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, including a conceptually new cause for progressive sharpening and the edge of stability. It also enables new predictions of training behavior which we confirm experimentally, plus a new lens through which to theoretically study and improve modern stochastic optimization on neural nets.

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.

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. These phenomena are typically considered in isolation—though they are not completely disparate, it is unknown what specific underlying factors they may share. However, it is clear that any commonality will be a valuable tool for further analysis.

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 (and do not claim to) give a complete explanation, we present strong qualitative and quantitative evidence for a single high-level idea which naturally fits into several existing narratives and suggests a more coherent picture of their origin. Specifically, we demonstrate the prevalence of paired groups of outliers in natural data which have a significant influence on a network's optimization dynamics. These groups are characterized by the inclusion of one or more large magnitude features that dominate the network's output throughout most of training. The other distinctive property of these features is that they provide large, consistent, and *opposing* gradients; because of this structure, we refer to them as *Opposing Signals*. These features share a non-trivial correlation with the target task, but they are often not the "correct" (e.g., human-aligned) signal. In fact, in many cases these features perfectly encapsulate the classic statistical conundrum of "correlation vs. causation".

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 2 and Appendix B. 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.

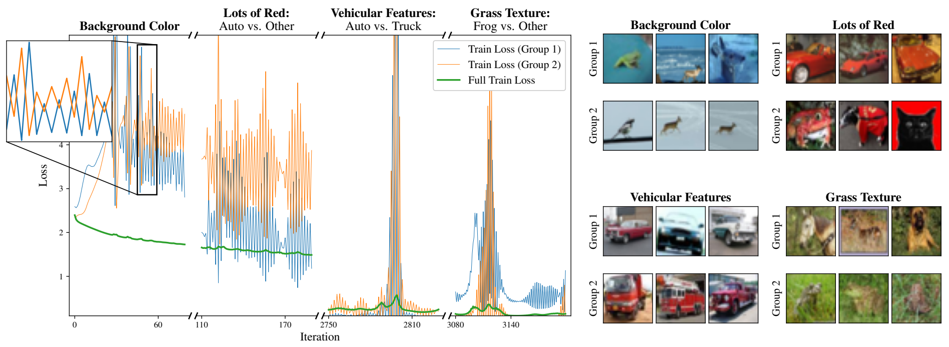


Figure 1: **Training dynamics of neural networks are heavily influenced by outliers with opposing signals.** We plot the overall loss of a ResNet-18 trained with GD on CIFAR-10, plus the losses of a small but representative subset of outlier groups. These groups have consistent *opposing signals* (e.g., wheels and headlights can mean either `car` or `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*.

Early in training, the network enters a narrow valley which balances the pairs' opposing gradients; subsequent sharpening of the loss landscape [21, 7] causes the network to oscillate with growing magnitude, upsetting this balance. Returning to our example, one step might result 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 result is that the network's loss on images of planes with a sky background will alternate between 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—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 this is the direct cause of the "edge-of-stability" phenomenon [7].

We repeat this experiment across a range of vision architectures and training hyperparameters: though the precise groups and their order of appearance change, the pattern occurs consistently. We also verify this behavior for transformers on next-token prediction of natural text and small ReLU MLPs on simple 1D functions; we give some examples of opposing signals in text in Appendix C. 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 3.

Contributions. The primary contribution of this paper is demonstrating the existence, pervasiveness, and large influence of opposing signals during NN optimization. We further present our current best understanding, with supporting experiments, of how these signals *cause* the observed training dynamics—in particular, we provide evidence 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 qualitative predictions of NN behavior during training, which we confirm experimentally. It also provides a new lens through which to study modern stochastic optimization methods, which we highlight via a case study of SGD vs. Adam. **We believe in many cases our result serves as direct evidence for the validity of the assumptions made by prior theoretical analyses of related phenomena, and that it may enable even more fine-grained analyses in the future.** We also see possible connections between opposing signals and a wide variety of existing phenomena in NN optimization, including "grokking" [45], "catapulting" [28, 54], simplicity bias [55], and Sharpness-Aware Minimization [11]. We discuss these connections along with further related work in Appendix A.

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). Some examples of such sets are given for several architectures and seeds in Appendix L. Given how these samples were selected, several other characterizations seem relevant, such as maximum gradient norm or largest eigenvalue of the loss of the *individual*

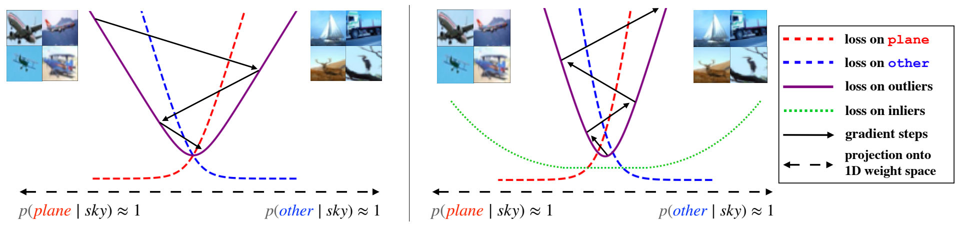

loss on plane
loss on other
loss on outliers
loss on inliers
gradient steps
projection onto 1D weight space
$p(plane \mid sky) \approx 1$
$p(other \mid sky) \approx 1$

Figure 2: We project the loss to the hypothetical weight-space dimension "sky = `plane`". **Left:** Early optimization approaches the minimum; further optimization grows the linear head. **Right**: The valley sharpens and the iterates diverge. Because these images are a small minority, the train loss is not noticeably affected. Eventually either (a) the network is forced to downweight "sky", returning to the first phase; or (b) the weights "catapult" to a different basin.

point. For large networks these options are far more compute-intensive, but we can evaluate them post-hoc. Fig. 32 in the Appendix tracks these metrics for various outliers; we find that they are consistently much larger than that of random samples from the training set.

2 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 Appendix B we give a simplified "mental picture" which serves as 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. Here we give a (very) brief summary of this discussion:

At initialization, a NN's features will typically be dominated by only slightly useful input variation. Training then aligns adjacent layers' singular values [51, 36] to amplify meaningful signal while downweighting noise. 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. The increase of this sensitivity represents precisely the growth of loss Hessian spectrum, with the strength of this effect increasing with depth [10, 36]. Crucially, as we argue in the Appendix, *genuine signals which oppose each other* will cause more consistent sharpening than traditionally envisioned "random noise". Next, to gain a more mechanistic understanding, we illustrate the precise dynamics on a toy example. We include in the main body the figure of this example with a high-level caption, but we again relegate the more detailed discussion to Appendix B. Though this explanation lacks precise details, it does enable concrete predictions of network behavior during training. Fig. 6 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 [53] in the appendix, with similar findings.

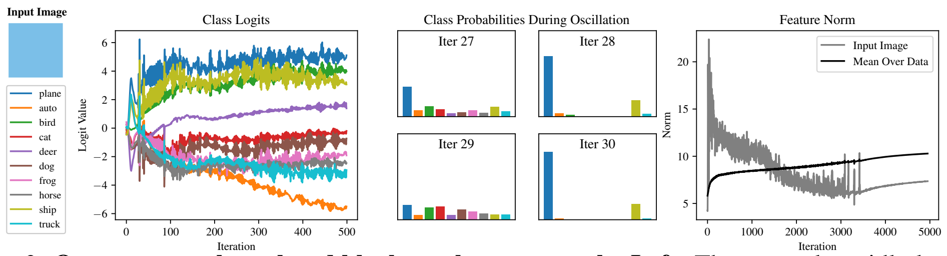


Figure 3: **Outputs on a sky-colored block track our example. Left:** The network rapidly learns to use the sky, amplifying the feature and sharpening the loss. **Middle:** During oscillation, gradient steps alternate along the axis "sky = `plane`". **Right:** After amplifying the sky input, the network then slowly downweights this feature and learns to use other signal.

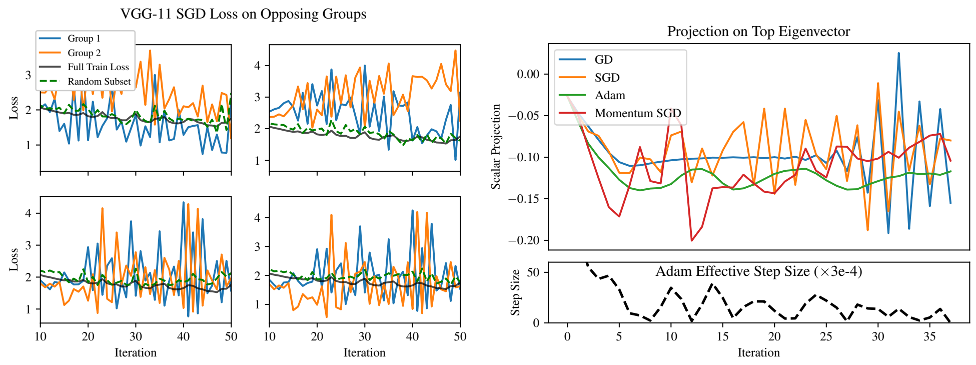


Figure 4: **Left:** Losses of paired outlier groups during SGD, along with the full train loss for comparison. The outliers' loss follow the same oscillatory pattern. See appendix for the same without BN. **Right:** SGD closely tracks GD; momentum somewhat mitigates the sharp jumps. In contrast, Adam smoothly oscillates along one side.

Theoretical analysis of opposing signals in a simple model. To demonstrate this effect more concretely, in Appendix J we study misspecified regression 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; we view this analysis only as an early investigation. We study the trajectory of gradient flow, proving first an initial decrease in sharpness due to the model downweighing the noise, followed by a continuous, steady growth in sharpness as the signal is amplified. We then argue that under gradient descent with large enough step size, the sharpness will cross the stability threshold, at which point the network will begin to *reintroduce* the opposing signal, which we confirm with syntehtica experiments. We also demonstrate identical behavior on an MLP trained on a 5k subset of CIFAR-10.

We make a few more observations which seem relevant for a more theoretical understanding of the effect of these outliers. We list them here, with a more in-depth discussion in Appendix E: (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.

3 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 (left) 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. We reproduce this plot with a VGG-11 without BN in Fig. 35 in the Appendix. Having verified that this behavior occurs in the stochastic setting, we conjecture that current best practices for neural network optimization owe much of their success to gracefully handling opposing signals. As a proof of concept, we present a preliminary investigation of this possibility for the Adam optimizer [23].

To better understand their differences, Fig. 4 (right) visualizes the parameter iterates of Adam and Momentum SGD on an MLP trained on a 5k subset of CIFAR, alongside GD and SGD. The top figure is the projection of these parameters onto the top eigenvector of the loss Hessian of the GD-trained network. We observe that SGD tracks a similar path to GD, though adding momentum mitigates the oscillation. In contrast, Adam markedly departs from this pattern, smoothly oscillating along one side. We identify several components of Adam which potentially contribute to this effect, which we expand upon in detail in Appendix H: (i) normalization means Adam takes very small steps near the minimum; (ii) the "trust region" enforces reduced dependence on imbalanced opposing signals, and means the gradient doesn't point too sharply "down the valley"; and (iii) Adam's EMA gradient uses *dampening* in addition to the usual momentum term, which we show is a surprisingly important inclusion. 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 I we discuss this approach in more detail and show that it matches Adam in multiple settings and with learning rates across several orders of magnitude.

4 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, 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 or simply coincides with it. 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 is somehow inherently tied to their instability.

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] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[4] 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.

[5] 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.

[6] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2(5):6, 2021.

[7] 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`.

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

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

[10] 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.

[11] 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`.

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

[13] 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`.

[14] Matteo Gamba, Hossein Azizpour, and Mårten Björkman. On the lipschitz constant of deep networks and double descent. *arXiv preprint arXiv:2301.12309*, 2023.

[15] 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`.

[16] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus, 2019.

[17] 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.

[18] 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.

[19] 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.

[20] 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`.

[21] 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`.

[22] 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`.

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

[24] 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.

[25] 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`.

[26] Ananya Kumar, Ruoqi Shen, Sébastien Bubeck, and Suriya Gunasekar. How to fine-tune vision models with sgd. *arXiv preprint arXiv:2211.09359*, 2022.

[27] Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=a65YK0cqH8g`.

[28] 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.

[29] 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.

[30] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[31] 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`.

[32] 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.

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

[34] Karttikeya Mangalam and Vinay Uday Prabhu. Do deep neural networks learn shallow learnable examples first? In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019. URL `https://openreview.net/forum?id=HkxHv4rn24`.

[35] 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`.

[36] 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`.

[37] 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.

[38] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1g5sA4twr`.

[39] 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`.

[40] 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.

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

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

[43] 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`.

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

[45] 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.

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

[47] Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization. *arXiv preprint arXiv:2202.06856*, 2022.

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

[49] 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.

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

[51] 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.

[52] 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.

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

[54] 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.

[55] 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`.

[56] Shengjie Wang, Abdel-rahman Mohamed, Rich Caruana, Jeff Bilmes, Matthai Plilipose, Matthew Richardson, Krzysztof Geras, Gregor Urban, and Ozlem Aslan. Analysis of deep neural networks with extended data jacobian matrix. In *International Conference on Machine Learning*, pages 718–726. PMLR, 2016.

[57] 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`.

[58] Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. Sharpness minimization algorithms do not only minimize sharpness to achieve better generalization. *arXiv preprint arXiv:2307.11007*, 2023.

[59] 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`.

[60] 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`.

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

[62] 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.

[63] 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`.

[64] 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

Many of the observations we make in this paper are not new, having been described in various prior works. Rather, this work identifies a possible *higher-order cause* which neatly ties these findings together. There are also many works which pursue a more theoretical understanding of each of these phenomena independently. Such analyses begin with a set of assumptions (on the data, in particular) and prove that the given behavior follows. In contrast, this work *begins* by identifying a condition—the presence of opposing signals—which we argue is likely a major cause of these behaviors. These two are not at odds: we believe in many cases our result serves as direct evidence for the validity of these modeling assumptions and that it may enable even more fine-grained analyses. This work provides an initial investigation which we hope will inspire future efforts towards a more complete understanding.

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 [48, 49, 42, 43, 12, 15, 29, 44, 24]. Later efforts focused on concretely linking these observations to corresponding behavior, often with an emphasis on SGD's bias towards particular solutions [59, 19, 21] and what this may imply about its resulting generalization [20, 64, 60]. Our method for identifying these paired groups, along with Fig. 32, 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 [20, 21] and GD [24, 7] (so-called "progressive sharpening"). This leads to rapid oscillation in weight space [61, 20, 7, 8]. Cohen et al. [7] 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 [63, 25]; several are deeply related to our findings. Ma et al. [32] connect this behavior to the existence of multiple "scales" of losses; the outliers we identify corroborate this point. Damian et al. [9] prove that GD implicitly regularizes the sharpness—we identify a conceptually distinct source of such regularization, as described in Section 2. 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. [57] 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. [33]. Our proposed explanation for the effect of opposing signals offers some insight into this relationship.

Roughly, our results seem to imply that progressive sharpening occurs when the network learns to rely on (or *not* rely on) opposing signals in a very specific way, while simultaneously amplifying overall sensitivity. This growth in sensitivity means a small parameter change modifying how opposing signals are used can massively increase loss. This leads to intermittent instability orthogonal to the "valley floor", accompanied by gradual training loss decay and occasional spikes as described by the toy example in Fig. 5 and depicted on real data in Fig. 1.

