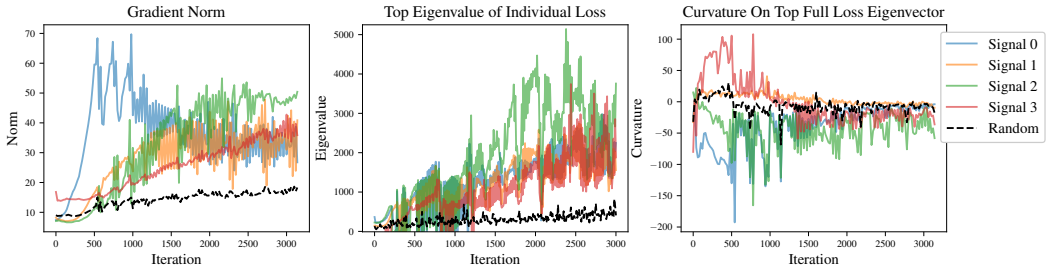


Figure 27: **Tracking other metrics which characterize outliers with opposing signals.** For computational considerations, we select outliers according to maximal per-step change in loss. However, this quantity relates to other useful metrics, such as gradient norm and curvature. We see that the samples we uncover are also significant outliers according to these metrics.

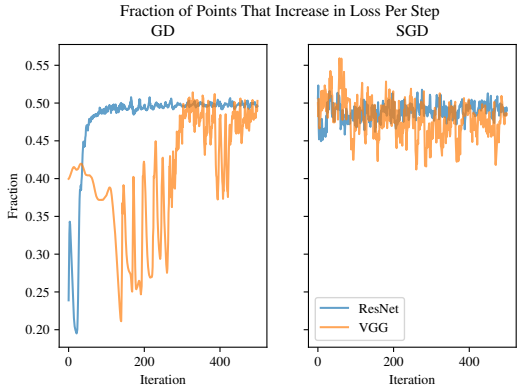


Figure 28: The fraction of overall training points which increase in loss on any given step. For both SGD and GD, it hovers around 0.5 (VGG without batchnorm takes a long time to reach the edge of stability). Though only the outliers are wildly swinging in loss, many more images contain *some* small component of the features they exemplify, and so these points also oscillate in loss at a smaller scale.

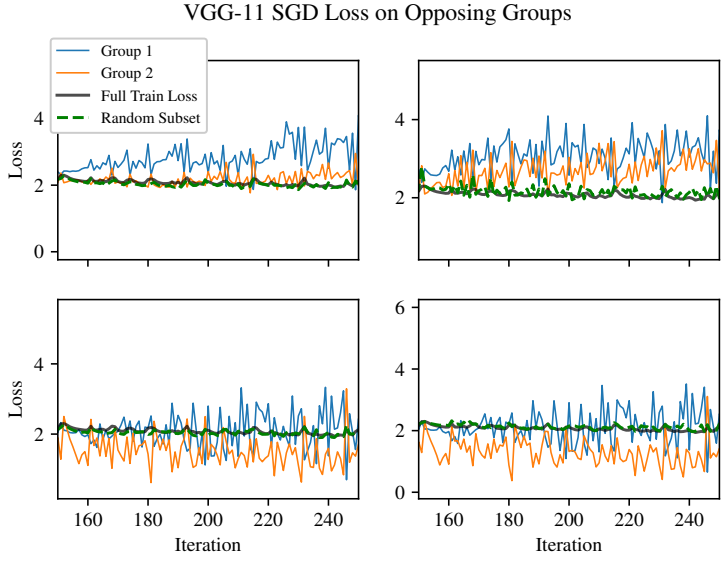
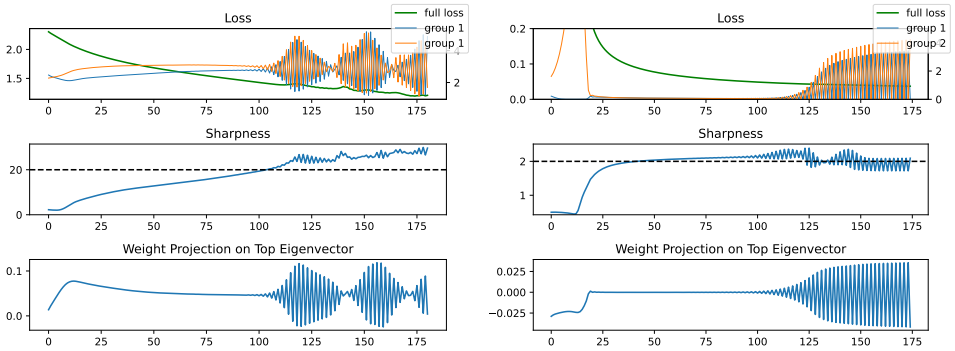


Figure 29: We reproduce Fig. 4 without batch normalization.



(a) A 3-layer ReLU MLP trained on a 5k-subset of CIFAR-10.

(b) Our model: a 2-layer linear network trained on mostly Gaussian data with opposing signals.

Figure 30: We compare a small ReLU MLP on a subset of CIFAR-10 to our simple model of linear regression with a two-layer network.

F Discussion on the Advantages of Adam

First, normalization means Adam takes very small steps near the minimum. The lower figure shows that the effective step size of Adam along the top eigenvector rapidly drops to zero as the iterates approach the barrier (in the opposite direction, the gradient negates the momentum for the same effect). However, we conjecture that this may not be essential to Adam's performance; we even expect this may be somewhat harmful by limiting exploration. Instead, [53] identified the "trust region" as an important contributor to Adam's success in attention models, blaming heavy-tailed noise in the stochastic gradients. We refine this observation by noting that often, the difficulty is not heavy-tailed *noise*—it is a strong, directed (but imbalanced) signal. We identify one additional possible advantage conferred by the trust region: the largest steps emulate sign SGD, which is notably *not* a descent method. Fig. 5 shows that Adam's steps are mostly parallel to the valley floor, which would not be possible with steepest descent. In line with this, Benzing [3] observe that true second order methods underperform on NNs. Pan and Li [35] also identify outlier directions of sharpness as a reason for SGD's underperformance, and they show that gradient clipping can help; such clipping would naturally help to avoid large steps towards (or away from) the minimum. Lastly, the third important factor: down-weighting the most recent gradient. Traditional SGD with momentum $\beta < 1$ takes a step which weights the current gradient by $\frac{1}{1+\beta} > \frac{1}{2}$. Though this makes intuitive sense, our results imply that heavily weighting the most recent gradient can be problematic. Instead, we expect an important addition is *dampening*, which multiplies the gradient at each step by some $(1-\tau) < 1$. We observe that Adam's (unnormalized) gradient is equivalent to SGD with momentum and dampening both equal to β_1, plus a debiasing step. Recently proposed alternatives also include dampening in their momentum update but do not explicitly identify the distinction [53, 35, 4].

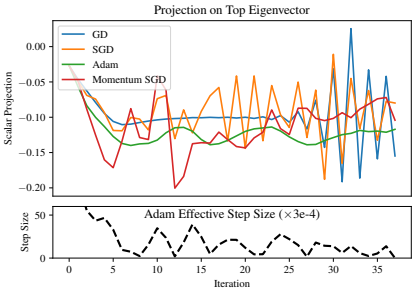


Figure 31: **Projected iterates of an MLP on CIFAR-10. Top:** SGD closely tracks GD, bouncing across the valley; momentum somewhat mitigates the sharp jumps. Adam smoothly oscillates along one side. **Bottom:** Adam's effective step size drops sharply when moving too close or far from the valley floor.

G Comparing our Variant of SGD to Adam

As described in the main text, we find that simply including dampening and taking a fixed step size on gradients above a certain threshold results in performance matching that of Adam for the experiments we tried. We found that setting this threshold equal to the $q = .1$ quantile of the very first gradient worked quite well—this was about 1e-4 for the ResNet-110 and 1e-6 for the transformer.

For the ResNet we also ablate the use of dampening: we find that masking appears to be much more important for early in training, while dampening is helpful for maintaining performance later. It is not immediately what may be the cause of this, nor if it will necessarily transfer to attention models.

Simply to have something to label it with, we name the method SplitSGD, because it performs SGD and SignSGD on different partitions of the data. We emphasize that we do not intend to introduce this as a new optimization algorithm or assert any kind of superior performance—our intent is only to demonstrate the insight gained from knowledge of opposing signals' influence on NN optimization.

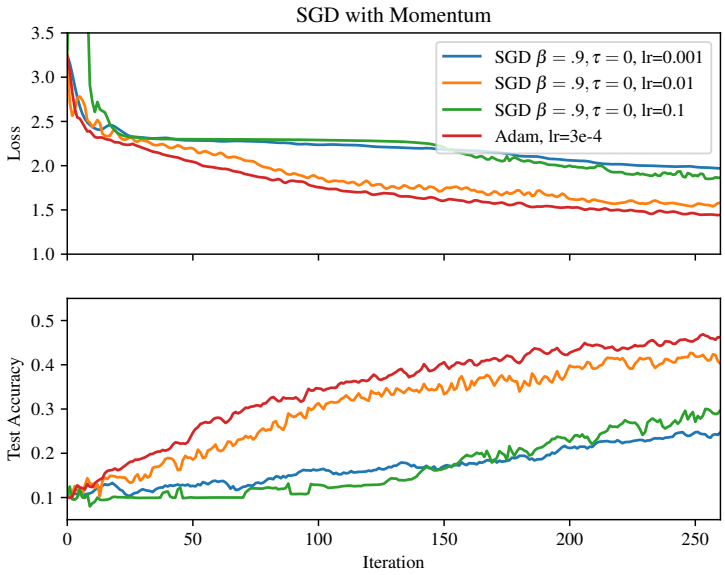


Figure 32: Adam versus standard SGD with Momentum

For the transformer, we use the public nanoGPT repository which trains GPT-2 on the OpenWebText dataset. As a full training run would be too expensive, we compare only for the early stage of optimization. Not only do the two methods track each other closely, it appears that they experience *exactly* the same oscillations in their loss. Though we do not track the parameters themselves, we believe it would be meaningful to investigate if these two methods follow very similar optimization trajectories as well.

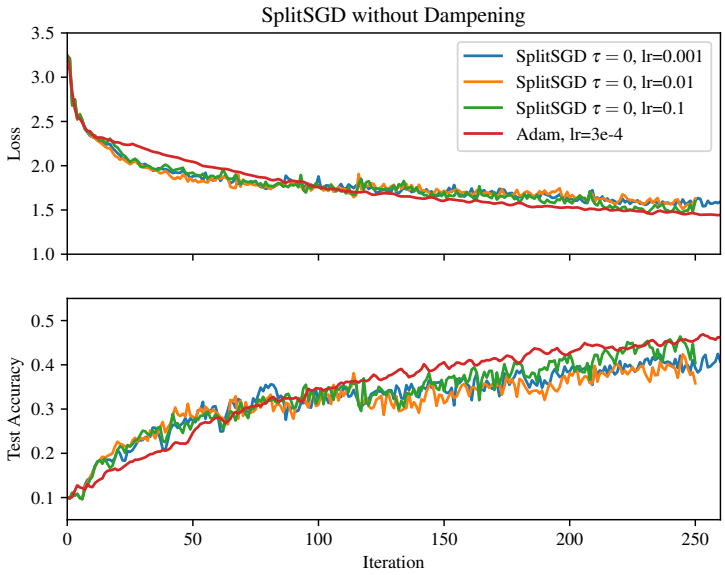
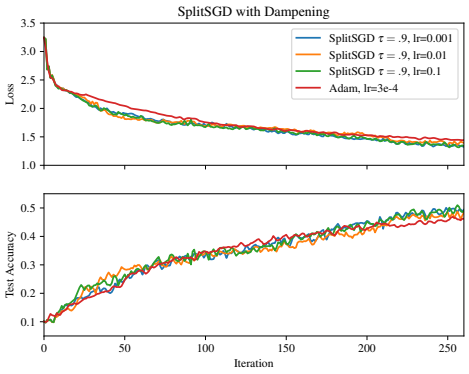
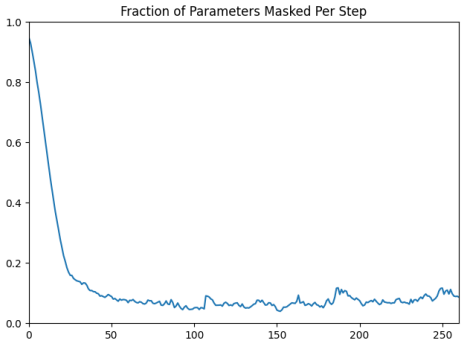


Figure 33: Adam vs. SplitSGD, but with no dampening.



(a) Adam vs. SplitSGD with dampening



(b) The fraction of parameters for which a fixed-size signed step was taken for each gradient step.

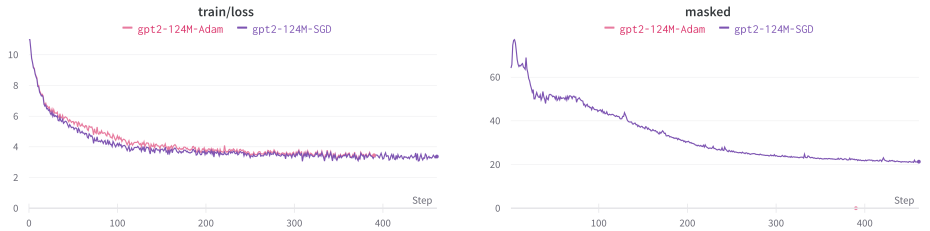


Figure 35: Adam versus SplitSGD on the initial stage of training GPT-2 on the OpenWebText dataset, and the fraction of parameters with a fixed-size signed step. All hyperparameters are the defaults from the nanoGPT repository. Observe that not only is their performance similar, they appear to have *exactly* the same loss oscillations.

H Presentation of Theoretical Results

We model the observed features as a distribution over $x \in \mathbb{R}^{d_1}$, assuming only that its covariance Σ exists—for clarity we treat $\Sigma = I$ in the main text. We further model an additional vector $x_o \in \mathbb{R}^{d_2}$ representing the opposing signal, with $d_2 > d_1$. We will suppose that on some small fraction of

outliers $p \ll 1$, $x_o \sim \text{Unif}\left(\left\{\pm\sqrt{\frac{\alpha}{pd_2}}\mathbf{1}\right\}\right)$ ($\mathbf{1}$ is the all-ones vector) for some α which governs the feature magnitude, and we let it be $\mathbf{0}$ on the remainder of the dataset. We model the target as the linear function $y = \beta^\top x + \frac{1}{\sqrt{d_2}}\mathbf{1}^\top |x_o|$; this captures the idea that the signal x_o correlates strongly with the target, but in opposing directions of equal strength. Finally, we parameterize the network with vectors $b \in \mathbb{R}^{d_1}, b_o \in \mathbb{R}^{d_2}$ and scalar c in one single vector θ, as $f_\theta(x) = c \cdot (b^\top x + b_o^\top x_o)$. Note the specific distribution of x_o is unimportant—furthermore, in our simulations we observed the exact same pattern with cross-entropy loss. From these experiments and our analysis, it seems that depth and a small signal-to-noise ratio are the only elements needed for this behavior to arise.

A standard initialization would be to sample $[b, b_o]^\top \sim \mathcal{N}(0, \frac{1}{d_1+d_2}I)$, which would then imply highly concentrated distributions for $\|b\|_2^2, \|b_o\|_2^2, b^\top\beta$. As tracking the precise concentration terms would not meaningfully contribute to the analysis, we simplify by directly assuming that at initialization $\|b\|_2^2 = \frac{d_1}{d_1+d_2}$, $\|b_o\|_2^2 = \frac{d_2}{d_1+d_2}$, and $b^\top\beta = \frac{\|\beta\|}{\sqrt{d_1+d_2}}$. Likewise, we let $c = 1$, ensuring that both layers have the same norm. We perform standard linear regression by minimizing the population loss $L(\theta) := \frac{1}{2}\mathbb{E}[(f_\theta(x) - y)^2]$. For our purposes, it will be sufficient to study the trajectory of *gradient flow*—results for gradient descent will then follow for step sizes η which are not too large, especially because the timescales of the phases of optimization we study shrink as α grows.

We see that the minimizer of this objective has $b_o = \mathbf{0}$ and $cb = \beta$. However, an analysis of the trajectory of gradient flow will elucidate how depth and large noise lead to sharpening—for larger α, the network will be forced initially to minimizes loss on the outliers. We note that in exploring the edge of stability, Cohen et al. [5] found that sometimes the model would have a brief *decrease* in sharpness, particularly for square loss. In fact, we show that this initial decay is expected in the presence of large magnitude, unhelpful signal.

Theorem H.1 (Initial *decrease* in sharpness)**.** *Let $k := \frac{d_2}{d_1}$, and assume $\|\beta\| > \max(\frac{d_1}{\sqrt{d_1+d_2}}, \frac{24}{5})$. At initialization, the sharpness $\|\nabla_\theta^2 L(\theta)\|_2$ lies in $[\alpha, \alpha(1 + 2\sqrt{d_2/(d_1+d_2)})]$. Further, if $\sqrt{\alpha} = \Omega(\|\beta\|k\ln k)$, then the sharpness will* decrease *as $O(e^{-\alpha t})$ from $t = 0$ until some time $t_1 \le \frac{\ln \|\beta\|/2}{2\|\beta\|}$.*

After this decrease, signal amplification can proceed—but the magnitude of the unhelpful feature means that the sharpness with respect to *how the network uses this feature* will grow, and so a small perturbation to this value will induce a large increase in loss.