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 [59, 28, 22, 54]. "Grokking" [45], 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, 39, 35]; it has even been noted to co-occur with weight oscillation [40]. We observe that the opposing signal gradients dominate the overall gradient for most of training—and our experiments on SGD in Section 3 and **??** 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 [13]: we think it is not unlikely that a pruned winning ticket is one which is not as influenced by 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 and double descent. Nakkiran et al. [37] observed that NNs learn functions of increasing 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 more complex subnetworks with smaller magnitude and better performance to take over. At first this seems at odds with the notion of *simplicity bias* [55, 52], defined broadly as a tendency of networks to rely on simple functions of their inputs. However, it does seem to be the case that the network will use the simplest (e.g., largest norm) features that it can, so long as such features allow it to approach zero training loss; otherwise it may eventually diverge. This tendency also suggests a possible explanation for *double descent* [3, 38]: even after interpolation, the network pushes towards greater confidence and the weight layers continue to balance [51, 10], increasing sharpness. This could lead to oscillation, pushing the network to learn new features which generalize better [59, 30, 47, 54]. This behavior would also be more pronounced for larger networks because they exhibit greater sharpening. Note that the true explanation is not quite so straightforward: generalization is sometimes improved via methods that *reduce* oscillation (like loss smoothing), implying that this behavior is not always advantageous. A better understanding of these nuances is an important subject for future study.

Sharpness-Aware Minimization Another connection we think merits further inquiry is Sharpness-Aware Minimization (SAM) [11], which is known to improve generalization of neural networks for reasons still not fully understood [58]. In particular, the better-performing variant is 1-SAM, which takes positive gradient steps on each training point in the batch individually. It it evident that several of these updates will point 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" oscillation and divergence out of this valley in both directions, enabling exploration in a manner that would not normally be possible until the sharpness grows large enough—these intermediate steps would also encourage the network to downweight these features sooner and faster. In contrast, standard SAM would only take this step in one of the two directions, or perhaps not at all if the opposing signals are equally balanced. Furthermore, unlike 1-SAM the intermediate step would blend together all opposing signals in the minibatch. These possibilities seem a promising direction for further exploration.

B Understanding the Effect of Opposing Signals

Beyond noting their existence, 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 outliers cause the observed behavior. In this section we give a simplified "mental picture" which serves as our current understanding this process. We begin with an informal discussion of the outsized influence of opposing signals and how they lead to progressive sharpening; this subsection collates and expands on prior work to give important context for how these signals differ from typically imagined "noise". Next, we give a mechanistic description of the specific effect of opposing signals 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. Finally, we prove that this behavior occurs on a two-layer linear network under a simple model.

B.1 Progressive Sharpening, and Intuition for Why these Features are so Influential

At a high level, most variation in the input is unneeded when training a network to minimize predictive error—particularly with depth and high dimension, only a small fraction of information will be propagated to the last linear layer [17]. Starting from random initialization, training a network aligns adjacent layers' singular values [51, 36] to amplify meaningful signal while downweighting noise,[1] growing *sensitivity* to the important signal. This sensitivity can be measured, for example, by the spectral norm of the input-output Jacobian, which grows during training [31]; it has also been connected to growth in the norm of the output layer [57].

Observe that with this growth, small changes to *how the network processes inputs* become more influential. Hypothetically, a small weight perturbation could massively increase loss by redirecting unhelpful noise to the subspace to which the network is most sensitive, or by changing how the last layer uses it. The increase of this sensitivity thus represents precisely the growth of loss Hessian spectrum, with the strength of this effect increasing with depth [56, 10, 36].[2]

Crucially, this sharpening also depends on the structure of the input. If the noise is independent of the target, it will be downweighted throughout training. In contrast, *genuine signals which oppose each other* will be retained and perhaps even further amplified by gradient descent; this is because the "correct" feature may be much smaller in magnitude (or not yet learned), so using the large, "incorrect" feature is often the most immediate way of minimizing loss. As a concrete example, observe that a randomly initialized network will lack the features required for the subtle task of distinguishing birds from planes. But it *will* capture the presence of sky, which is very useful for reducing loss on such images by predicting the conditional $p(\text{class} \mid \text{sky})$ (this is akin to the "linear/shallow-first" behavior described by Nakkiran et al. [37], Mangalam and Prabhu [34]). Thus, any method attempting to minimize loss as fast as possible (e.g., steepest descent) may actually upweight these features. Furthermore, amplified opposing signals will cause greater sharpening than random noise, because using a signal to the benefit of one group is maximally harmful for the other—e.g., confidently predicting `plane` whenever there is sky will cause enormous loss on images of other classes with sky. Since random noise is more diffuse, this effect is less pronounced.

This description is somewhat abstract. To gain a more precise understanding, we illustrate the dynamics more explicitly on a toy example.

B.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—**among images with a sky background, we consider**

[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.

[2]The coincident growth of these two measures was previously noted by Ma and Ying [31], Gamba et al. [14], MacDonald et al. [33], though they did not make explicit this connection to how the network processes different types of input variance.

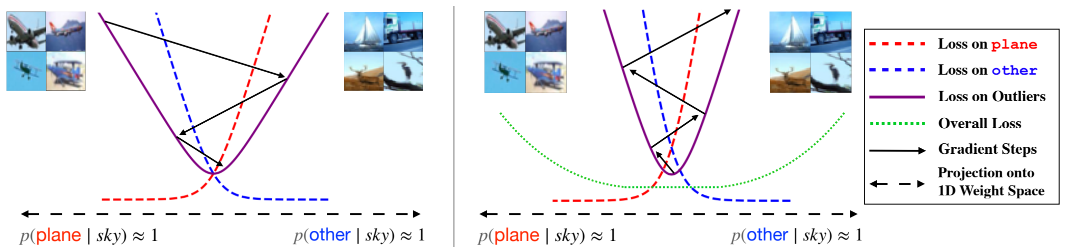


Figure 5: **A toy illustration of the effect of opposing signals.** Images with many blue pixels cause large activations, with high loss sensitivity. We project the loss to the hypothetical weight-space dimension "sky = `plane`". **Left:** Early optimization approaches the minimum, balancing the opposing gradients for `plane` and `other` (these are losses for *separate* training subsets: those labeled `plane` vs. those with any other label—the purple curve is their average). Progress continues through this valley, further growing the feature magnitude. **Right**: The valley sharpens and the iterates diverge, alternating between high and low loss for each group. Because most training points are insensitive to this axis, the overall loss may not be noticeably affected at first. Eventually either (a) the loss growth forces the network to downweight "sky", flattening the valley; or (b) the weights "catapult" to a different basin.

***separately* the loss on those labeled `plane` versus those with any other label.** Because the sky feature has large magnitude, a small change in weight space will induce a large change in the network outputs— i.e., a small movement in the direction "sky = `plane`" will greatly increase loss on these non-`plane` images.

Fig. 5 depicts this heavily simplified scenario. Early in training, optimizing this network with GD will rapidly move towards the minimum along this direction. In particular, until better features are learned, the direction of steepest descent will lead to a network which upweights the sky feature and predicts $p(\text{class} \mid \text{sky})$ whenever it occurs. Once sufficiently close to the minimum, the gradient will point "through the valley" towards amplifying the more relevant signal [61]. However, this will also cause the sky feature to grow in magnitude—as well as its *potential* influence were the weights to be selectively perturbed, as described above. Both these factors contribute to progressive sharpening.

Here we emphasize the distinction between the loss on the *outliers* and the full train loss. As images without sky are not nearly as sensitive to movement along this axis, their gradient and curvature is much smaller—and since they comprise the majority of the dataset, the global loss landscape may not at first be significantly affected. Continued optimization will oscillate across the minimum with growing magnitude, but this growth may not be immediately apparent. Furthermore, *progress orthogonal to these oscillations need not be affected*—we find some evidence that these two processes occur somewhat independently, which we present in Section 3. Returning to the loss decomposition, we see that 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 [59, 28, 54]. This phenomenon is depicted in Fig. 1. Finally, we note that if one visualizes the dynamics in Fig. 5 from above—so the left/right direction on the page becomes up/down—it gives exactly the pattern of a network's weights projected onto the top eigenvector of the Hessian (e.g., **??** later in this work).

Verifying our toy examples's predictions. Though this explanation lacks precise details, it does enable concrete predictions of network behavior during training. Fig. 6 tracks the predictions of a ResNet-18 on an image of sky—to eliminate possible confounders, we create a synthetic image as a single color block. Though the "`plane` vs. `other`" example seems almost *too* simple, 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, as evidenced by the slow decay in feature norm. We reproduce this figure for many other inputs and for a VGG-11-BN in Appendix D, with similar findings.

Our example also suggests that **oscillation serves as a valuable regularizer that reduces reliance on easily learned opposing signals which may not generalize.** When a signal is used to the benefit of one group and the detriment of another, the advantaged group's loss goes down while the other's

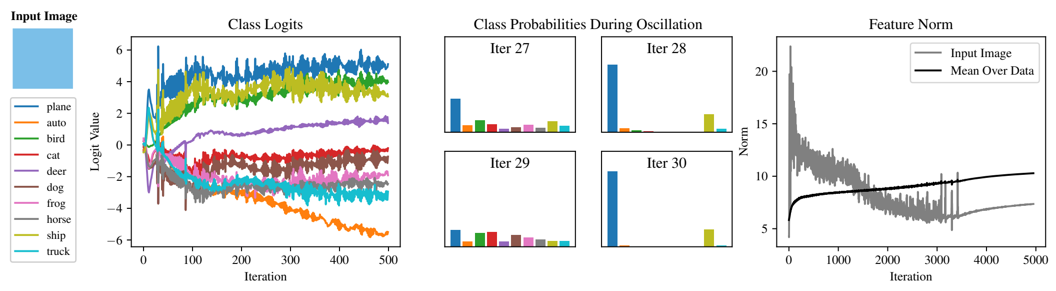


Figure 6: **Passing a sky-colored block through a ResNet during GD precisely tracks the predictions of our toy example. Left:** In the first phase, the network rapidly learns to use the sky feature to minimize loss. As signal is amplified, so too is the sky-colored input, and oscillation begins as depicted in Fig. 5. **Middle:** During oscillation, gradient steps alternate along the axis "sky $=$ `plane`" (and a bit `ship`). **Right:** The initial phase amplifies the sky input, causing rapid growth in feature norm. The network then oscillates, slowly learning to downweight this feature and rely on other signal (average feature norm provided for comparison).

goes up, meaning the latter's gradient grows in magnitude while the former's shrinks. As the now gradient-dominating group is also the one disadvantaged by the use of this signal, the network will be encouraged to downweight this feature. In Appendix D.3 we reproduce Fig. 6 with a VGG-11-BN trained with a very small learning rate to closely approximate gradient flow. We see that gradient flow and GD are very similar until reaching the edge of stability. After this point, the feature norm under GD begins to slowly decay while oscillating; in contrast, in the absence of oscillation, the feature norms of opposing signals under gradient flow grow continuously. If it is the case that opposing signals represent "simple" features which generalize worse, this could help to explain the poor generalization of gradient flow. A similar effect was observed by Jastrzębski et al. [21], who noted that large initial learning rate leads to a better-conditioned loss landscape later.