Theorem H.2 (Progressive sharpening)**.** *If $\sqrt{\alpha} = \Omega\left(1 + \|\beta\|^2 k \ln k\right)$, then at starting at time t_1 the sharpness will increase linearly in $\|\beta\|$ until some time $t_2 \ge \frac{1}{2\|\beta\|_2^2}$, reaching at least $\frac{5}{8}\|\beta\|\alpha$. This lower bound on sharpness applies to each dimension of b_o.*

Oscillation will not occur during gradient flow—but for SGD with step size η larger than $\frac{16}{5\|\beta\|\alpha}$, this result means that b_o will start to increase in magnitude. If this growth continues for long enough, it will rapidly *reintroduce* the outlier feature, which will also cause loss alternation on the opposing outliers. We simulate this model and verify exactly this behavior: in Appendix E we visualize the dynamics alongside an MLP trained on CIFAR-10, which displays the same characteristic behavior. Due to the network being linear, the reintroduction of x_o can only lead to downweighting x_o and returning to the first phase. However, in a non-linear model the sudden reintroduction of a feature which was removed earlier in training seems quite useful for non-linear feature learning (e.g., around iteration 3000 of Fig. 3), where a particular signal may not be useful unless combined with others in a precise way. Though we are unable to reproduce this in our current model, we see the exploration of this behavior and its implications for optimization and generalization as an interesting direction for future study.

I Proofs of Theoretical Results

Before we begin the analysis, we must identify the quantities of interest during gradient flow and the system of equations that determines how they evolve.

We start by writing out the loss:

$$2L(\theta) = \mathbb{E}[(c(b^\top x + b_o^\top x_o) - (\beta^\top x + d_2^{-1/2}\mathbf{1}^\top |x_o|))^2] \tag{1}$$

$$= \mathbb{E}[((cb-\beta)^\top x)^2] + \mathbb{E}[((cb_o - d_2^{-1/2}\operatorname{sign}(x_o)\mathbf{1})^\top x_o)^2] \tag{2}$$

$$= \|cb-\beta\|^2 + \frac{p}{2}\left(\left(\sqrt{\frac{\alpha}{p}}(cb_o - 1)\right)^2 + \left(\sqrt{\frac{\alpha}{p}}(cb_o+1)\right)^2\right) \tag{3}$$

$$= \|cb-\beta\|^2 + \alpha(c^2\|b_o\|^2 + 1). \tag{4}$$

This provides the gradients

$$\nabla_b L = c(cb-\beta), \tag{5}$$

$$\nabla_{b_o} L = \alpha c^2 b_o, \tag{6}$$

$$\nabla_c L = b^\top(cb-\beta) + \alpha\|b_o\|^2 c. \tag{7}$$

We will also make use of the Hessian to identify its top eigenvalue; it is given by

$$\nabla_\theta^2 L(\theta) = \begin{bmatrix} c^2 I_{d_1} & \mathbf{0}_{d_1\times d_2} & 2cb \\ \mathbf{0}_{d_2\times d_1} & \alpha c^2 I_{d_2} & 2c\alpha b_o \\ 2cb^\top & 2c\alpha b_o^\top & \|b\|^2 + \alpha\|b_o\|^2 \end{bmatrix}. \tag{8}$$

The maximum eigenvalue $\lambda_{\max}$ at initialization is upper bounded by the maximum row sum of this matrix, and thus $\lambda_{\max} \le 3\frac{d_1+\alpha d_2}{d_1+d_2} < 3\alpha$. Clearly, we also have $\lambda_{\max} \ge \alpha$.

We observe that tracking the precise vectors b, b_o are not necessary to uncover the dynamics when optimizing this loss. First, let us write $b := \epsilon\frac{\beta}{\|\beta\|} + \delta v$, where v is the direction of the rejection of b from β (i.e., $\beta^\top v = 0$) and δ is its norm. Then we have the gradients

$$\nabla_\epsilon L = (\nabla_\epsilon b)^\top(\nabla_b L) \tag{9}$$

$$= \frac{\beta}{\|\beta\|}^\top\left(c^2\left(\epsilon\frac{\beta}{\|\beta\|} + \delta v\right) - c\beta\right) \tag{10}$$

$$= c^2\epsilon - c\|\beta\|, \tag{11}$$

$$\nabla_\delta L = (\nabla_\delta b)^\top(\nabla_b L) \tag{12}$$

$$= v^\top\left(c^2\left(\epsilon\frac{\beta}{\|\beta\|} + \delta v\right) - c\beta\right) \tag{13}$$

$$= c^2\delta, \tag{14}$$

$$\nabla_c L = \left(\epsilon\frac{\beta}{\|\beta\|} + \delta v\right)^\top\left(c\left(\epsilon\frac{\beta}{\|\beta\|} + \delta v\right) - \beta\right) + \alpha\|b_o\|^2 c \tag{15}$$

$$= c(\epsilon^2 + \delta^2 + \alpha\|b_o\|^2) - \epsilon\|\beta\|. \tag{16}$$

Finally, define the scalar quantity $o := \|b_o\|^2$, noting that $\nabla_o L = 2b_o^\top\nabla_{b_o}L = 2\alpha c^2 o$. Minimizing this loss via gradient flow is therefore characterized by the following ODE on four scalars:

$$\frac{d\epsilon}{dt} = -c^2\epsilon + c\|\beta\|, \tag{17}$$

$$\frac{d\delta}{dt} = -c^2\delta, \tag{18}$$

$$\frac{do}{dt} = -2\alpha c^2 o, \tag{19}$$

$$\frac{dc}{dt} = -c(\epsilon^2 + \delta^2 + \alpha o) + \epsilon\|\beta\|. \tag{20}$$

$$\tag{21}$$

Furthermore, we have the boundary conditions

$$\epsilon(0) = \sqrt{\frac{1}{d_1 + d_2}}, \tag{22}$$

$$\delta(0) = \sqrt{\frac{d_1 - 1}{d_1 + d_2}}, \tag{23}$$

$$o(0) = \frac{d_2}{d_1 + d_2}, \tag{24}$$

$$c(0) = 1. \tag{25}$$

Given these initializations and dynamics, we make a few observations: (i) all four scalars are initialized at a value greater than 0, and remain greater than 0 at all time steps; (ii) δ and o will decrease towards 0 monotonically, and ϵ will increase monotonically until $c\epsilon = \|\beta\|$; (iii) c will be decreasing at initialization. Lastly, we define the quantity $r := (\epsilon(0)^2 + \delta(0)^2 + \alpha o(0)) = \frac{d_1 + \alpha d_2}{d_1 + d_2}$ and $k := \frac{d_2}{d_1}$.

Before we can prove the main results, we present a lemma which serves as a key tool for deriving continuously valid bounds on the scalars we analyze:

Lemma I.1. *Consider a vector valued ODE with scalar indices $v_1, v_2, \ldots$, where each index is described over the time interval $[t_{\min}, t_{\max}]$ by the continuous dynamics $\frac{dv_i(t)}{dt} = a_i(v_{-i}(t)) \cdot v_i(t) + b_i(v_{-i}(t))$ with $a_i \leq 0, b_i \geq 0$ for all i, t (v_{-i} denotes the vector v without index i). That is, each scalar's gradient is an affine function of that scalar with a negative coefficient. Suppose we define continuous functions $\hat{a}_i, \hat{b}_i : \mathbb{R} \to \mathbb{R}$ such that $\forall i, t$, $\hat{a}_i(t) \leq a_i(v_{-i}(t))$ and $\hat{b}_i(t) \leq b_i(v_{-i}(t))$. Let $\hat{v}$ be the vector described by these alternate dynamics, with the boundary condition $\hat{v}_i(t_{\min}) = v_i(t_{\min})$ and $v_i(t_{\min}) \geq 0$ for all i (if a solution exists). Then for $t \in [t_{\min}, t_{\max}]$ it holds that*

$$\hat{v}(t) \leq v(t), \tag{26}$$

elementwise. If $\hat{a}_i, \hat{b}_i$ upper bound a_i, b_i, the inequality is reversed.

Proof. Define the vector $w(t) := \hat{v}(t) - v(t)$. This vector has the dynamics

$$\frac{dw_i}{dt} = \frac{d\hat{v}_i}{dt} - \frac{dv_i}{dt} \tag{27}$$

$$= \hat{a}_i(t) \cdot \hat{v}_i(t) + \hat{b}_i(t) - a_i(v_{-i}(t)) \cdot v_i(t) - b_i(v_{-i}(t)) \tag{28}$$

$$\leq \hat{a}_i(t) \cdot \hat{v}_i(t) - a_i(v_{-i}(t)) \cdot v_i(t). \tag{29}$$

The result will follow by showing that $w(t) \leq \mathbf{0}$ for all $t \in [t_{\min}, t_{\max}]$ (this clearly holds at $t_{\min}$). Assume for the sake of contradiction there exists a time $t' \in (t_{\min}, t_{\max}]$ and index i such that $w_i(t') > 0$ (let i be the first such index for which this occurs, breaking ties arbitrarily). By continuity, we can define $t_0 := \max \{t \in [t_{\min}, t'] \ : \ w_i(t) \leq 0\}$. By definition of t_0 it holds that $w_i(t_0) = 0$ and $\forall \epsilon > 0$, $w_i(t_0 + \epsilon) - w_i(t_0) = w_i(t_0 + \epsilon) > 0$, and thus $\frac{dw_i(t_0)}{dt} > 0$. But by the definition of w we also have

$$\hat{v}_i(t_0) = v_i(t_0) + w_i(t_0) \tag{30}$$

$$= v_i(t_0), \tag{31}$$

and therefore

$$\frac{dw_i(t_0)}{dt} \leq \hat{a}_i(t_0) \cdot \hat{v}_i(t_0) - a_i(v_{-i}(t_0)) \cdot v_i(t_0) \tag{32}$$

$$= \big(\hat{a}_i(t_0) - a_i(v_{-i}(t_0))\big) \cdot v_i(t_0) \tag{33}$$

$$\leq 0, \tag{34}$$

with the last inequality following because $\hat{a}_i(t) \leq a_i(v_{-i}(t))$ and $v_i(t) > 0$ for all $i, t \in [t_{\min}, t_{\max}]$. Having proven both $\frac{dw_i(t_0)}{dt} > 0$ and $\frac{dw_i(t_0)}{dt} \leq 0$, we conclude that no such t' can exist. The other direction follows by analogous argument. □

We make use of this lemma repeatedly and its application is clear so we invoke it without direct reference. We are now ready to prove the main results:

I.1 Proof of Theorem H.1

Proof. At initialization, we have $\|\beta\| \geq \frac{d_1}{\sqrt{d_1+d_2}} \implies \|\beta\|\epsilon(0) \geq \frac{d_1}{d_1+d_2} = c(0)(\epsilon(0)^2 + \delta(0)^2)$. Therefore, we can remove these terms from $\frac{dc}{dt}$ at time $t = 0$, noting simple that $\frac{dc}{dt} \geq -\alpha o c$. Further, so long as c is still decreasing (and therefore less than $c(0) = 1$),

$$\begin{aligned}
\frac{d(\|\beta\|\epsilon - c(\epsilon^2+\delta^2))}{dt} &\geq \frac{d(\|\beta\|\epsilon - (\epsilon^2+\delta^2))}{dt} && (35)\\
&= (\|\beta\| - 2\epsilon)\frac{d\epsilon}{dt} - 2\delta\frac{d\delta}{dt} && (36)\\
&= (\|\beta\| - 2\epsilon)(-c^2\epsilon + \|\beta\|c) - 2\delta(-c^2\delta) && (37)\\
&= -c^2(\epsilon\|\beta\| - 2(\epsilon^2+\delta^2) + c(\|\beta\|^2 - 2\epsilon) && (38)\\
&\geq -c(\epsilon\|\beta\| - 2(\epsilon^2+\delta^2)) + c(\|\beta\|^2 - 2\epsilon) && (39)\\
&= c(\|\beta\|^2 - 2\epsilon - \epsilon\|\beta\| + 2(\epsilon^2+\delta^2)) && (40)\\
&\geq c(\|\beta\|^2 - \epsilon(2 + \|\beta\|)). && (41)
\end{aligned}$$

Since $c > 0$ at all times, this is non-negative so long as the term in parentheses is non-negative, which holds so long as $\epsilon \leq \frac{\|\beta\|^2}{\|\beta\|+2}$. Further, since $\epsilon c \leq \|\beta\|$ we have

$$\begin{aligned}
\frac{d\epsilon^2}{dt} &= 2\epsilon\frac{d\epsilon}{dt} && (42)\\
&= -2c^2\epsilon^2 + 2\epsilon c\|\beta\| && (43)\\
&\leq 2\|\beta\|^2. && (44)
\end{aligned}$$

This implies $\epsilon(t)^2 \leq \epsilon(0)^2 + 2t\|\beta\|^2$. Therefore, for $t \leq \frac{\ln \|\beta\|/2}{2\|\beta\|}$ we have $\epsilon(t)^2 \leq \frac{1}{d_1+d_2} + \|\beta\| \ln \|\beta\|/2 \leq \frac{\|\beta\|^4}{(\|\beta\|+2)^2}$ (this inequality holds for $\|\beta\| \geq 2$). This satisfies the desired upper bound.

Thus the term in Eq. (41) is non-negative for all $t \leq \frac{\ln \|\beta\|/2}{2\|\beta\|}$, and so we have $\frac{dc}{dt} \geq -\alpha o c$ under the above conditions. Since the derivative of o is negative in c, a lower bound on $\frac{dc}{dt}$ gives us an upper bound on $\frac{do}{dt}$, which in turn maintains a valid lower bound on $\frac{dc}{dt}$ This allows us to solve for just the ODE given by

$$\begin{aligned}
\frac{dc^2}{dt} &= -2\alpha c^2 o, && (45)\\
\frac{do}{dt} &= -2\alpha c^2 o. && (46)
\end{aligned}$$

Define $m := \frac{d_1}{d_1+d_2} = \frac{1}{1+k}$. Recalling the initial values of c^2, o, The solution to this system is given by

$$\begin{aligned}
c(t)^2 &= \frac{m}{1 - \frac{(1-m)}{\exp(2\alpha m t)}}, && (47)\\
o(t) &= \frac{m}{\frac{\exp(2\alpha m t)}{1-m} - 1} && (48)\\
&= \frac{m}{\exp(2\alpha m t)(1+k^{-1}) - 1} && (49)
\end{aligned}$$

Since these are bounds on the original problem, we have $c(t)^2 \geq m$ and $o(t)$ shrinks exponentially fast in t. In particular, note that under the stated condition $\sqrt{\alpha} \geq \frac{\|\beta\| \ln k}{m(\ln \|\beta\|/2)}$ (recalling $k := \frac{d_2}{d_1} > 1$), we have $\frac{\ln k}{2\sqrt{\alpha} m} \leq \frac{\ln \|\beta\|/2}{2\|\beta\|}$. Therefore we can plug in this value for t, implying $o(t) \leq m\left(\frac{d_1}{d_2}\right)^{\sqrt{\alpha}} = mk^{-\sqrt{\alpha}}$ at some time before $t = \frac{\ln \|\beta\|/2}{2\|\beta\|}$.

Now we solve for the time at which $\frac{dc}{dt} \geq 0$. Returning to Eq. (41), we can instead suppose that $\epsilon \leq \frac{\|\beta\|^2 - \gamma}{\|\beta\|+2} \implies \|\beta\|^2 - \epsilon(2 + \|\beta\|) \geq \gamma$ for some $\gamma > 0$. If this quantity was non-negative and

has had a derivative of at least γ until time $t = \frac{\ln k}{2\sqrt{\alpha}m}$, then its value at that time must be at least $\frac{\gamma \ln k}{2\sqrt{\alpha}m}$. For $\frac{dc}{dt}$ to be non-negative, we need this to be greater than $c(t)^2\alpha o(t)$, so it suffices to have $\frac{\gamma \ln k}{2\sqrt{\alpha}m} \geq \frac{\alpha m}{\exp(2\alpha m t)(1+k^{-1})-1} \iff \gamma \ln k \geq \frac{2\alpha^{3/2}m^2}{\left(\frac{d_2}{d_1}\right)^{\sqrt{\alpha}}(1+k^{-1})-1} \iff \gamma \geq \frac{2\alpha^{3/2}m^2 k^{-\sqrt{\alpha}}}{\ln k}$. Observe that the stated lower bound on α directly implies this inequality.

Finally, note that $\|b\|^2 = \epsilon^2 + \delta^2$, and therefore

$$\frac{d\|b\|^2}{dt} = 2\epsilon\frac{d\epsilon}{dt} + 2\delta\frac{d\delta}{dt} \tag{50}$$

$$= -2c^2(\epsilon^2 + \delta^2) + 2c\epsilon\|\beta\|. \tag{51}$$

Since $c(0) = 1$ and $c\epsilon < \|\beta\|$, this means $\|b\|^2$ will also be decreasing at initialization. Thus we have shown that all relevant quantities will decrease towards 0 at initialization, but that by time $t = \frac{\ln k}{2\sqrt{\alpha}m}$, we will have $\frac{dc}{dt} \geq 0$. □

I.2 Proof of Proof of Theorem H.2

Proof. Recall from the previous section that we have shown that at some time $t_1 \leq \frac{\ln k}{2\sqrt{\alpha}m}$, $c(t)^2$ will be greater than m and increasing, and $o(t)$ will be upper bounded by $mk^{-\sqrt{\alpha}}$. Furthermore, $\epsilon(t)^2 \leq \frac{1}{d_1+d_2} + 2t\|\beta\|^2$. To show that the sharpness reaches a particular value, we must demonstrate that c grows large enough before the point $c\epsilon \approx \|\beta\|$ where this growth will rapidly slow. To do this, we study the relative growth of c vs. ϵ.