C 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:  [the] / other acts of love, both divine
and human:  [the] / integrate fighters from the
Kurds’ two main political parties:  [the] / of
precisely what makes it so wonderful:  [the]
```

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 precedes “the”, other times it announces the start of a new line. The model must *unlearn* “: $\mapsto$ `[\n]`” versus “: $\mapsto$ `[ ]`” and instead use other contextual information (possibly by diverging out of the valley).

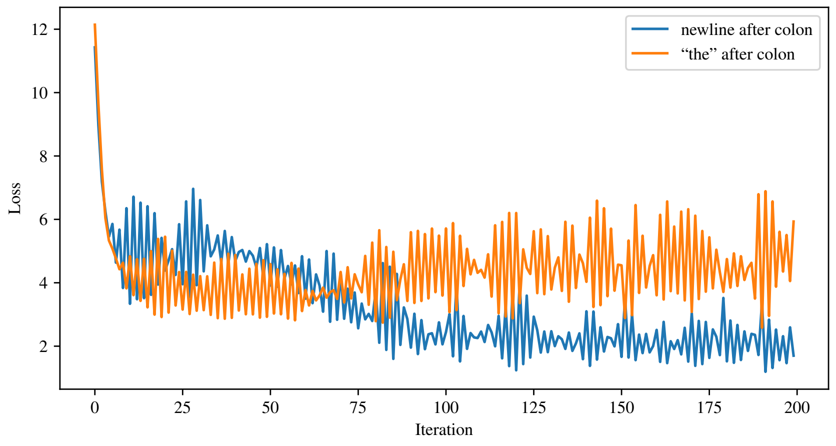


Figure 7: Loss of GPT-2 on the above opposing signals between ‘the’ and ‘\n’ after ‘:‘.

D Reproducing Fig. 6 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.

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

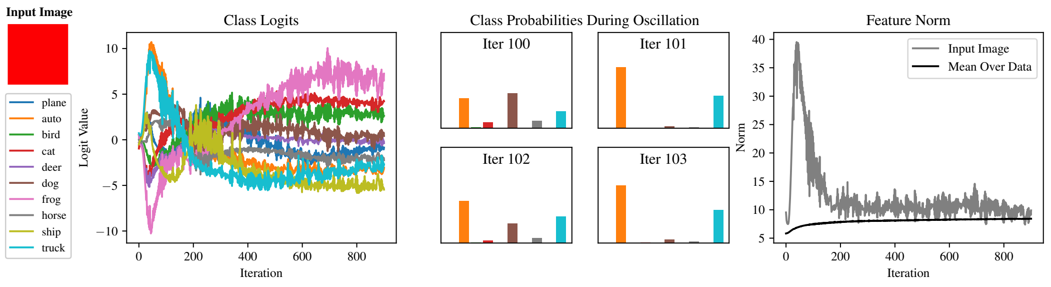


Figure 8: ResNet-18 on a red color block.

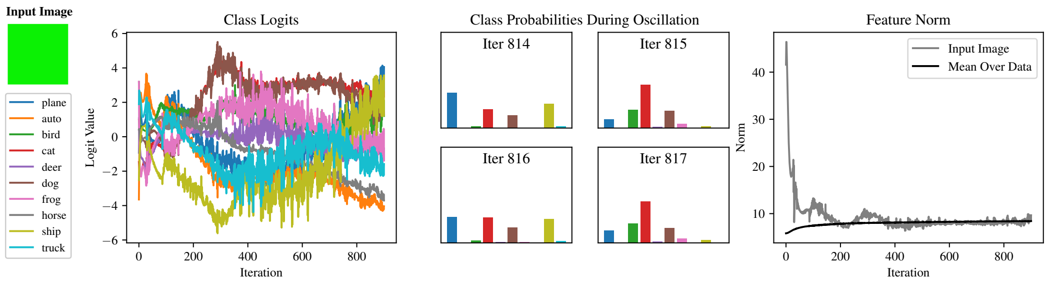


Figure 9: ResNet-18 on a green color block. As this color seems unnatural, we've included two examples of relevant images in the dataset.

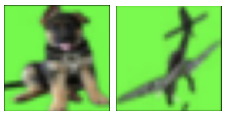


Figure 10: Examples of images with the above green color.

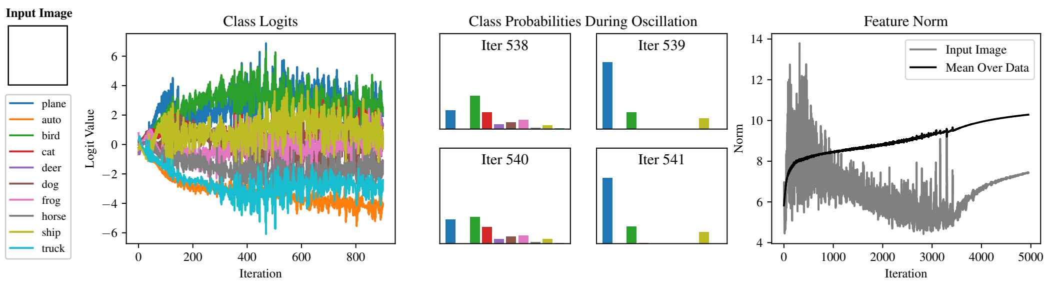


Figure 11: ResNet-18 on a white color block.

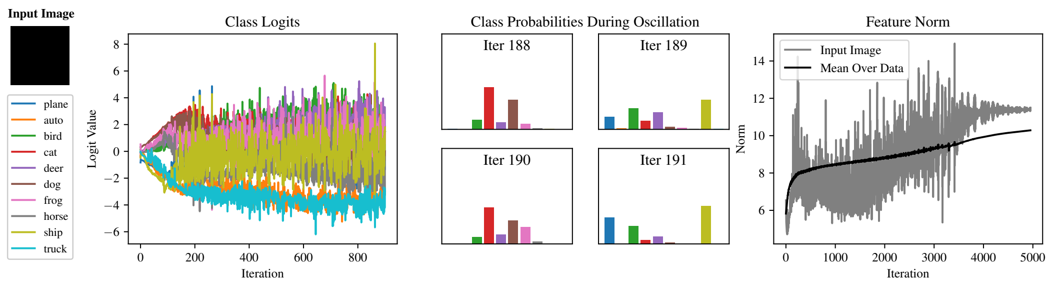


Figure 12: ResNet-18 on a black color block.

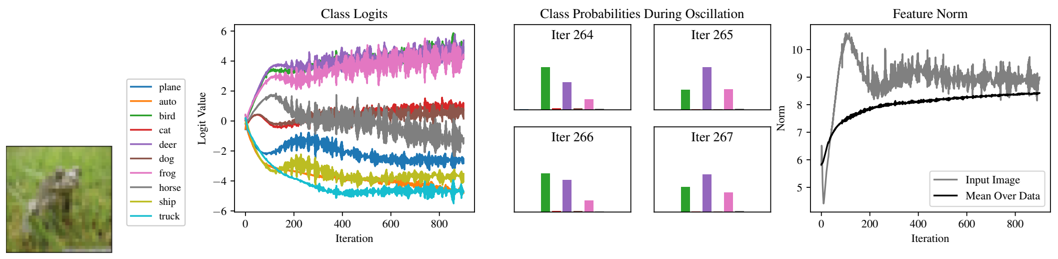


Figure 13: ResNet-18 on an image with mostly grass texture.

D.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.

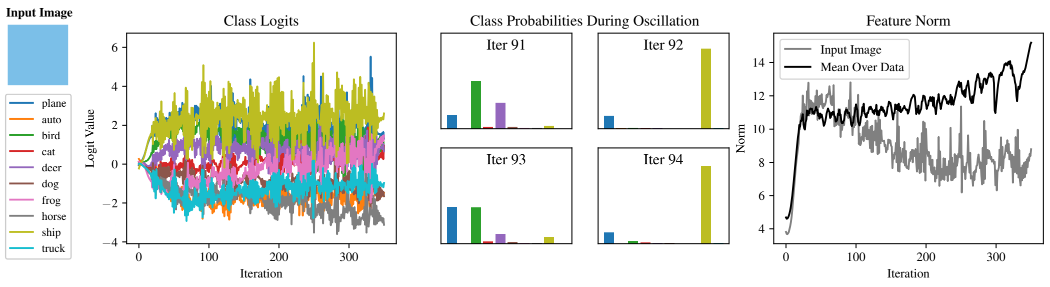


Figure 14: VGG-11-BN on a sky color block.

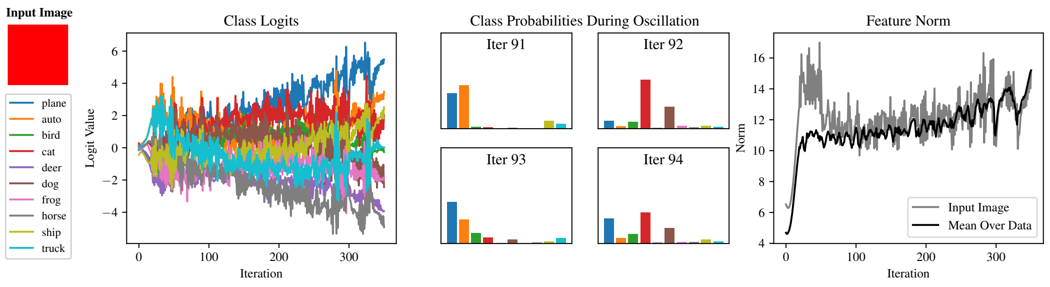


Figure 15: VGG-11-BN on a red color block.

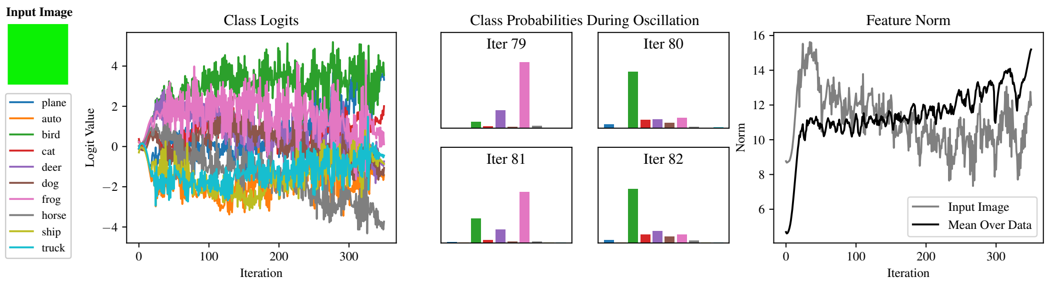


Figure 16: VGG-11-BN on a green color block. See above for two examples of relevant images in the dataset.

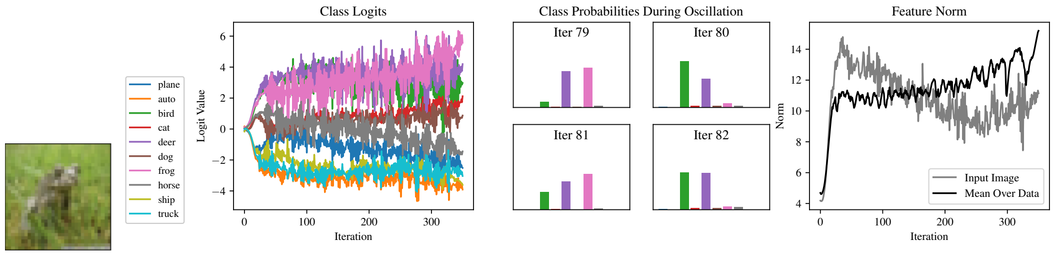


Figure 17: VGG-11-BN on an image with mostly grass texture.

D.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..

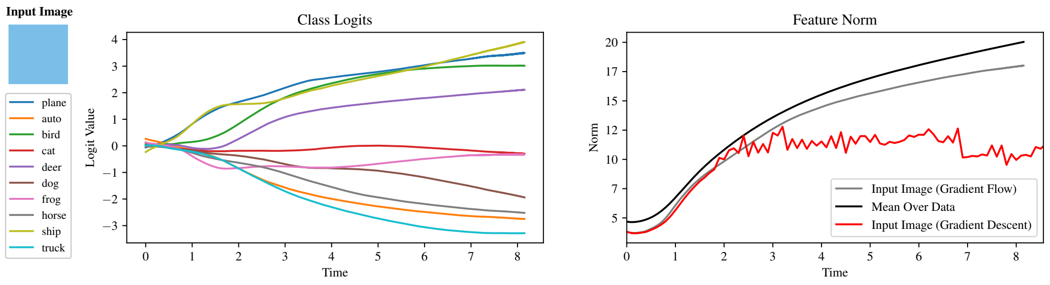


Figure 18: VGG-11-BN on a sky color block with learning rate 0.005 (approximating gradient flow) compared to 0.1.

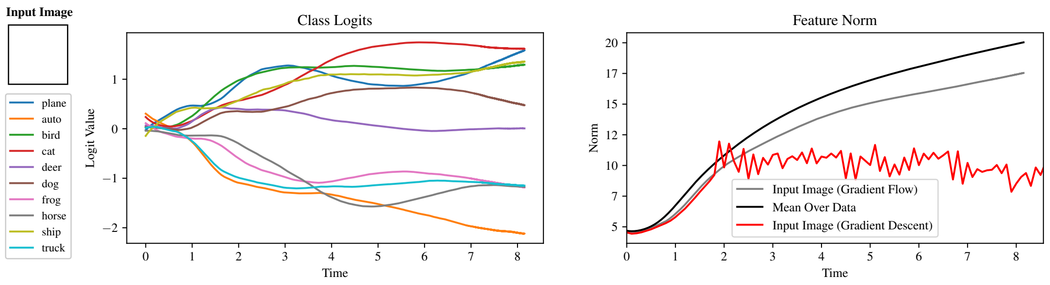


Figure 19: VGG-11-BN on a white color block with learning rate 0.005 (approximating gradient flow) compared to 0.1.

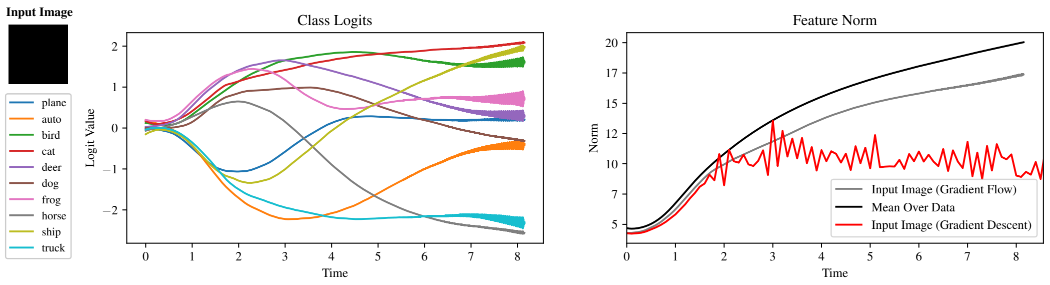


Figure 20: VGG-11-BN on a black color block with learning rate 0.005 (approximating gradient flow) compared to 0.1.

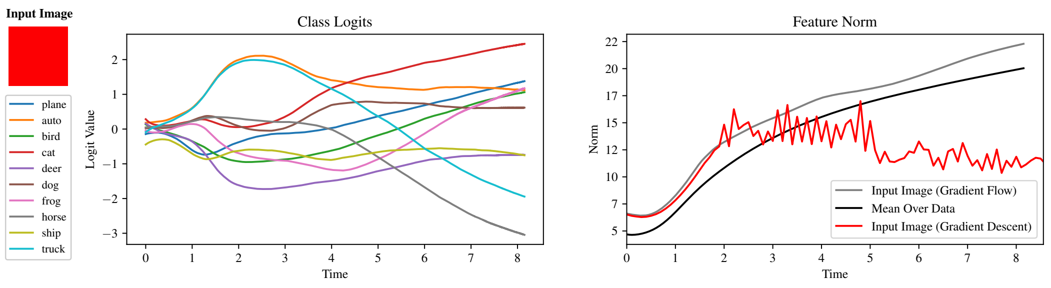


Figure 21: VGG-11-BN on a red color block with learning rate 0.005 (approximating gradient flow) compared to 0.1.

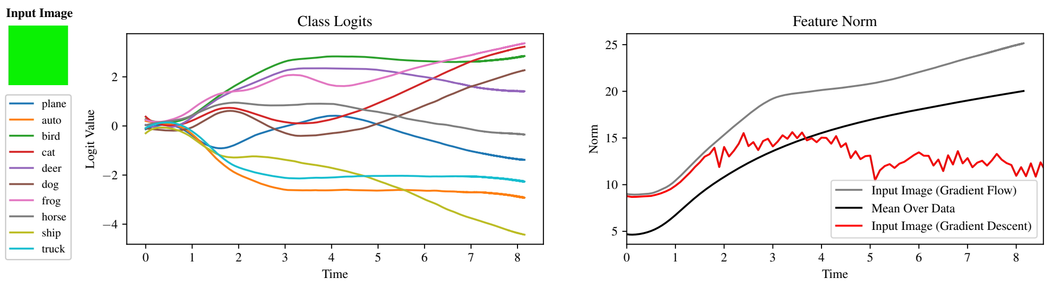


Figure 22: VGG-11-BN on a green color block with learning rate 0.0005 (approximating gradient flow) compared to 0.1.

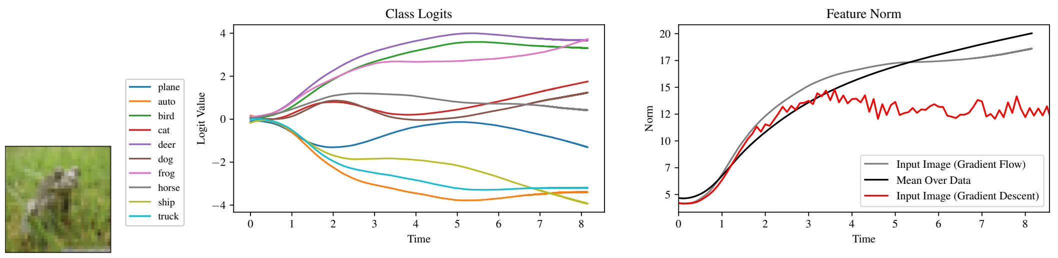


Figure 23: VGG-11-BN on an image with mostly grass texture with learning rate 0.0005 (approximating gradient flow) compared to 0.1.

D.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.

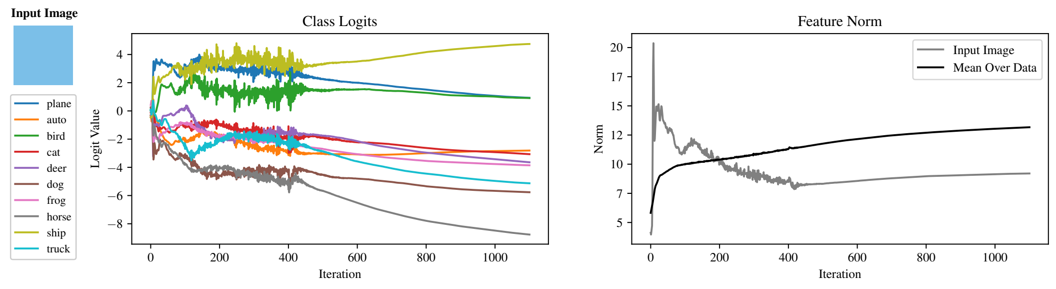


Figure 24: ResNet-18 on a sky color block trained with Adam.

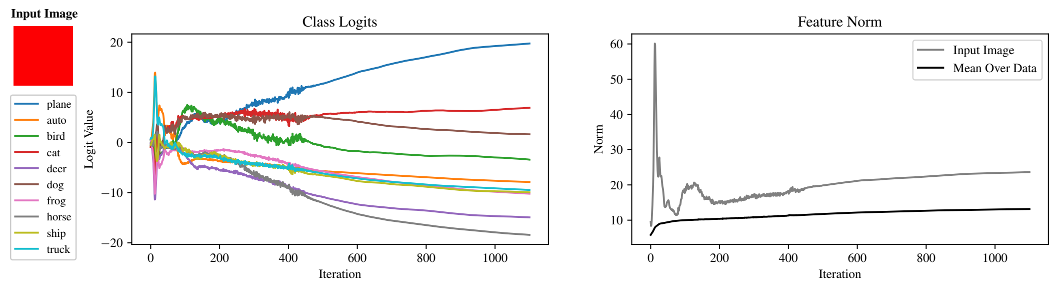


Figure 25: ResNet-18 on a red color block trained with Adam.

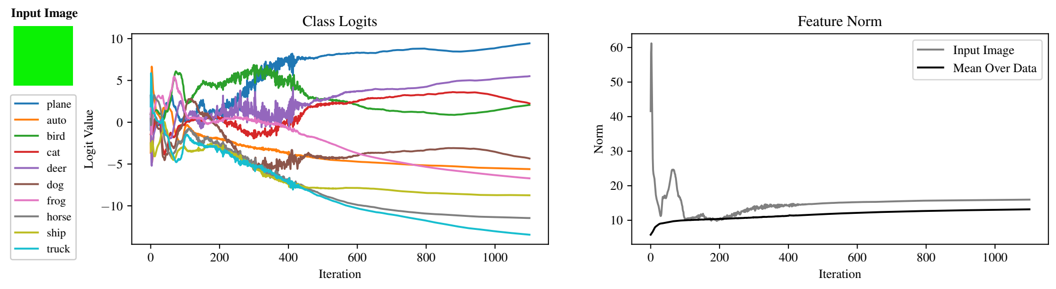


Figure 26: ResNet-18 on a green color block trained with Adam.

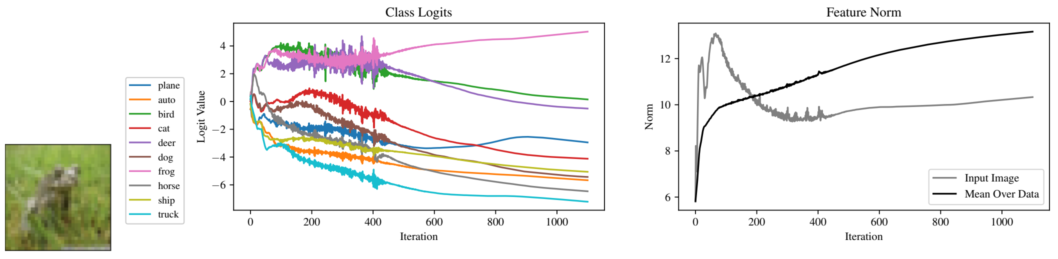


Figure 27: ResNet-18 on an image with mostly grass texture trained with Adam.

E Additional Experimental Findings

Sharpness often occurs overwhelmingly in the first few layers. Theorem J.2 shows that progressive sharpening occurs specifically in b_o. Generally, our model suggests that sharpness will begin in the last layer but that that early in training it will shift to the earlier layers since they have more capacity to redirect the signal. In Appendix F we track what fraction of curvature[3] lies in each layer of various networks during training with GD. In a ResNet-18, sharpness occurrs almost exclusively in the first convolutional layer after the first few training steps; the same pattern appears more slowly while training a VGG-11. In a Vision Transformer curvature occurs overwhelmingly in the embedding layer and very slightly in the earlier MLP projection heads. The text transformer (NanoGPT) follows the same pattern, though with less extreme concentration in the embedding. Thus it does seem to be the case that earlier layers have the most significant sharpness—especially if they perform dimensionality reduction or have particular influence over how the signal is propagated to later layers. This seems the likely cause of large gradients in the early layers of vision models [6, 26], suggesting that this effect is equally influential during finetuning and pretraining and that further study can improve optimization.