Recall the derivatives of these two terms:

$$\frac{dc}{dt} = -(\epsilon^2 + \delta^2 + \alpha o^2)c + \|\beta\|\epsilon, \tag{52}$$

$$\frac{d\epsilon}{dt} = -c^2\epsilon + \|\beta\|c. \tag{53}$$

Considering instead their squares,

$$\frac{dc^2}{dt} = 2c\frac{dc}{dt} \tag{54}$$

$$= -2(\epsilon^2 + \delta^2 + \alpha o^2)c^2 + 2\|\beta\|\epsilon c, \tag{55}$$

$$\frac{d\epsilon^2}{dt} = 2\epsilon\frac{d\epsilon}{dt} \tag{56}$$

$$= -2\epsilon^2 c^2 + 2\|\beta\|\epsilon c. \tag{57}$$

Since δ, o decrease monotonically, we have $\frac{dc^2}{dt} \geq -2(\epsilon^2 + \frac{d_1}{d_1+d_2} + \alpha m\left(\frac{d_1}{d_2}\right)^{\sqrt{\alpha}})c^2 + 2\|\beta\|\epsilon$. Thus if we can show that

$$\|\beta\|\epsilon c \geq (\epsilon^2 + 2(\frac{d_1}{d_1+d_2} + \alpha m\left(\frac{d_1}{d_2}\right)^{\sqrt{\alpha}}))c^2, \tag{58}$$

we can conclude that $\frac{dc^2}{dt} \geq (\epsilon^2 c^2 + \|\beta\|\epsilon c) = \frac{1}{2}\frac{d\epsilon^2}{dt}$—that is, that $c(t)^2$ grows at least half as fast as $\epsilon(t)^2$. And since δ, o continue to decrease, this inequality will continue to hold thereafter.

Simplifying the above desired inequality, we get

$$\|\beta\|\frac{\epsilon}{c} \geq \epsilon^2 + 2m(1 + \alpha k^{-\sqrt{\alpha}}). \tag{59}$$

Noting that $\frac{\epsilon}{c} \geq 1$ and $m = \frac{d_1}{d_1+d_2} \leq \frac{1}{2}$, and recalling the upper bound on $\epsilon(t)^2$, this reduces to proving

$$\|\beta\| \geq \frac{1}{d_1+d_2} + 2t\|\beta\|^2 + 1 + \alpha k^{-\sqrt{\alpha}}. \tag{60}$$

Since this occurs at some time $t_1 \leq \frac{\ln k}{2\sqrt{\alpha}m}$, and since $m^{-1} = 1 + k$, we get

$$\|\beta\| \geq \frac{1}{d_1 + d_2} + \frac{\|\beta\|^2(1+k)\ln k}{\sqrt{\alpha}} + 1 + \alpha k^{-\sqrt{\alpha}}. \tag{61}$$

The assumed lower bound on $\sqrt{\alpha}$ means the sum of the first three terms can be upper bounded by a small $1 + o(1)$ term (say, $9/5$) and recalling $\|\beta\| \geq 24/5$ it suffices to prove

$$\|\beta\| \geq \frac{9}{5} + \alpha k^{-\sqrt{\alpha}} \tag{62}$$

$$\Longleftarrow \alpha k^{-\sqrt{\alpha}} \leq 3. \tag{63}$$

Taking logs,

$$\frac{2\ln\sqrt{\alpha}}{\ln k} - \sqrt{\alpha} \leq \ln 3, \tag{64}$$

which is clearly satisfied for $\sqrt{\alpha} \geq 1 + k\ln k$. As argued above, this implies $\frac{dc^2}{dt} \geq \frac{1}{2}\frac{d\epsilon^2}{dt}$ by some time $t_2 \leq \frac{\ln k}{2\sqrt{\alpha}m}$.

Consider the time t_2 at which this first occurs, whereby $c(t_2)^2$ is growing by at least one-half the rate of $\epsilon(t_2)^2$. Here we note that we can derive an upper bound on c and ϵ at this time using our lemma and the fact that

$$\frac{dc}{dt} \leq \|\beta\|\epsilon, \tag{65}$$

$$\frac{d\epsilon}{dt} \leq \|\beta\|c. \tag{66}$$

The solution to this system implies

$$c(t_2) \leq \frac{1}{2}\left(\frac{\exp(\|\beta\|t_2) - \exp(-\|\beta\|t_2)}{\sqrt{d_1+d_2}} + \exp(\|\beta\|t_2) + 1\right) \tag{67}$$

$$\leq \frac{1}{2}\left(\exp(\|\beta\|t_2)\left(1 + \frac{1}{\sqrt{d_1+d_2}}\right) + 1\right) \tag{68}$$

$$\leq \frac{1}{2}\left(\exp\left(\frac{\|\beta\|\ln k}{2\sqrt{\alpha}m}\right)\left(1 + \frac{1}{\sqrt{d_1+d_2}}\right) + 1\right), \tag{69}$$

$$\epsilon(t_2) \leq \frac{1}{2}\left(\exp(\|\beta\|t_2)\left(1 + \frac{1}{\sqrt{d_1+d_2}}\right) + \frac{1}{\sqrt{d_1+d_2}} - 1\right) \tag{70}$$

$$\leq \frac{1}{2}\left(\exp\left(\frac{\|\beta\|\ln k}{2\sqrt{\alpha}m}\right)\left(1 + \frac{1}{\sqrt{d_1+d_2}}\right) + \frac{1}{\sqrt{d_1+d_2}} - 1\right) \tag{71}$$

Then for $\alpha > \left(\frac{\|\beta\|\ln\frac{d_2}{d_1}}{m(\ln\|\beta\| - \ln 2)}\right)^2$, the exponential term is upper bounded by $\frac{\sqrt{\|\beta\|}}{2}$, giving

$$c(t_2) \leq \frac{1}{2}\left(\frac{\sqrt{\|\beta\|}}{2}\left(1 + \frac{1}{\sqrt{d_1+d_2}}\right) + 1\right) \tag{72}$$

$$\leq \frac{\sqrt{\|\beta\|}}{2}, \tag{73}$$

$$\epsilon(t_2) \leq \frac{1}{2}\left(\frac{\sqrt{\|\beta\|}}{2}\left(1 + \frac{1}{\sqrt{d_1+d_2}}\right) + \frac{1}{\sqrt{d_1+d_2}} - 1\right) \tag{74}$$

$$\leq \frac{\sqrt{\|\beta\|}}{2}. \tag{75}$$

We know that optimization will continue until $\epsilon^2 c^2 = \|\beta\|^2$, and also that $\frac{dc^2}{dt} \geq \frac{1}{2}\frac{d\epsilon^2}{dt}$. Since $c \leq \epsilon$, this implies that $\epsilon^2 \geq \|\beta\|$ before convergence. Suppose that starting from time t_2, ϵ^2 grows until

time t' by an additional amount s. Then we have

$$s = \epsilon(t')^2 - \epsilon(t_2)^2 \tag{76}$$

$$= \int_{t_2}^{t'} \frac{d\epsilon(t)^2}{dt} \tag{77}$$

$$\leq \int_{t_2}^{t'} 2\frac{dc(t)^2}{dt} \tag{78}$$

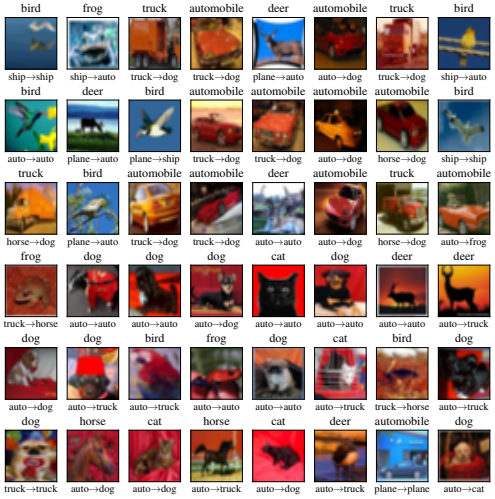
$$= 2(c(t')^2 - c(t_2)^2). \tag{79}$$

In other words, c^2 must have grown by at least half that amount. Since $\epsilon(t_2)^2 \leq \frac{\|\beta\|}{4}$ and therefore $\epsilon(t')^2 \leq \frac{\|\beta\|}{4} + s$, even if $c(t')^2$ is the minimum possible value of $\frac{s}{2}$ we must have at convergence $\frac{s}{2} = c^2 = \frac{\|\beta\|^2}{\epsilon^2} \geq \frac{\|\beta\|^2}{\frac{\|\beta\|}{4}+s}$. This is a quadratic in s and solving tells us that we must have $s \geq \frac{5}{4}\|\beta\|$. Therefore, $c(t')^2 \geq \frac{5}{8}\|\beta\|$ is guaranteed to occur. Noting our derivation of the loss Hessian, this implies the sharpness must reach at least $\frac{5}{8}\alpha\|\beta\|$ for each dimension of b_o. □