Batchnorm may smooth training, even if not the loss itself. Cohen et al. [7] noted that batchnorm (BN) [18] does not prevent networks from reaching the edge of stability and concluded, contrary to Santurkar et al. [50], that BN does not smooth the loss landscape. We conjecture that the benefit of BN may be in downweighting the influence of opposing signals and mitigating this oscillation. In other words, 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. [7] between regularity and smoothness). In Section 3 we demonstrate that Adam *also* smooths the optimization trajectory and that minor changes to emulate this effect can aid stochastic optimization. We imagine that the effect of BN could also depend on the use of GD vs. SGD. Specifically, our findings hint at a possible benefit of BN which applies only to SGD: reducing the influence of imbalanced opposing signals in a random minibatch.

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. 34 in the Appendix shows that the fraction of points which increase in loss hovers around 50% for every step—to some extent, a small degree of oscillation appears to be happening to the entire dataset.

Different losses have different effects on sharpening. Our model would predict that adding label smoothing to the cross-entropy loss should reduce sharpening, because smoothing reduces loss curvature under extreme overconfidence. Indeed, MacDonald et al. [33] show this to be the case. This also hints at why logistic loss may be more suitable for NN optimization, because it only has substantial curvature around $x = 0$ (x being the logit, i.e. when prediction entropy is high), so unlike square or exponential loss, large magnitude features will not massively increase sharpness. We expect a similar property could contribute to the relative behavior of different activations (e.g. ReLU or tanh).

[3]The "fraction of curvature" is with respect to the top eigenvector of the loss Hessian. We partition this vector by network layer, so each sub-vector's squared norm represents that layer's contribution to the overall curvature.

F Tracking the Amount of Curvature in each Parameter Layer

Here we plot the “fraction of curvature” of different architectures at various training steps. Recall the fraction of curvature is defined with respect to the top eigenvector of the loss Hessian. We partition this vector by network layer and evaluate each sub-vector’s squared norm. This represents that layer’s contribution to the overall curvature. To keep the plots readable, we omit layers whose fraction is never greater than 0.01 at any training step (including the intermediate ones not plotted), though we always include the last linear layer. The total number of layers is 45, 38, 106, and 39 for the ResNet, VGG-11, ViT, and NanoGPT respectively.

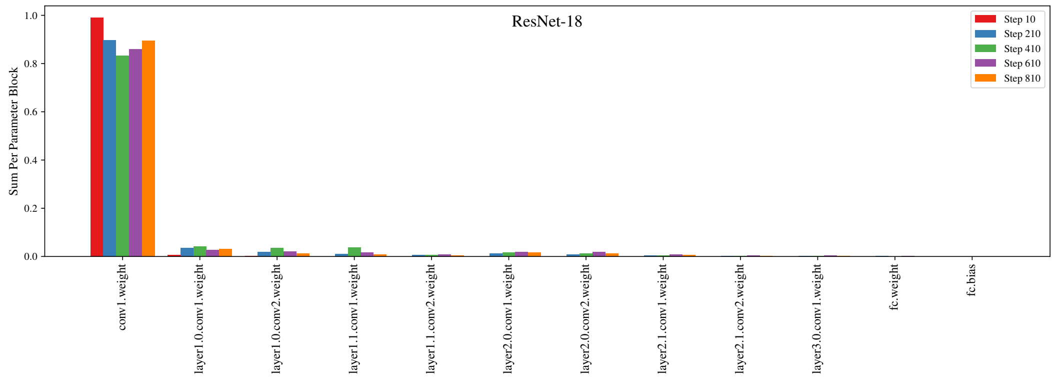


Figure 28: Sum of squared entries of the top eigenvector of the loss Hessian which lie in each parameter layer of a ResNet-18 throughout training.

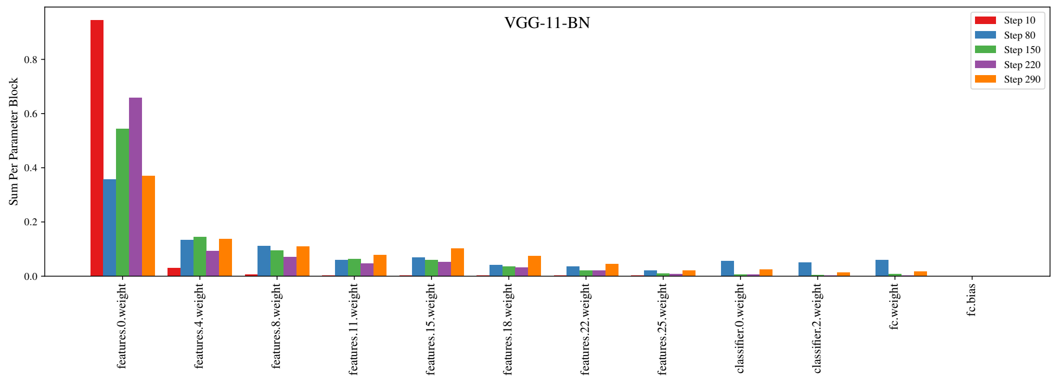


Figure 29: Sum of squared entries of the top eigenvector of the loss Hessian which lie in each parameter layer of a VGG-11-BN throughout training.

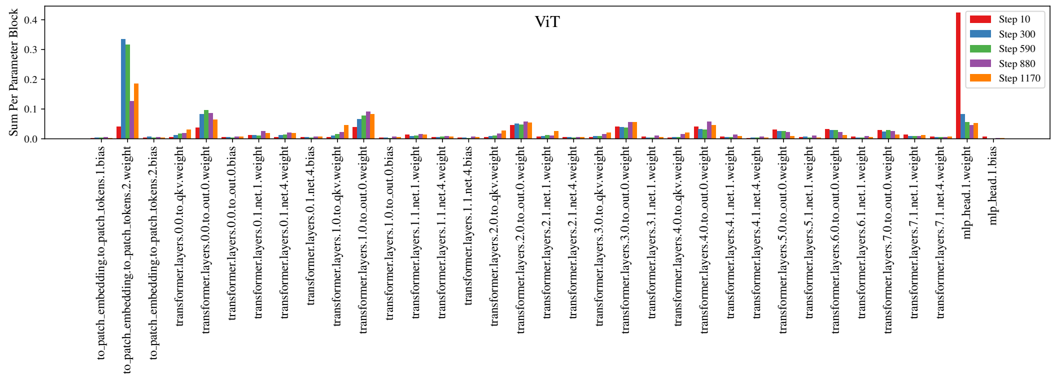


Figure 30: Sum of squared entries of the top eigenvector of the loss Hessian which lie in each parameter layer of a ViT throughout training.

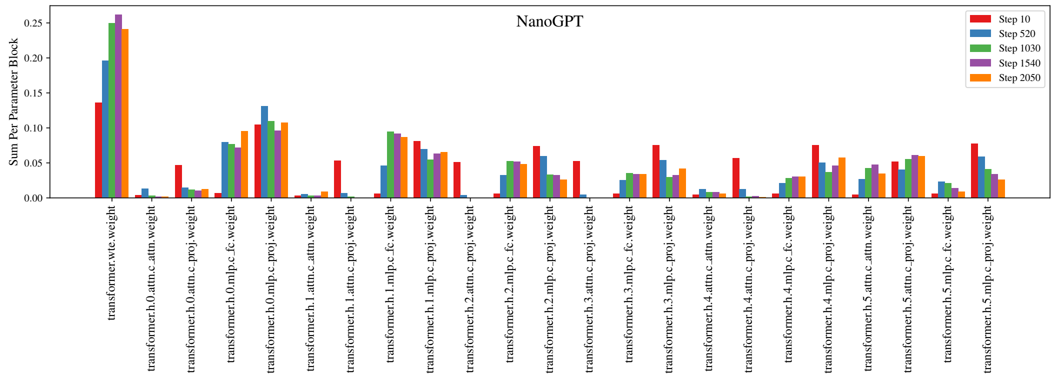


Figure 31: NanoGPT (6 layers, 6 head per layer, embedding dimension 384) trained on the default shakespeare character dataset in the NanoGPT repository. Due to difficulty calculating the true top eigenvector, we approximate it with the exponential moving average of the squared gradient.

G Additional Figures

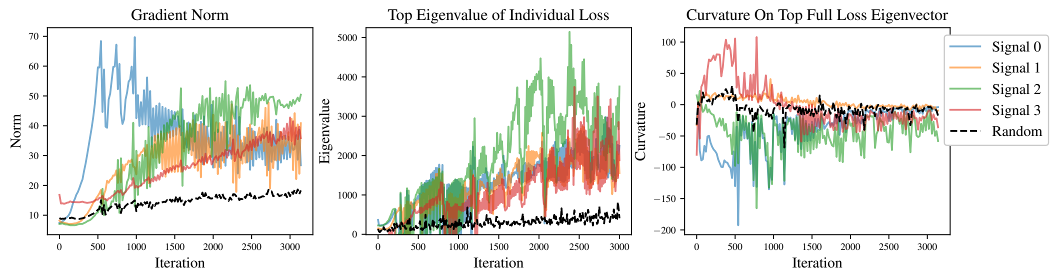


Figure 32: **Tracking other metrics which characterize outliers with opposing signals.** For computational considerations, we select outliers according to maximal per-step change in loss. However, this quantity relates to other useful metrics, such as gradient norm and curvature. We see that the samples we uncover are also significant outliers according to these metrics.

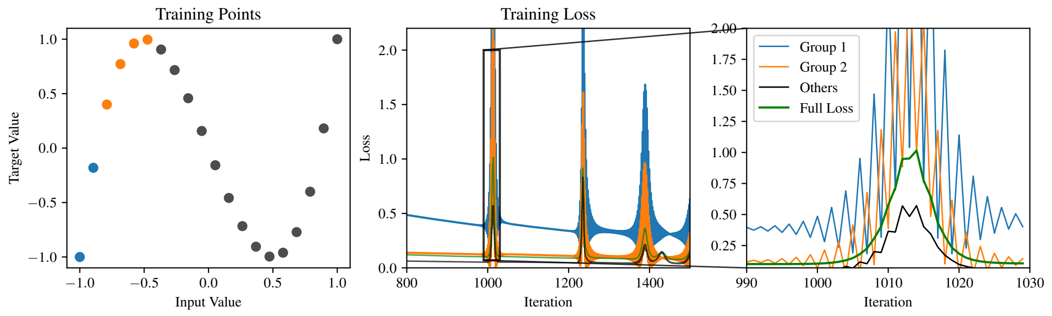


Figure 33: **Opposing signals when fitting a Chebyshev polynomial with a small MLP.** Though the data lacks traditional "outliers", it is apparent that the network has some features which are most influential only on the more negative inputs (or whose effect is otherwise cancelled out by other features). Since the correct use of this feature is opposite for these two groups, they provide opposing signals.

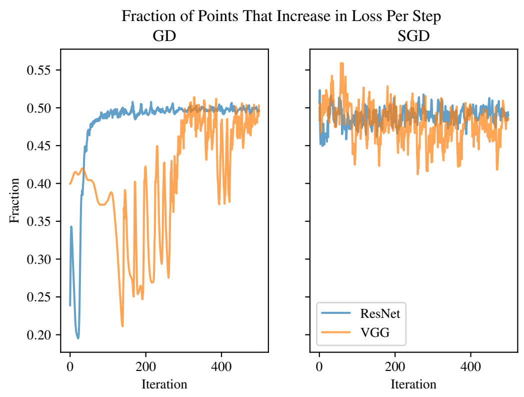


Figure 34: The fraction of overall training points which increase in loss on any given step. For both SGD and GD, it hovers around 0.5 (VGG without batchnorm takes a long time to reach the edge of stability). Though only the outliers are wildly swinging in loss, many more images contain *some* small component of the features they exemplify, and so these points also oscillate in loss at a smaller scale.

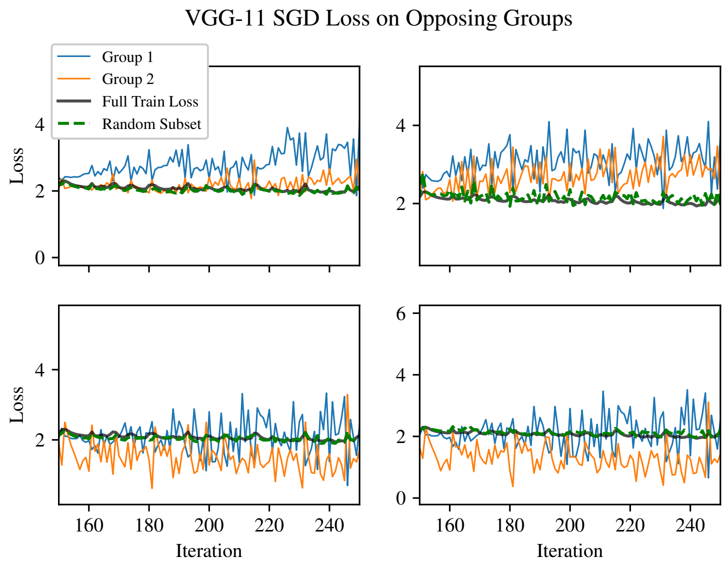


Figure 35: We reproduce Fig. 4 without batch normalization.

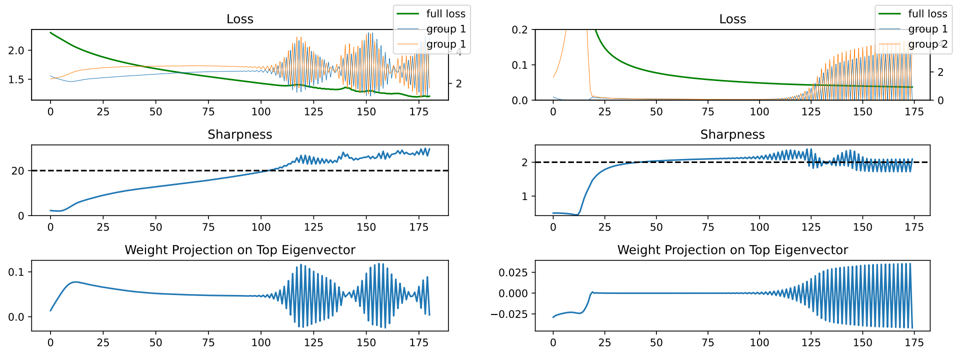


Figure 36: A 3-layer ReLU MLP trained on a 5k-subset of CIFAR-10 compared to our model: a 2-layer linear network trained on mostly Gaussian data with opposing signals.

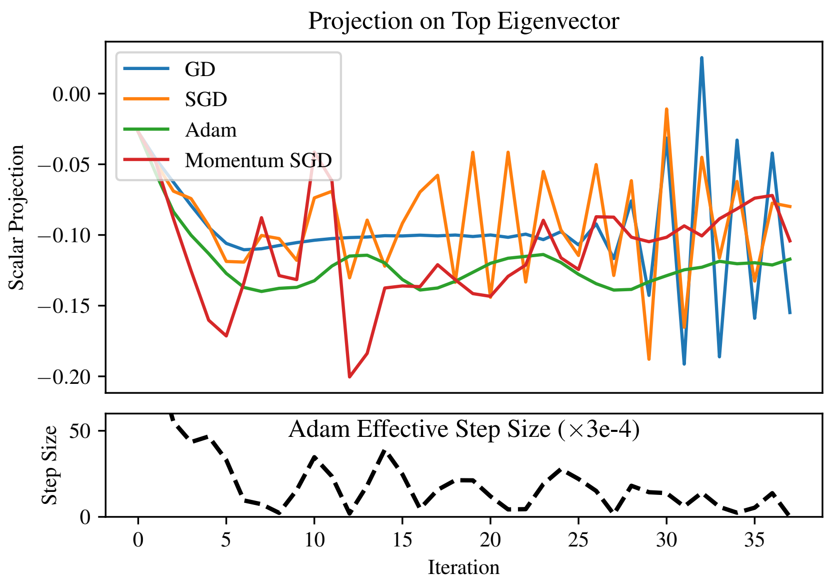


Figure 37: **Projected iterates of an MLP on CIFAR-10. Top:** SGD closely tracks GD, bouncing across the valley; momentum somewhat mitigates the sharp jumps. Adam smoothly oscillates along one side. **Bottom:** Adam's effective step size drops sharply when moving too close or far from the valley floor.

H Discussion on the Advantages of Adam

To better understand their differences, Fig. 37 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 (all methods use the same initialization and sequence of training batches). 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. We identify three components of Adam which potentially contribute to this effect:

Advantage 1: Smaller steps along high curvature directions. Adam's normalization causes smaller steps along the top eigenvector, especially near the minimum. The lower plot in Fig. 37 shows that the effective step size in this direction—i.e., the absolute inner product of the parameter-wise step sizes and the top eigenvector—rapidly drops to zero as the iterates approach the valley floor (in the opposite direction, the gradient negates the momentum for the same effect). We conjecture that general normalization may not be essential to Adam's performance; we even expect it could be somewhat harmful by limiting exploration. On the other hand, normalizing steps by curvature *parameter-wise* does seem important; Pan and Li [41] argue the same and show that parameter-wise gradient clipping improves SGD substantially. We highlight why this may be useful in the next point.

Advantage 2: Managing heavy-tailed gradients and avoiding steepest descent. Zhang et al. [62] identified the "trust region" as an important contributor to Adam's success in attention models, pointing to heavy-tailed noise in the stochastic gradients. More recently, Kunstner et al. [27] argued that Adam's superiority does not come from better handling noise, which they supported by experimenting with large batch sizes. Our result reconciles these contradictory claims by showing that **the difficulty is not heavy-tailed *noise*, but strong, directed (and perhaps imbalanced) opposing signals.** Unlike traditional "gradient noise", larger batch sizes may not reduce the effect of these signals—that is, the gradient is heavy-tailed (across parameters) even without being stochastic. Furthermore, the largest steps emulate Sign SGD, which is notably *not* a descent method. Fig. 37 shows that Adam's steps are more parallel to the valley floor than those of steepest descent. **Thus it seems advantageous to *intentionally* avoid steps along the gradient which point towards the**

local minimum, which might lead to over-reliance on these features. Indeed, Benzing [4] observe that true second order methods perform worse than SGD on NNs, and Kunstner et al. [27] show that Adam shares some behavior with Sign SGD with momentum. This point is also consistent with the observed generalization benefits of a large learning rate for SGD on NNs [21]; in fact, opposing signals naturally fit the concept of "easy-to-generalize" features as modeled by Li et al. [30].

Advantage 3: Dampening. Lastly, Adam's third important factor: downweighting 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 stochastic 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 [62, 41, 5].

H.1 Proof of Concept: Using These Insights to Aid Stochastic Optimization

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 parameter is above this threshold, we take a fixed step size; otherwise, we take a gradient step as normal. The exact method appears in Appendix I. **We emphasize that our goal here is not to propose a new optimization algorithm.** Instead, we are exploring the potential value gained from knowledge of the existence and influence of opposing signals.

Results in Appendix I show that this approach matches Adam when training ResNet-56/110 on CIFAR-10 with learning rates for the unthresholded parameters across several orders of magnitude ranging from 10^{-4} to 10^{3}. Notably, the fraction of parameters above the threshold (whose step size is fixed) is only around 10-25% per step. This implies that the trajectory and behavior of the network is dominated by this small fraction of parameters; the remainder can be optimized much more robustly, but their effect on the network's behavior is obscured. We therefore see the influence of opposing signals as a possible explanation for the "hidden" progress in grokking [2, 39]. We also compare this method to Adam for the early phase of training GPT-2 [46] on the OpenWebText dataset [16]—not only do they perform the same, their loss similarity suggests that their exact trajectory may be very similar (Appendix I.2). Here the fraction of parameters above the threshold hovers around 50% initially and then gradually decays. The fact that many more parameters in the attention model are above the threshold suggests that the attention mechanism is more sensitive to opposing signals and that further investigation of how to mitigate this instability may be fruitful.

I Comparing our Variant of SGD to Adam

Algorithm 1 SplitSGD