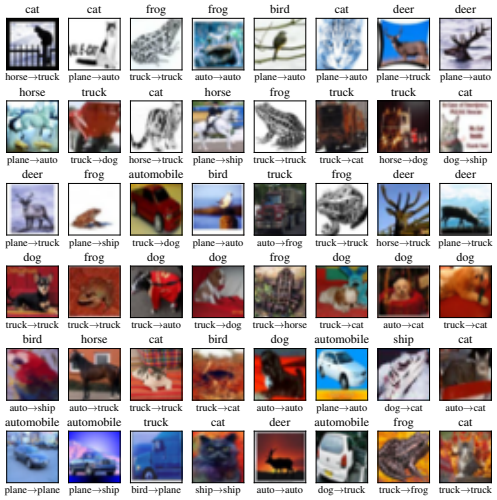
J Additional Samples Under Various Architectures/Seeds

To demonstrate the robustness of our finding we train a ResNet-18, VGG-11, and a Vision Transformer for 1000 steps with full-batch GD, each with multiple random initializations. For each run, we identify the 24 training examples with the most positive and most negative change in loss from step i to step $i+1$, for $i \in \{100, 250, 500, 750\}$. We then display these images along with their label (above) and the network's predicted label before and after the gradient step (below). The change in the network's predicted labels display a clear pattern, where certain training samples cause the network to associate an opposing signal with a new class, which the network then overwhelmingly predicts whenever that feature is present.

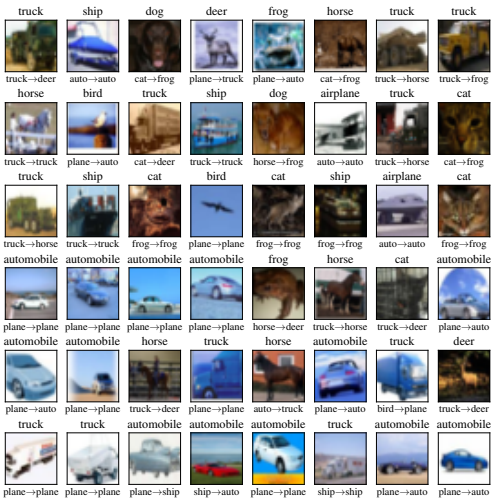
Consistent with our other experiments, we find that early opposing signals tend to be "simpler", e.g. raw colors, whereas later signals are more nuanced, such as the presence of a particular texture. We also see that the Vision Transformer seems to learn complex features earlier, and that they are less obviously aligned with human perception—this is not surprising since they process inputs in a fundamentally different manner than traditional ConvNets.



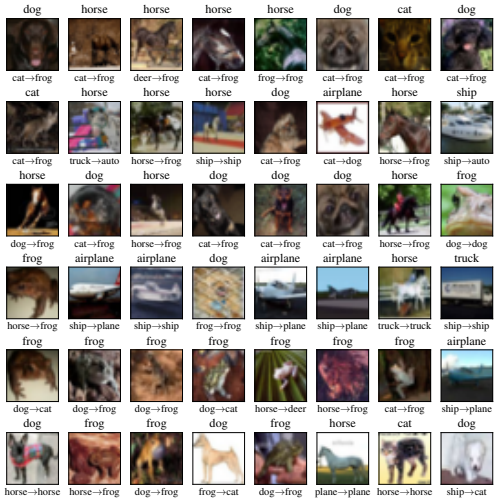
(a) Step 100 to 101



(b) Step 250 to 251

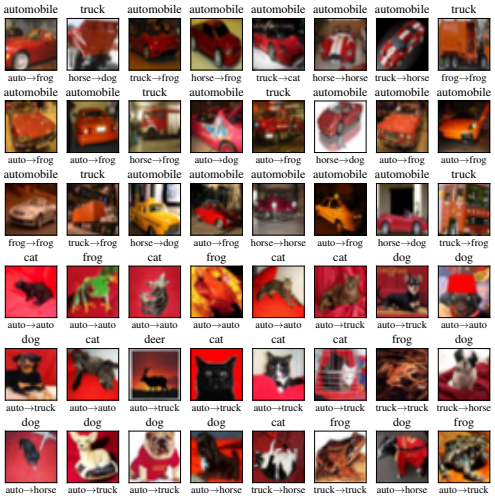


(c) Step 500 to 501

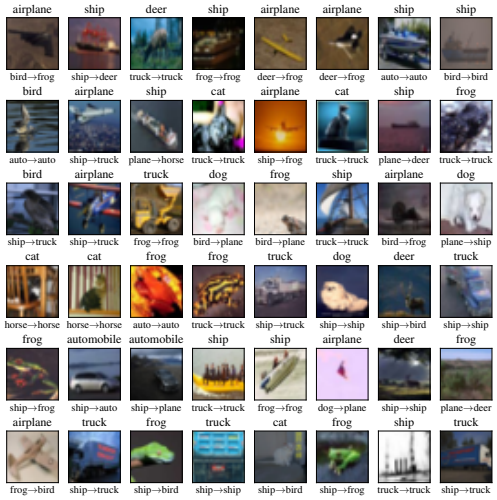


(d) Step 750 to 751

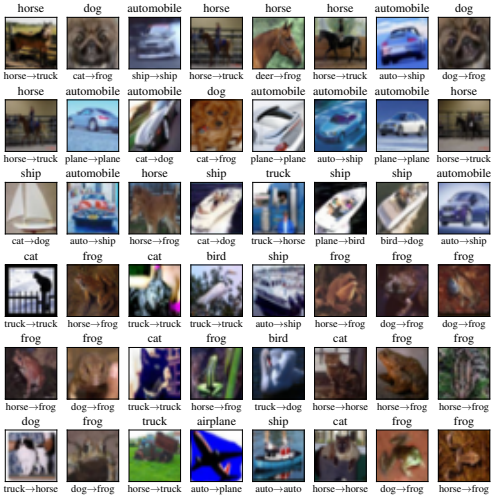
Figure 36: **(ResNet-18, seed 1)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).



(a) Step 100 to 101



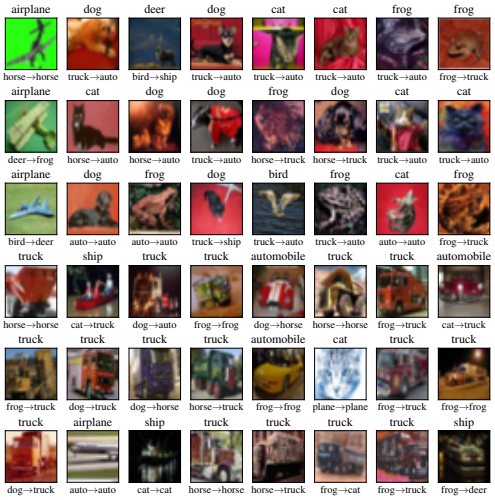
(b) Step 250 to 251



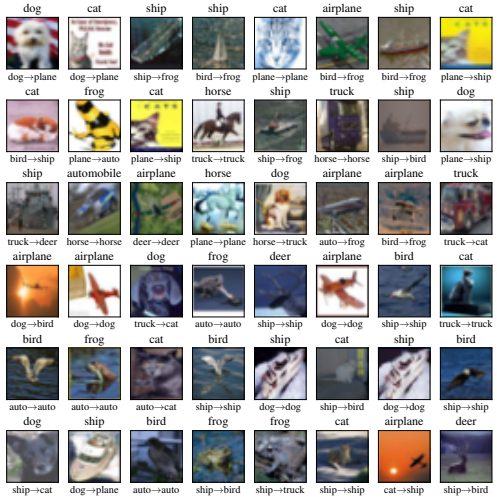
(c) Step 500 to 501

(d) Step 750 to 751

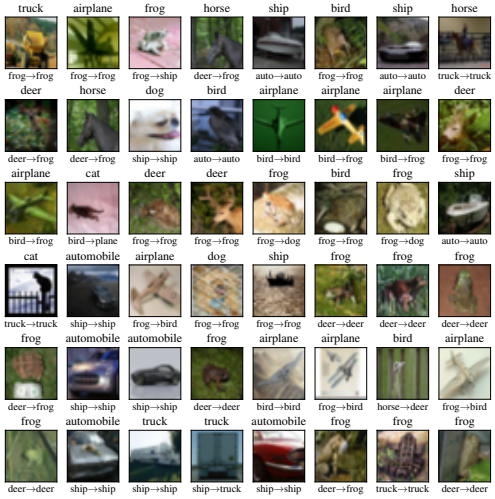
Figure 37: **(ResNet-18, seed 2)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).



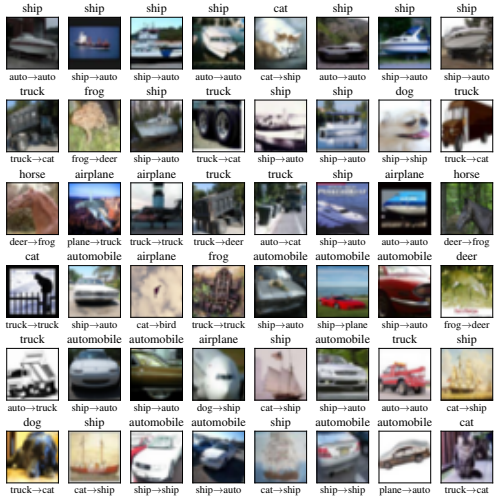
(a) Step 100 to 101



(b) Step 250 to 251

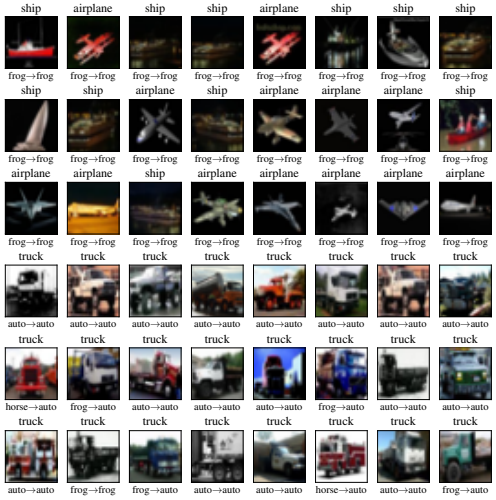


(c) Step 500 to 501

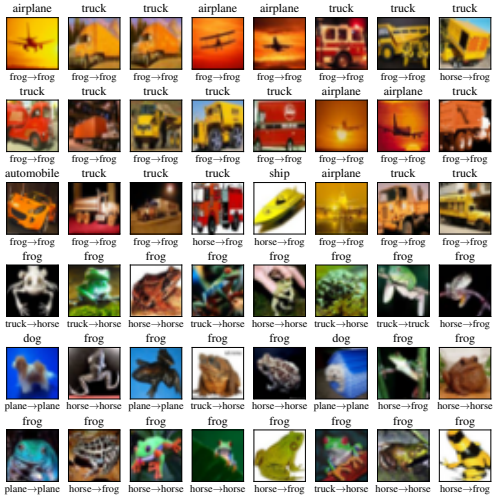


(d) Step 750 to 751

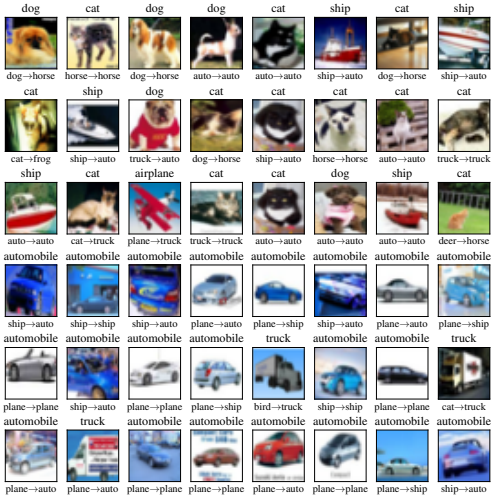
Figure 38: **(ResNet-18, seed 3)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).



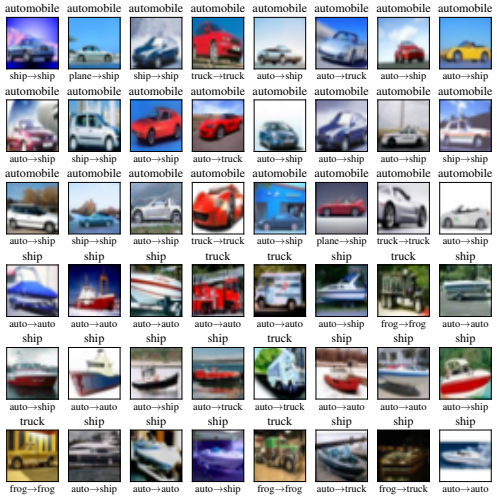
(a) Step 100 to 101



(b) Step 250 to 251

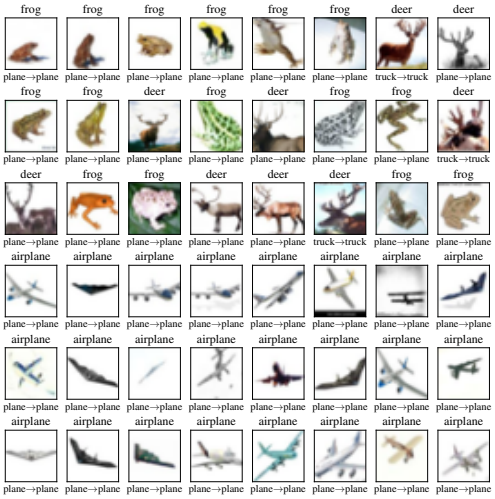


(c) Step 500 to 501

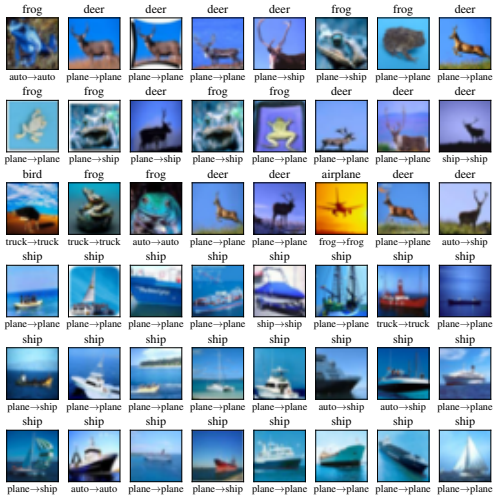


(d) Step 750 to 751

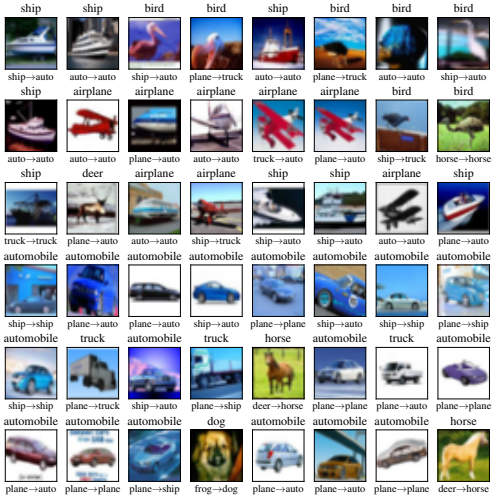
Figure 39: **(VGG-11, seed 1)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).



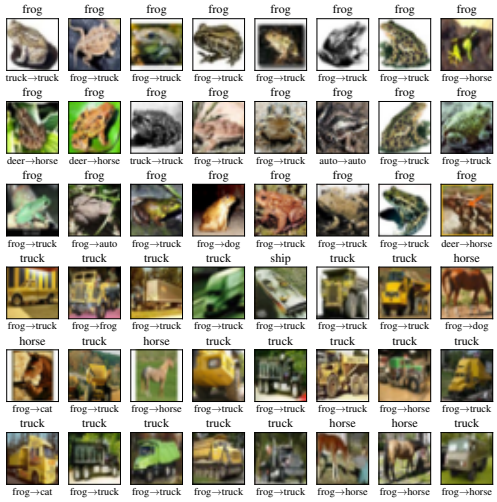
(a) Step 100 to 101



(b) Step 250 to 251



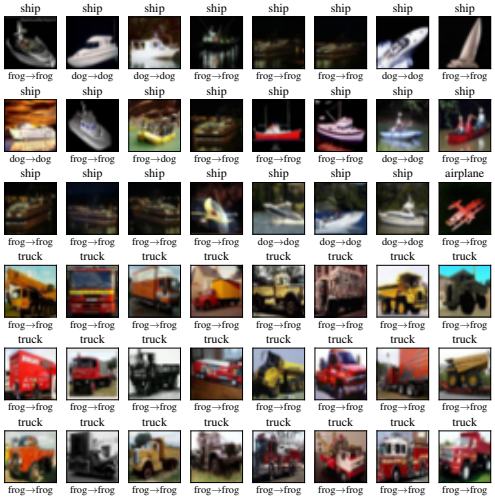
(c) Step 500 to 501



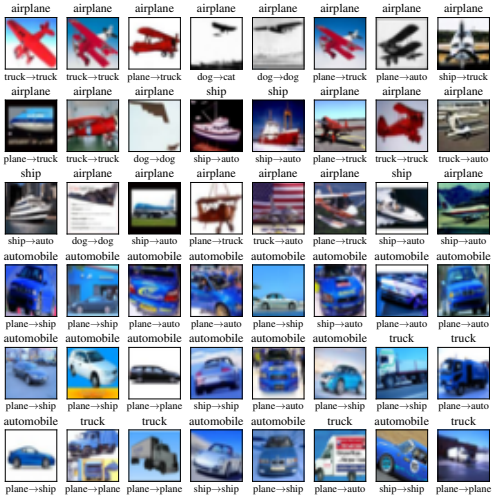
(d) Step 750 to 751

Figure 40: **(VGG-11, seed 2)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).