```
input: Initial parameters $\theta_0$, SGD step size $\eta_1$, SignSGD step size $\eta_2$, momentum $\beta$, dampening $\tau$, threshold $r$.
initialize: $m_0 = \mathbf{0}$.
for $t \leftarrow 1, \dots, T$ do
    $g_t \leftarrow \nabla_\theta L_t(\theta_{t-1})$                                          ▷ Get stochastic gradient
    $m_t \leftarrow \beta m_{t-1} + (1-\tau) g_t$                                ▷ Update momentum with dampening
    $\hat{m}_t \leftarrow m_t / (1 - \tau^t)$                                                  ▷ Debias
    $v_{\text{mask}} \leftarrow \mathbf{1}\{|\hat{m}_t| \leq r\}$                                ▷ Split parameters by threshold
    $\theta_t \leftarrow \theta_{t-1} - \eta_1(\hat{m}_t \odot v_{\text{mask}}) - \eta_2(\text{sign}(\hat{m}_t) \odot (1 - v_{\text{mask}}))$   ▷ unmasked SGD, masked SignSGD
end for
```

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 first gradient worked quite well—this was about `1e-4` for the ResNet-56/110 and `1e-6` for GPT-2.

Simply to have something to label it with, we name the method SplitSGD, because it performs SGD and SignSGD on different partitions of the parameters. The precise method is given above in Algorithm 1. We emphasize that we do not claim that this algorithm achieves superior performance—our goal is only to demonstrate the insight gained from knowledge of opposing signals' influence on NN optimization. For all plots, β represents the momentum parameter and τ is dampening. Adam has a single parameter β_1 which represents both simultaneously, which we fix at 0.9, and we do the same for SplitSGD. As in Algorithm 1, we let η_1 refer to the learning rate for standard SGD on the parameters with gradient below the magnitude threshold, and η_2 to the learning rate for the remainder which are optimized with SignSGD.

I.1 SplitSGD on ResNet

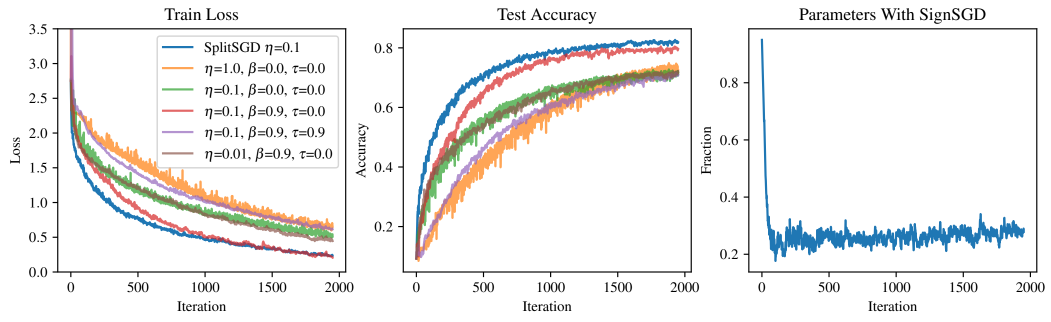


Figure 38: Standard SGD with varying learning rates and momentum/dampening parameters on a ResNet-56 on CIFAR-10, with one run of SplitSGD for comparison. Omitted SGD hyperparameter combinations performed much worse. Notice that SGD is extremely sensitive to hyperparameters. Rightmost plot is the fraction of parameters with fixed step size by SplitSGD.

We begin with a comparison on ResNets trained on CIFAR-10. Fig. 38 compares SplitSGD to standard versions of SGD with varying momentum and dampening on a ResNet-56. As expected, SGD is extremely sensitive to hyperparameters, particularly the learning rate, and even the best choice in a grid search underperforms SplitSGD. Furthermore, the rightmost plot depicts the fraction of parameters for which SplitSGD takes a fixed-size signed step. This means that after the first few training steps, 70-80% of the parameters are being optimized simply with standard SGD (with $\beta = \tau = 0.9$).

Next, Fig. 39 plots SplitSGD with varying η_1 and η_2 fixed at .001. This is compared to Adam with learning rate .005, which was chosen via oracle grid search. Even though the SGD learning rate

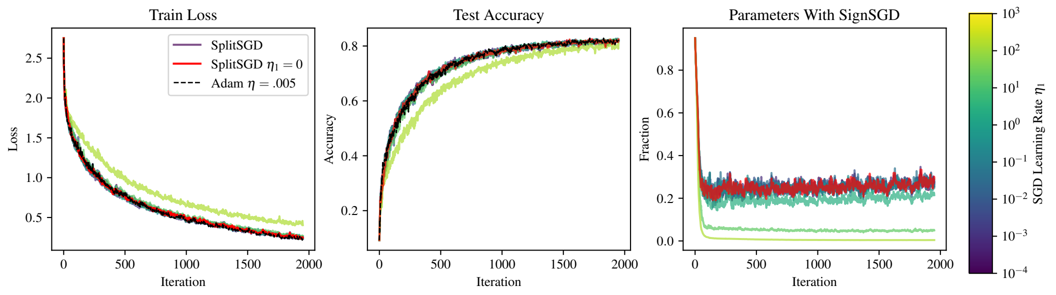


Figure 39: SplitSGD with varying SGD learning rates η_1 versus Adam on a ResNet-56 on CIFAR-10. The SignSGD learning rate is fixed at $\eta_2 = .001$; Adam uses $\eta = .005$, which was found to be the best performing choice via oracle selection grid search. The rightmost plot is the fraction of parameters with fixed step size by SplitSGD—that is, 1 minus this value is the fraction of parameters taking a regular gradient step with step size as given in the legend. This learning rate ranges over several orders of magnitude, is used for ~70-80% of parameters, and can even be set to 0, with no discernible difference in performance.

η_1 ranges over *seven orders of magnitude* and is used for ~70-80% of parameters, we see no real difference in the train loss or test accuracy of SplitSGD. In fact, we find that we can even eliminate it completely! This suggests that for most of parameters and most of training, it is only a small fraction of parameters in the entire network which are influencing the overall performance. We posit a deeper connection here to the "hidden" progress described in grokking [2, 39]—if the correct subnetwork and its influence on the output grows slowly during training, that behavior will not be noticeable until the dominating signals are first downweighted.

Figs. 40 and 41 depict the train loss and test accuracy of Adam and SplitSGD for varying learning rates (the standard SGD learning rate η_1 is fixed at 0.1). We see that SplitSGD is at least as robust as Adam to learning rate choice, if not more. The results also suggest that SplitSGD benefits from a slightly smaller learning rate than Adam, which we attribute to the fact that it will *always* take step sizes of that fixed size, whereas the learning rate for Adam represents an upper bound on the step size for each parameter.

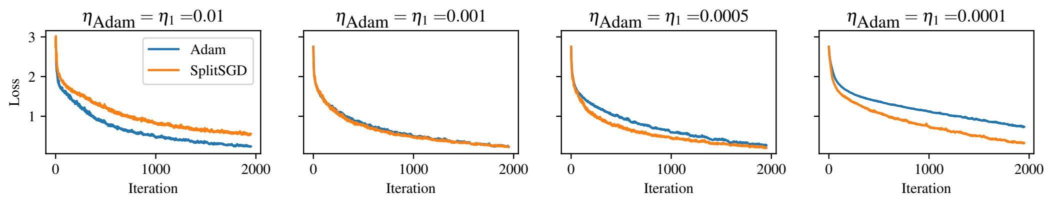


Figure 40: Train loss of Adam and SplitSGD for varying learning rates. The regular SGD step size for SplitSGD is fixed at 0.1. SplitSGD seems at least as robust to choice of learning rate as Adam, and it appears to benefit from a slightly smaller learning rate because it cannot adjust per-parameter.

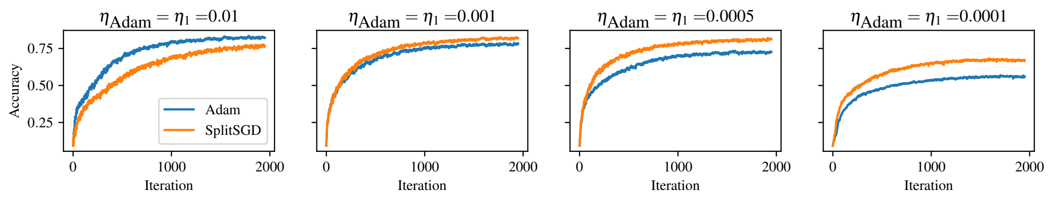


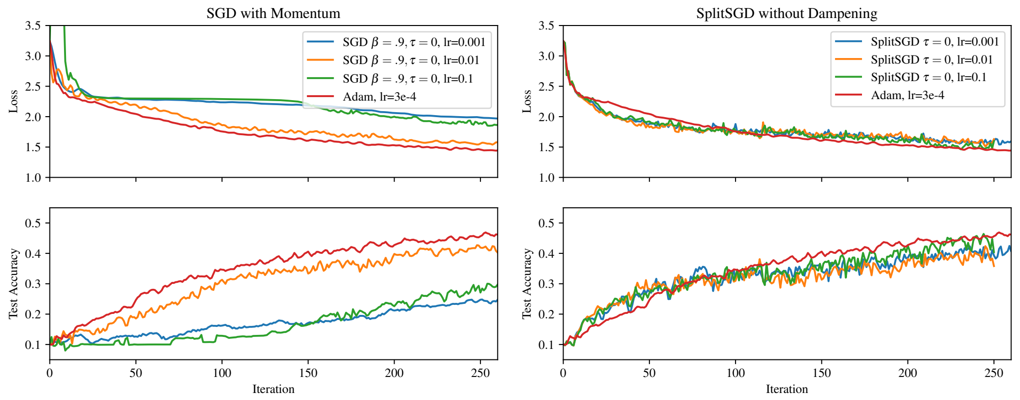
Figure 41: Test accuracy of Adam and SplitSGD for varying learning rates. The regular SGD step size for SplitSGD is fixed at 0.1. SplitSGD seems at least as robust to choice of learning rate as Adam, and it appears to benefit from a slightly smaller learning rate because it cannot adjust per-parameter.

We repeat these experiments with a ResNet-110, with similar findings. Fig. 42(a) compares the train loss and test accuracy of SGD with $\beta = 0.9, \tau = 0$ to Adam, and again the sensitivity of this optimizer to learning rate is clear. Fig. 42(b) compares Adam to SplitSGD (both with fixed-step learning rate .0003) but ablates the use of dampening: we find that the fixed-size signed steps appear to be more important for early in training, while dampening is helpful for maintaining performance later. It is not immediately clear what causes this bifurcation, nor if it will necessarily transfer to attention models.

Finally, Fig. 43(a) compares Adam to the full version of SplitSGD; we see essentially the same performance, and furthermore SplitSGD maintains its robustness to the choice of standard SGD learning rate.

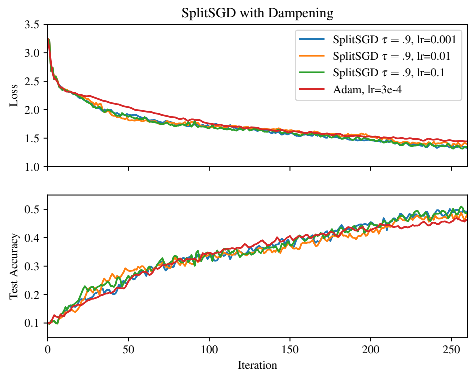
I.2 SplitSGD on GPT-2

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. All hyperparameters are the defaults from that repository, with the SGD learning rate η_1 set equal to the other learning rate η_2. We observe that not only do the two methods track each other closely in training loss, it appears that they experience *exactly* the same oscillations. Though we do not track the parameters themselves, this suggests that these two methods follow very similar optimization trajectories as well, which we believe is an intriguing possibility worth further study.

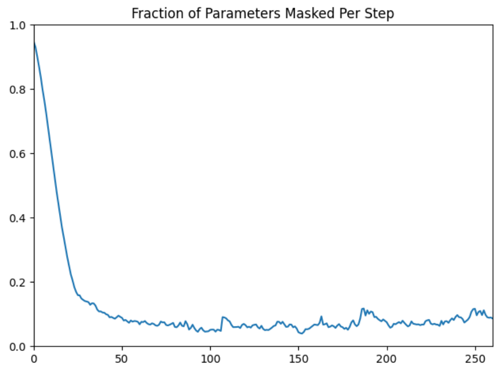


(a) Adam versus standard SGD with Momentum. SGD remains extremely sensitive to choice of learning rate.

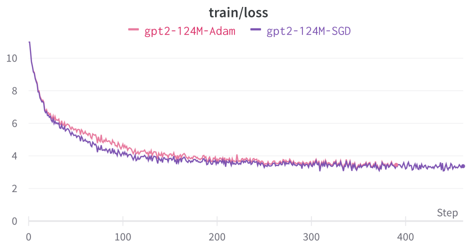
(b) Adam vs. SplitSGD with $\tau = 0$. Fixed-size learning rate for both is .0003.



(a) Adam vs. SplitSGD with $\tau = 0.9$. Fixed-size learning rate for both is .0003.



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



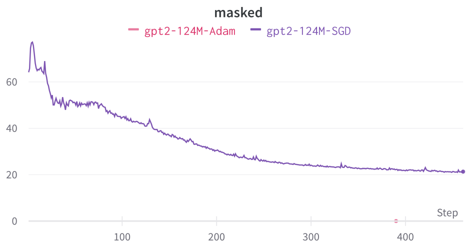


Figure 44: Adam versus SplitSGD on the initial stage of training GPT-2 on the OpenWebText dataset, and the fraction of parameters with a fixed-size signed step. All hyperparameters are the defaults from the nanoGPT repository. Observe that not only is their performance similar, they appear to have *exactly* the same loss oscillations.

J Presentation of Theoretical Results

To demonstrate this effect formally, we study misspecified linear regression on inputs $x \in \mathbb{R}^d$ with a two-layer linear network. Though this model is simplified, it enables preliminary insight into the factors we think are most important for these dynamics to occur. Since we are analyzing the dynamics from initialization until the stability threshold, it will be sufficient to study the trajectory of *gradient flow*—for reasonable step sizes η, a similar result then follows for gradient descent. Our analysis reproduces the initial phase of quickly reducing loss on the outliers, followed by the subsequent growth in sensitivity to the *way* the opposing signal is used—i.e., progressive sharpening. We also verify this pattern (and the subsequent oscillation, which we do not formally prove) in experiments on real and synthetic data in Appendix G.

We remark that one relevant factor which our model lacks is the concept of a "partially useful" signal as described at the end of Appendix B.1. This seems to require a somewhat more complex model to properly capture (e.g., multinomial logistic regression) so we view this analysis as an early investigation, capturing only part of relevant aspects of the phenomena we observe.

Model. 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 $d_2 \geq d_1$. 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 our experiments and this analysis, it seems that depth and a small signal-to-noise ratio are the only elements needed for this behavior to arise.

Setup. 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 the quantities of interest. As tracking the precise concentration terms would not meaningfully contribute to the analysis, we simplify by directly assuming that at initialization these quantities are equal to their expected order of magnitude: $\|b\|_2^2 = \mathbf{1}^\top b = \frac{d_1}{d_1+d_2}$, $\|b_o\|_2^2 = \mathbf{1}^\top b_o = \frac{d_2}{d_1+d_2}$, 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 gradient flow will elucidate how depth and large noise lead to sharpening as this minimum is approached.

Results. In exploring progressive sharpening, Cohen et al. [7] found that sometimes the model would have a brief *decrease* in sharpness, particularly for square loss. In fact, this is consistent with our above explanation: for larger α and a sharper loss (e.g. the square loss), the network will initially prioritize minimizing loss on the outliers, thus heavily reducing sharpness. Our first result proves that this occurs in the presence of large magnitude opposing signals:

Theorem J.1 (Initial *decrease* in sharpness). *Let $k := \frac{d_2}{d_1} \geq 1$, and assume $\|\beta\| > \max\left(\frac{d_1}{\sqrt{d_1+d_2}}, \frac{24}{5}\right)$. At initialization, the sharpness $\|\nabla_\theta^2 L(\theta)\|_2$ lies in $(\alpha, 3\alpha)$. Further, if $\sqrt{\alpha} = \Omega(\|\beta\| k \ln k)$, then both b_o and the overall sharpness will* decrease *as $\tilde{O}(e^{-\alpha t})$ from $t = 0$ until some time $t_1 \leq \frac{\ln \|\beta\|/2}{2\|\beta\|}$.*

Proofs can be found in Appendix K. After this decrease, signal amplification can proceed—but this also means that the sharpness with respect to *how the network uses the feature x_o* will grow, so a small perturbation to the parameters b_o will induce a large increase in loss.

Theorem J.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 $t_2 \geq \frac{1}{2\|\beta\|_2^2}$, 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 $\eta > \frac{16}{5\|\beta\|\alpha}$, b_o will start to increase in magnitude while oscillating across the origin. If this growth continues, it will rapidly *reintroduce* the feature, causing the loss on the outliers to grow and alternate. Such reintroduction (an example of which occurs around iteration 3000 in Fig. 6) seems potentially helpful for exploration. In **??** we simulate our model and verify exactly this sequence of events. We also show that an MLP trained on CIFAR-10 displays the same characteristic behavior.

K 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:

$$\begin{aligned} 2L(\theta) &= \mathbb{E}[(c(b^\top x + b_o^\top x_o) - (\beta^\top x + d_2^{-1/2}\mathbf{1}^\top |x_o|))^2] && (1)\\ &= \mathbb{E}[((cb-\beta)^\top x)^2] + \mathbb{E}[((cb_o - d_2^{-1/2}\operatorname{sign}(x_o)\mathbf{1})^\top x_o)^2] && (2)\\ &= \|cb-\beta\|^2 + \frac{p}{2}\left(\left(\sqrt{\frac{\alpha}{p}}(cb_o - 1)\right)^2 + \left(\sqrt{\frac{\alpha}{p}}(cb_o+1)\right)^2\right) && (3)\\ &= \|cb-\beta\|^2 + \alpha(c^2\|b_o\|^2 + 1). && (4)\end{aligned}$$

This provides the gradients

$$\begin{aligned} \nabla_b L &= c(cb-\beta), && (5)\\ \nabla_{b_o} L &= \alpha c^2 b_o, && (6)\\ \nabla_c L &= b^\top(cb-\beta) + \alpha\|b_o\|^2 c. && (7)\end{aligned}$$

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

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

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

$$\begin{aligned} \nabla_\epsilon L &= (\nabla_\epsilon b)^\top(\nabla_b L) && (9)\\ &= \frac{\beta}{\|\beta\|}^\top\left(c^2\left(\epsilon\frac{\beta}{\|\beta\|} + \delta v\right) - c\beta\right) && (10)\\ &= c^2\epsilon - c\|\beta\|, && (11)\\ \nabla_\delta L &= (\nabla_\delta b)^\top(\nabla_b L) && (12)\\ &= v^\top\left(c^2\left(\epsilon\frac{\beta}{\|\beta\|} + \delta v\right) - c\beta\right) && (13)\\ &= c^2\delta, && (14)\\ \nabla_c L &= \left(\epsilon\frac{\beta}{\|\beta\|} + \delta v\right)^\top\left(c\left(\epsilon\frac{\beta}{\|\beta\|} + \delta v\right) - \beta\right) + \alpha\|b_o\|^2 c && (15)\\ &= c(\epsilon^2 + \delta^2 + \alpha\|b_o\|^2) - \epsilon\|\beta\|. && (16)\end{aligned}$$

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:

$$\begin{aligned} \frac{d\epsilon}{dt} &= -c^2\epsilon + c\|\beta\|, && (17)\\ \frac{d\delta}{dt} &= -c^2\delta, && (18)\\ \frac{do}{dt} &= -2\alpha c^2 o, && (19)\\ \frac{dc}{dt} &= -c(\epsilon^2 + \delta^2 + \alpha o) + \epsilon\|\beta\|. && (20)\\ & && (21)\end{aligned}$$

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 K.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:

K.1 Proof of Theorem J.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 $\frac{dc}{dt} \geq -\alpha oc$. Further, so long as c is still decreasing (and therefore less than $c(0) = 1$),

$$\frac{d(\|\beta\|\epsilon - c(\epsilon^2+\delta^2))}{dt} \geq \frac{d(\|\beta\|\epsilon - (\epsilon^2+\delta^2))}{dt} \tag{35}$$

$$= (\|\beta\| - 2\epsilon)\frac{d\epsilon}{dt} - 2\delta\frac{d\delta}{dt} \tag{36}$$

$$= (\|\beta\| - 2\epsilon)(-c^2\epsilon + \|\beta\|c) - 2\delta(-c^2\delta) \tag{37}$$

$$= -c^2(\epsilon\|\beta\| - 2(\epsilon^2+\delta^2) + c(\|\beta\|^2 - 2\epsilon) \tag{38}$$

$$\geq -c(\epsilon\|\beta\| - 2(\epsilon^2+\delta^2)) + c(\|\beta\|^2 - 2\epsilon) \tag{39}$$

$$= c(\|\beta\|^2 - 2\epsilon - \epsilon\|\beta\| + 2(\epsilon^2+\delta^2)) \tag{40}$$

$$\geq c(\|\beta\|^2 - \epsilon(2 + \|\beta\|)). \tag{41}$$

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

$$\frac{d\epsilon^2}{dt} = 2\epsilon\frac{d\epsilon}{dt} \tag{42}$$

$$= -2c^2\epsilon^2 + 2\epsilon c\|\beta\| \tag{43}$$

$$\leq 2\|\beta\|^2. \tag{44}$$

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 $\frac{dc}{dt} \geq -\alpha oc$ 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

$$\frac{dc^2}{dt} = -2\alpha c^2 o, \tag{45}$$

$$\frac{do}{dt} = -2\alpha c^2 o. \tag{46}$$

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

$$c(t)^2 = \frac{m}{1 - \frac{(1-m)}{\exp(2\alpha mt)}}, \tag{47}$$

$$o(t) = \frac{m}{\frac{\exp(2\alpha mt)}{1-m} - 1} \tag{48}$$

$$= \frac{m}{\exp(2\alpha mt)(1+k^{-1}) - 1} \tag{49}$$

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 $\frac{\gamma \ln k}{2\sqrt{\alpha m}} \geq \frac{\alpha m}{\exp(2\alpha m t)(1+k^{-1})-1} \impliedby \gamma \ln k \geq \frac{2\alpha^{3/2} m^2}{\left(\frac{d_2}{d_1}\right)^{\sqrt{\alpha}}(1+k^{-1})-1} \impliedby \gamma \geq \frac{2\alpha^{3/2} m^2 k^{-\sqrt{\alpha}}}{\ln k}$. 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 $\frac{dc}{dt} \geq 0$. □

K.2 Proof of Proof of Theorem J.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

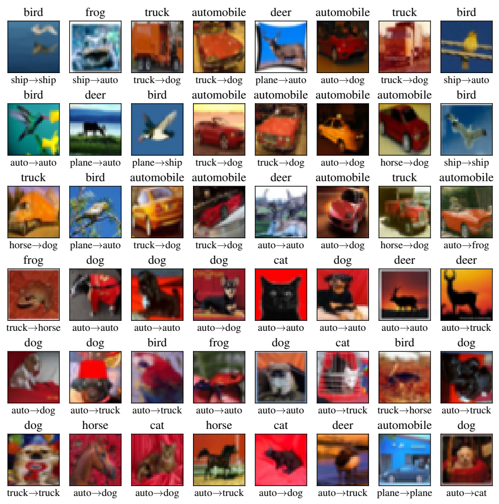
$$
\begin{aligned}
s &= \epsilon(t')^2 - \epsilon(t_2)^2 && (76)\\
&= \int_{t_2}^{t'} \frac{d\epsilon(t)^2}{dt} && (77)\\
&\leq \int_{t_2}^{t'} 2\frac{dc(t)^2}{dt} && (78)\\
&= 2(c(t')^2 - c(t_2)^2). && (79)
\end{aligned}
$$

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. $\square$

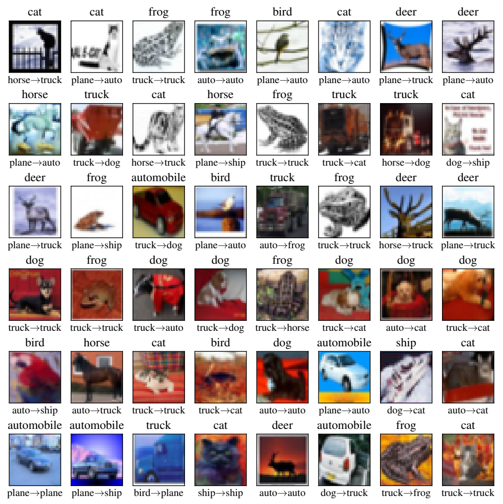
L Additional Samples Under Various Architectures/Seeds

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.

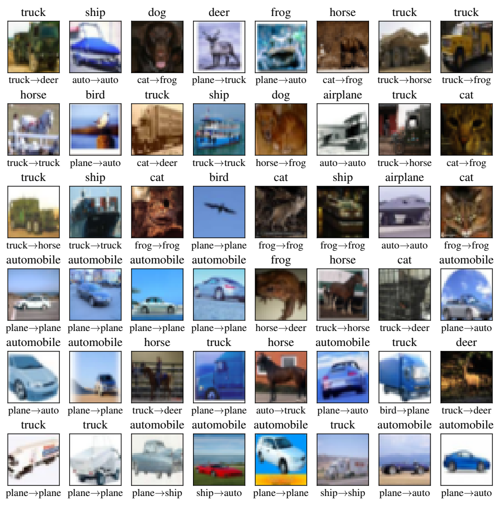
Consistent with our other experiments, we find that early opposing signals tend to be "simpler", e.g. raw colors, whereas later signals are more nuanced, such as the presence of a particular texture. We also see that the Vision Transformer seems to learn complex features earlier, and that they are less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


bird frog truck automobile deer automobile truck bird
ship→ship ship→auto truck→dog truck→dog plane→auto auto→dog truck→dog ship→auto
bird deer bird automobile automobile automobile automobile bird
auto→auto plane→auto plane→ship truck→dog truck→dog auto→dog horse→dog ship→ship
truck bird automobile automobile deer automobile truck automobile
horse→dog plane→auto truck→dog truck→dog auto→auto auto→dog horse→dog auto→frog
frog dog dog dog cat dog deer deer
truck→horse auto→auto auto→auto auto→dog auto→auto auto→auto auto→auto auto→truck
dog dog bird frog dog cat bird dog
auto→dog auto→truck auto→truck auto→auto auto→auto auto→truck truck→horse auto→truck
dog horse cat horse cat deer automobile dog
truck→truck auto→dog auto→dog auto→truck auto→dog auto→truck plane→plane auto→cat

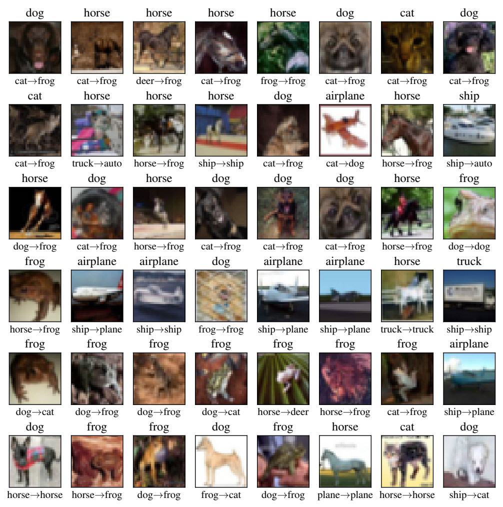
(a) Step 100 to 101


cat cat frog frog bird cat deer deer
horse→truck plane→auto truck→truck auto→auto plane→auto plane→auto plane→truck plane→auto
horse truck cat horse frog truck truck cat
plane→auto truck→dog horse→truck plane→ship truck→truck truck→cat horse→dog dog→ship
deer frog automobile bird truck frog deer deer
plane→truck plane→ship truck→dog plane→auto auto→frog truck→truck horse→truck plane→truck
dog frog dog dog frog dog dog dog
truck→truck truck→truck truck→auto truck→dog truck→horse truck→cat auto→cat truck→cat
bird horse cat bird dog automobile ship cat
auto→ship auto→truck truck→truck truck→cat auto→auto plane→auto dog→cat auto→cat
automobile automobile truck cat deer automobile frog cat
plane→plane plane→ship bird→plane ship→ship auto→auto dog→truck truck→frog truck→truck

(b) Step 250 to 251

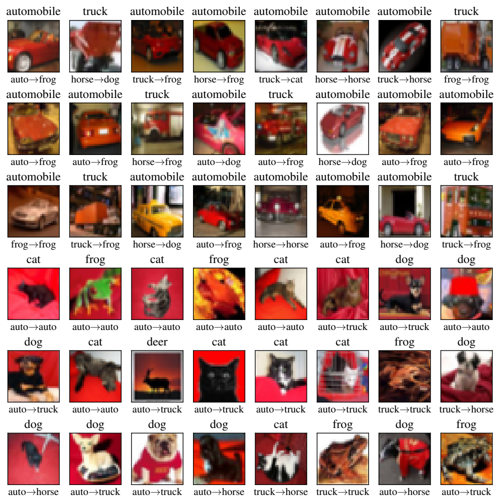

truck ship dog deer frog horse truck truck
truck→deer auto→auto cat→frog plane→truck plane→auto cat→frog truck→horse truck→frog
horse bird truck ship dog airplane truck cat
truck→truck plane→auto cat→deer truck→truck horse→frog auto→auto truck→horse cat→frog
truck ship cat bird cat ship airplane cat
truck→horse truck→truck frog→frog plane→plane frog→frog frog→frog auto→auto frog→frog
automobile automobile automobile automobile frog horse cat automobile
plane→plane plane→plane plane→plane plane→plane horse→deer truck→horse truck→deer plane→auto
automobile automobile horse truck horse automobile truck deer
plane→auto plane→plane truck→deer plane→plane auto→truck plane→auto bird→plane truck→deer
truck truck automobile automobile automobile truck automobile automobile
plane→plane plane→plane plane→ship ship→auto plane→plane ship→ship plane→auto plane→auto

(c) Step 500 to 501

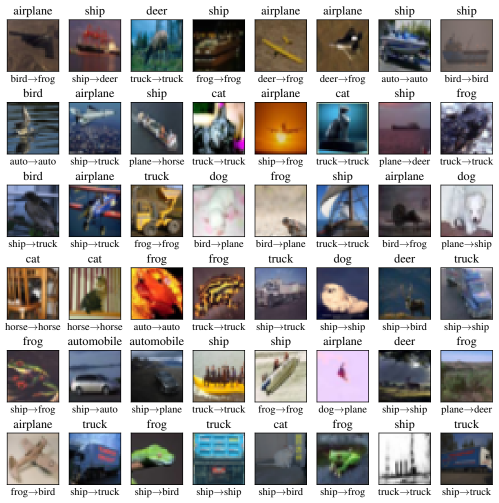

dog horse horse horse horse dog cat dog
cat→frog cat→frog deer→frog cat→frog frog→frog cat→frog cat→frog cat→frog
cat horse horse horse dog airplane horse ship
cat→frog truck→auto horse→frog ship→ship cat→frog cat→dog horse→frog ship→auto
horse dog horse dog dog dog horse frog
dog→frog cat→frog horse→frog cat→frog cat→frog cat→frog horse→frog dog→dog
frog airplane airplane dog airplane airplane horse truck
horse→frog ship→plane ship→ship frog→frog ship→plane ship→plane truck→truck ship→ship
frog frog frog frog frog frog frog airplane
dog→cat dog→frog dog→frog dog→cat horse→deer horse→frog cat→frog ship→plane
dog frog frog dog frog horse cat dog
horse→horse horse→frog dog→frog frog→cat dog→frog plane→plane horse→horse ship→cat

(d) Step 750 to 751

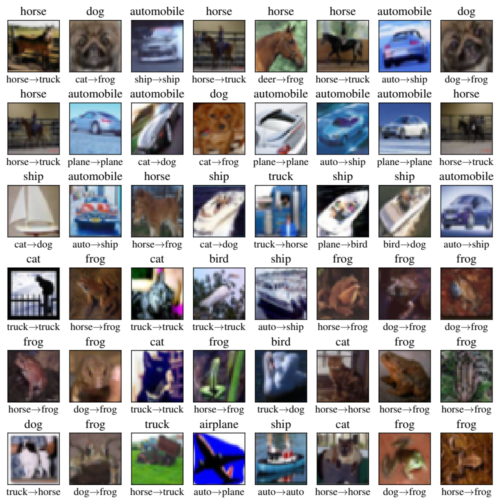
Figure 45: **(ResNet-18, seed 1)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).


automobile truck automobile automobile automobile automobile automobile truck
auto→frog horse→dog truck→frog horse→frog truck→cat horse→horse truck→horse frog→frog
automobile automobile truck automobile truck automobile automobile automobile
auto→frog auto→frog horse→frog auto→dog auto→frog horse→dog auto→frog auto→frog
automobile truck automobile automobile automobile automobile automobile truck
frog→frog truck→frog horse→dog auto→frog horse→horse auto→frog horse→dog truck→frog
cat frog cat frog cat cat dog dog
auto→auto auto→auto auto→auto auto→auto auto→auto auto→truck auto→truck auto→auto
dog cat deer cat cat cat frog dog
auto→truck auto→auto auto→truck auto→truck auto→truck auto→truck truck→truck truck→horse
dog dog dog dog cat frog dog frog