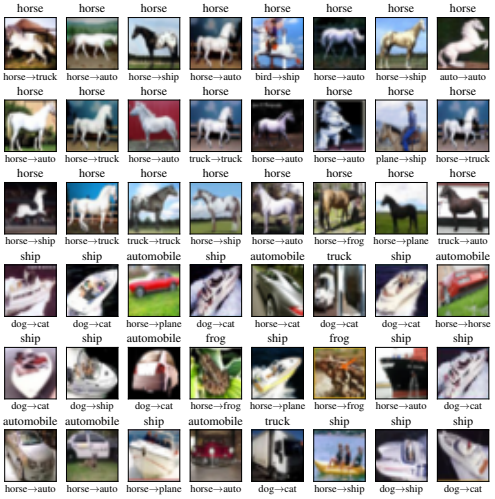
(a) Step 100 to 101



(b) Step 250 to 251

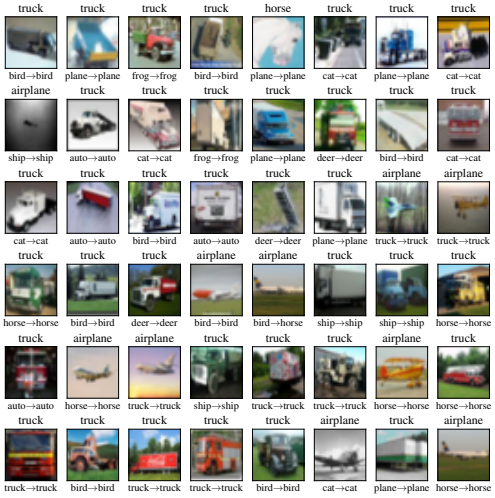


(c) Step 500 to 501

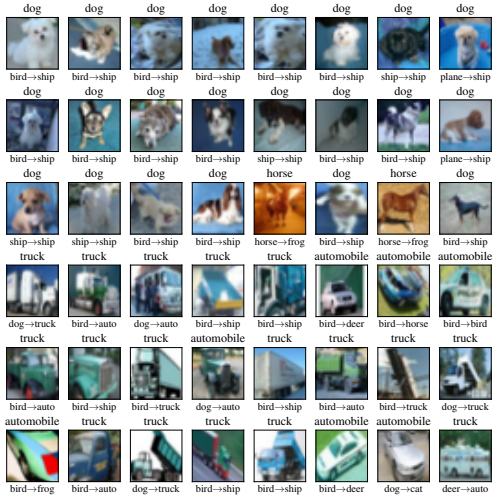


(d) Step 750 to 751

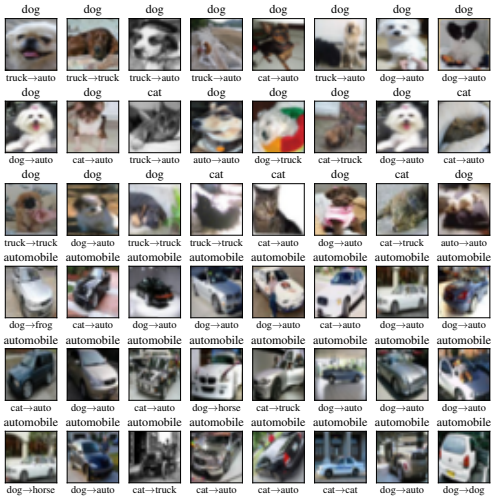
Figure 41: **(VGG-11, seed 3)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).



(a) Step 100 to 101



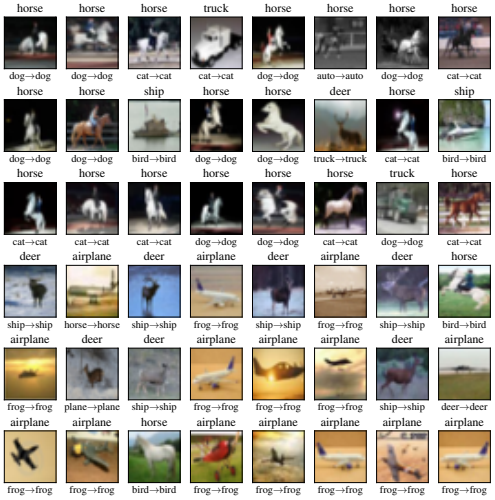
(b) Step 250 to 251



(c) Step 500 to 501

(d) Step 750 to 751

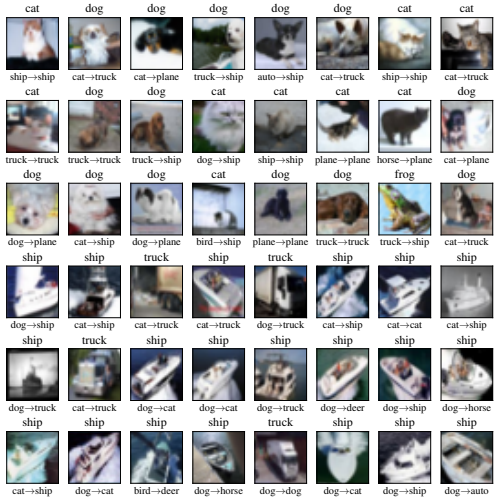
Figure 42: **(ViT, seed 1)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).



(a) Step 100 to 101

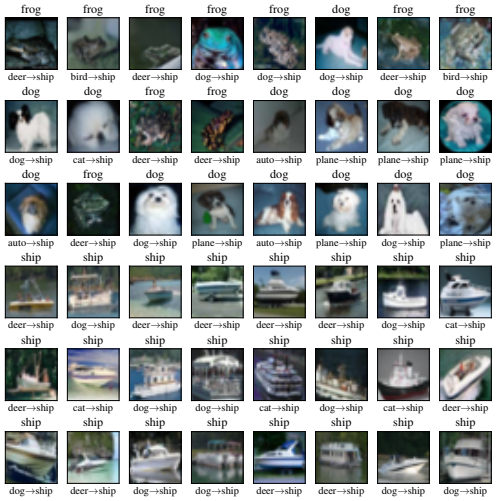
(b) Step 250 to 251

(c) Step 500 to 501

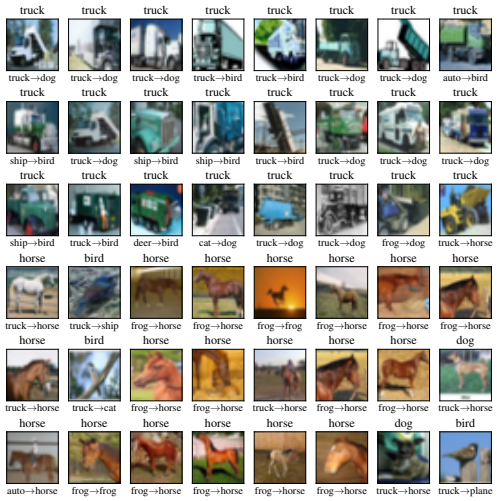


(d) Step 750 to 751

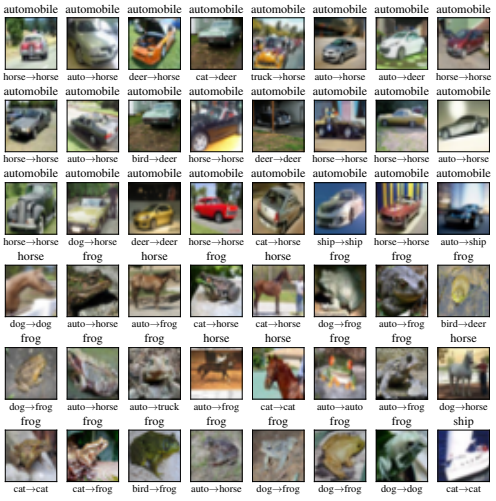
Figure 43: **(ViT, seed 2)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).



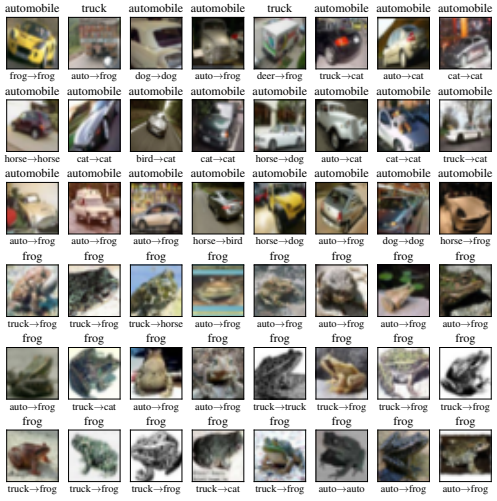
(a) Step 100 to 101



(b) Step 250 to 251



(c) Step 500 to 501



(d) Step 750 to 751

Figure 44: **(ViT, seed 3)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).