auto→horse auto→truck auto→truck auto→horse truck→horse truck→truck auto→horse auto→truck

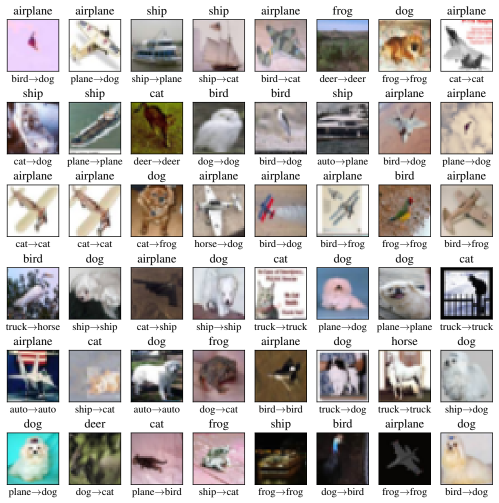
(a) Step 100 to 101


airplane ship deer ship airplane airplane ship ship
bird→frog ship→deer truck→truck frog→frog deer→frog deer→frog auto→auto bird→bird
bird airplane ship cat airplane cat ship frog
auto→auto ship→truck plane→horse truck→truck ship→frog truck→truck plane→deer truck→truck
bird airplane truck dog frog ship airplane dog
ship→truck ship→truck frog→frog bird→plane bird→plane truck→truck bird→frog plane→ship
cat cat frog frog truck dog deer truck
horse→horse horse→horse auto→auto truck→truck ship→truck ship→ship ship→bird ship→ship
frog automobile automobile ship ship airplane deer frog
ship→frog ship→auto ship→plane truck→truck frog→frog dog→plane ship→ship plane→deer
airplane truck frog truck cat frog ship truck
frog→bird ship→truck ship→bird ship→ship ship→bird ship→frog truck→truck ship→truck

(b) Step 250 to 251

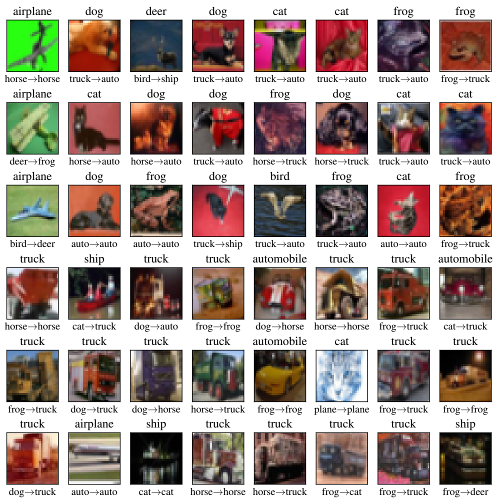

horse dog automobile horse horse horse automobile dog
horse→truck cat→frog ship→ship horse→truck deer→frog horse→truck auto→ship dog→frog
horse automobile automobile dog automobile automobile automobile horse
horse→truck plane→plane cat→dog cat→frog plane→plane auto→ship plane→plane horse→truck
ship automobile horse ship truck ship ship automobile
cat→dog auto→ship horse→frog cat→dog truck→horse plane→bird bird→dog auto→ship
cat frog cat bird ship frog frog frog
truck→truck horse→frog truck→truck truck→truck auto→ship horse→frog dog→frog dog→frog
frog frog cat frog bird cat frog frog
horse→frog dog→frog truck→truck horse→frog truck→dog horse→horse horse→frog horse→frog
dog frog truck airplane ship cat frog frog
truck→horse dog→frog horse→truck auto→plane auto→auto horse→horse dog→frog horse→frog

(c) Step 500 to 501

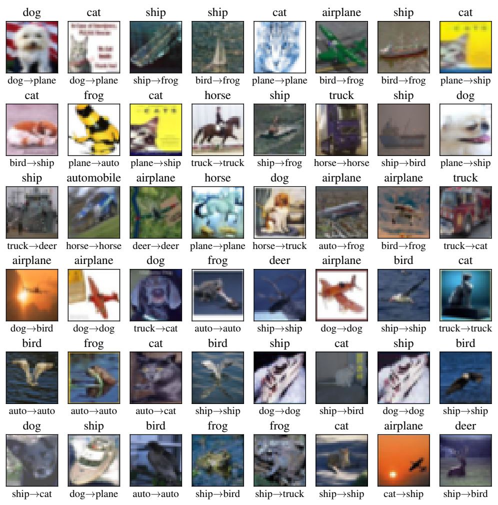

airplane airplane ship ship airplane frog dog airplane
bird→dog plane→dog ship→plane ship→cat bird→cat deer→deer frog→frog cat→cat
ship ship cat bird bird ship airplane airplane
cat→dog plane→plane deer→deer dog→dog bird→dog auto→plane bird→dog plane→dog
airplane airplane dog airplane airplane airplane bird airplane
cat→cat cat→cat cat→frog horse→dog bird→dog bird→frog frog→frog bird→frog
bird dog airplane dog cat dog dog cat
truck→horse ship→ship cat→ship ship→ship truck→truck plane→dog plane→plane truck→truck
airplane cat dog frog airplane dog horse dog
auto→auto ship→cat auto→auto dog→cat bird→bird truck→dog truck→truck ship→dog
dog deer cat frog ship bird airplane dog
plane→dog dog→cat plane→bird ship→cat frog→frog dog→bird frog→frog bird→dog

(d) Step 750 to 751

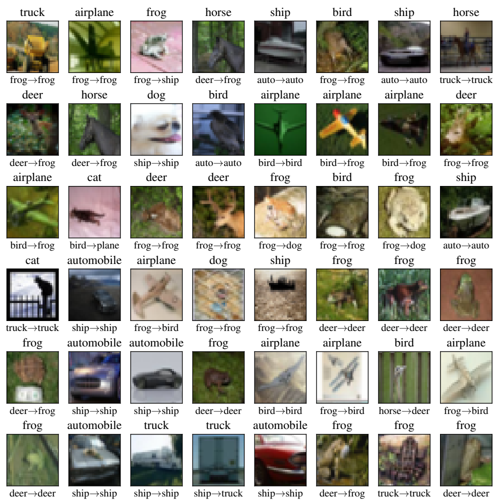
Figure 46: **(ResNet-18, seed 2)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).


airplane dog deer dog cat cat frog frog
horse→horse truck→auto bird→ship truck→auto truck→auto truck→auto truck→auto frog→truck
airplane cat dog dog frog dog cat cat
deer→frog horse→auto horse→auto truck→auto horse→truck horse→truck truck→auto truck→auto
airplane dog frog dog bird frog cat frog
bird→deer auto→auto auto→auto truck→ship truck→auto truck→auto auto→auto frog→truck
truck ship truck truck automobile truck truck automobile
horse→horse cat→truck dog→auto frog→frog dog→horse horse→horse frog→truck cat→truck
truck truck truck truck automobile cat truck truck
frog→truck dog→truck dog→horse horse→truck frog→frog plane→plane frog→truck frog→frog
truck airplane ship truck truck truck truck ship
dog→truck auto→auto cat→cat horse→horse horse→truck frog→cat frog→truck frog→deer

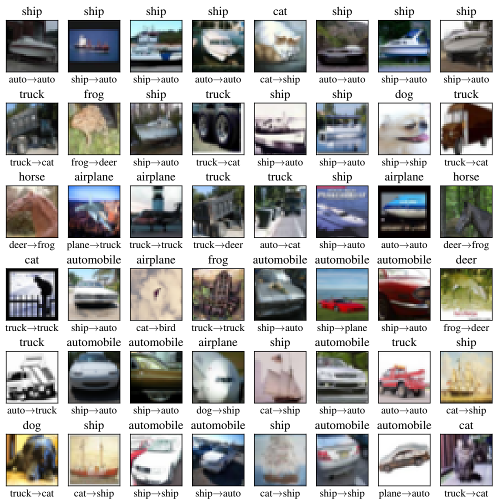
(a) Step 100 to 101


dog cat ship ship cat airplane ship cat
dog→plane dog→plane ship→frog bird→frog plane→plane bird→frog bird→frog plane→ship
cat frog cat horse ship truck ship dog
bird→ship plane→auto plane→ship truck→truck ship→frog horse→horse ship→bird plane→ship
ship automobile airplane horse dog airplane airplane truck
truck→deer horse→horse deer→deer plane→plane horse→truck auto→frog bird→frog truck→cat
airplane airplane dog frog deer airplane bird cat
dog→bird dog→dog truck→cat auto→auto ship→ship dog→dog ship→ship truck→truck
bird frog cat bird ship cat ship bird
auto→auto auto→auto auto→cat ship→ship dog→dog ship→bird dog→dog ship→ship
dog ship bird frog frog cat airplane deer
ship→cat dog→plane auto→auto ship→bird ship→truck ship→ship cat→ship ship→bird

(b) Step 250 to 251

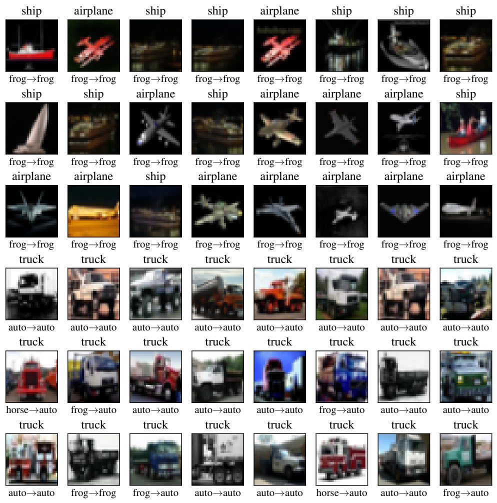

truck airplane frog horse ship bird ship horse
frog→frog frog→frog frog→ship deer→frog auto→auto frog→frog auto→auto truck→truck
deer horse dog bird airplane airplane airplane deer
deer→frog deer→frog ship→ship auto→auto bird→bird bird→frog bird→frog frog→frog
airplane cat deer deer frog bird frog ship
bird→frog bird→plane frog→frog frog→frog frog→dog frog→frog frog→dog auto→auto
cat automobile airplane dog ship frog frog frog
truck→truck ship→ship frog→bird frog→frog frog→frog deer→deer deer→deer deer→deer
frog automobile automobile frog airplane airplane bird airplane
deer→frog ship→ship ship→ship deer→deer bird→bird frog→bird horse→deer frog→bird
frog automobile truck truck automobile frog frog frog
deer→deer ship→ship ship→ship ship→truck ship→ship deer→frog truck→truck deer→deer

(c) Step 500 to 501

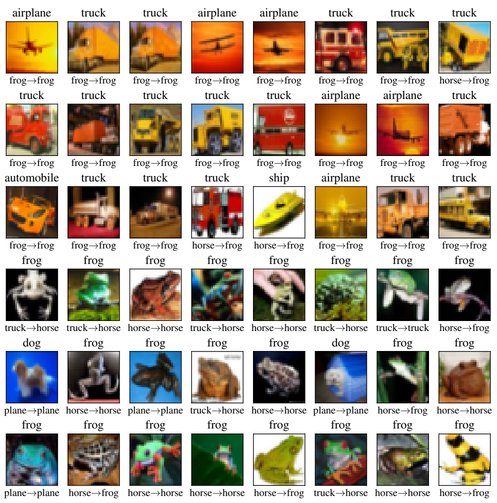

ship ship ship ship cat ship ship ship
auto→auto ship→auto ship→auto auto→auto cat→ship auto→auto ship→auto ship→auto
truck frog ship truck ship ship dog truck
truck→cat frog→deer ship→auto truck→cat ship→auto ship→auto ship→ship truck→cat
horse airplane airplane truck truck ship airplane horse
deer→frog plane→truck truck→truck truck→deer auto→cat ship→auto auto→auto deer→frog
cat automobile airplane frog automobile automobile automobile deer
truck→truck ship→auto cat→bird truck→truck ship→auto ship→plane ship→auto frog→deer
truck automobile automobile airplane ship automobile truck ship
auto→truck ship→auto ship→auto dog→ship cat→ship ship→auto auto→auto cat→ship
dog ship automobile automobile ship automobile automobile cat
truck→cat cat→ship ship→ship ship→auto cat→ship ship→ship plane→auto truck→cat

(d) Step 750 to 751

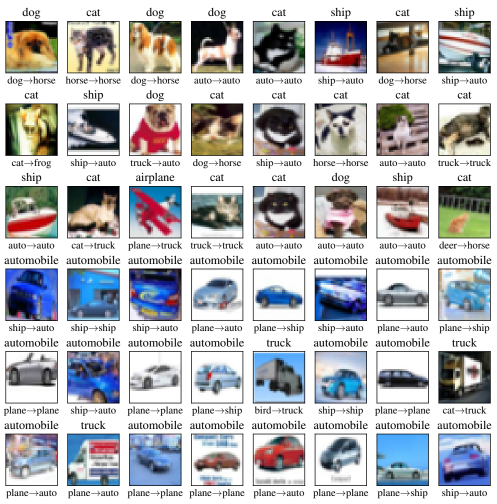
Figure 47: **(ResNet-18, seed 3)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).


ship airplane ship ship airplane ship ship ship
frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog
ship ship airplane ship airplane airplane airplane ship
frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog
airplane airplane ship airplane airplane airplane airplane airplane
frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog
truck truck truck truck truck truck truck truck
auto→auto auto→auto auto→auto auto→auto auto→auto auto→auto auto→auto auto→auto
truck truck truck truck truck truck truck truck
horse→auto frog→auto auto→auto auto→auto auto→auto frog→auto auto→auto auto→auto
truck truck truck truck truck truck truck truck
auto→auto frog→frog frog→auto auto→auto auto→auto horse→auto auto→auto frog→auto

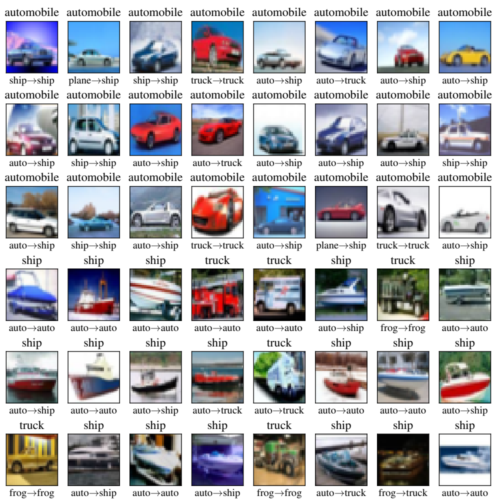
(a) Step 100 to 101


airplane truck truck airplane airplane truck truck truck
frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog horse→frog
truck truck truck truck truck airplane airplane truck
frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog
automobile truck truck truck ship airplane truck truck
frog→frog frog→frog frog→frog horse→frog horse→frog frog→frog frog→frog frog→frog
frog frog frog frog frog frog frog frog
truck→horse truck→horse horse→horse truck→horse horse→horse truck→horse truck→truck horse→frog
dog frog frog frog frog dog frog frog
plane→plane horse→horse plane→plane truck→horse horse→horse plane→plane horse→frog horse→horse
frog frog frog frog frog frog frog frog
plane→plane horse→frog horse→horse horse→horse horse→frog truck→horse horse→horse horse→frog

(b) Step 250 to 251

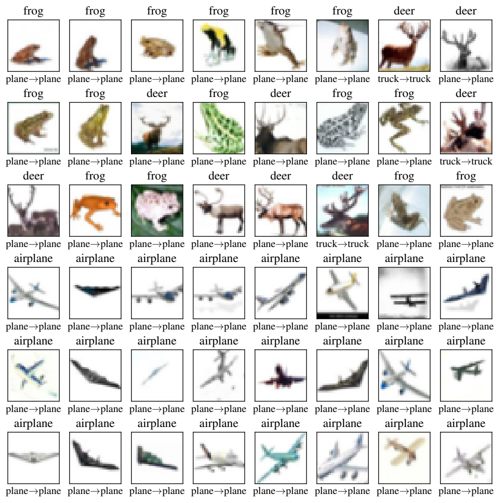

dog cat dog dog cat ship cat ship
dog→horse horse→horse dog→horse auto→auto auto→auto ship→auto dog→horse ship→auto
cat ship dog cat cat cat cat cat
cat→frog ship→auto truck→auto dog→horse ship→auto horse→horse auto→auto truck→truck
ship cat airplane cat cat dog ship cat
auto→auto cat→truck plane→truck truck→truck auto→auto auto→auto auto→auto deer→horse
automobile automobile automobile automobile automobile automobile automobile automobile
ship→auto ship→ship ship→auto plane→auto plane→ship ship→auto plane→auto plane→ship
automobile automobile automobile automobile truck automobile automobile truck
plane→plane ship→auto plane→plane plane→ship bird→truck ship→ship plane→plane cat→truck
automobile truck automobile automobile automobile automobile automobile automobile
plane→auto plane→auto plane→plane plane→plane plane→auto plane→plane plane→ship ship→auto

(c) Step 500 to 501

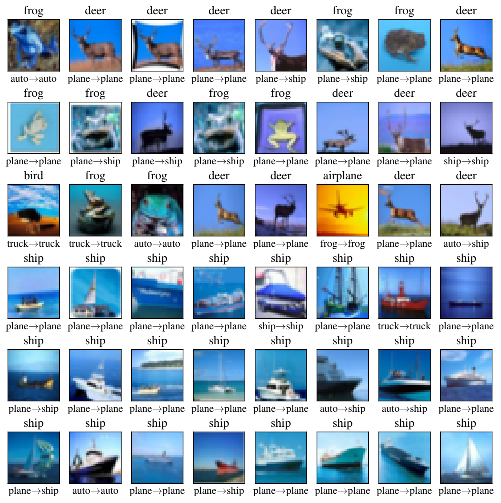

automobile automobile automobile automobile automobile automobile automobile automobile
ship→ship plane→ship ship→ship truck→truck auto→ship auto→truck auto→ship auto→ship
automobile automobile automobile automobile automobile automobile automobile automobile
auto→ship ship→ship auto→ship auto→truck auto→ship auto→ship auto→ship ship→ship
automobile automobile automobile automobile automobile automobile automobile automobile
auto→ship ship→ship auto→ship truck→truck auto→ship plane→ship truck→truck auto→ship
ship ship ship truck truck ship truck ship
auto→auto auto→auto auto→auto auto→auto auto→auto auto→ship frog→frog auto→auto
ship ship ship ship truck ship ship ship
auto→ship auto→auto auto→ship auto→truck auto→truck auto→auto auto→auto auto→ship
truck ship ship ship truck ship ship ship
frog→frog auto→ship auto→auto auto→ship frog→frog auto→truck frog→truck auto→auto

(d) Step 750 to 751

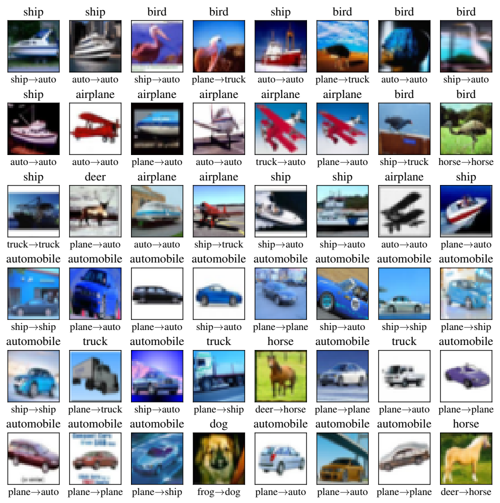
Figure 48: **(VGG-11, seed 1)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).


frog frog frog frog frog frog deer deer
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane truck→truck plane→plane
frog frog deer frog deer frog frog deer
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane truck→truck
deer frog frog deer deer deer frog frog
plane→plane plane→plane plane→plane plane→plane plane→plane truck→truck plane→plane plane→plane
airplane airplane airplane airplane airplane airplane airplane airplane
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane
airplane airplane airplane airplane airplane airplane airplane airplane
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane
airplane airplane airplane airplane airplane airplane airplane airplane
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane

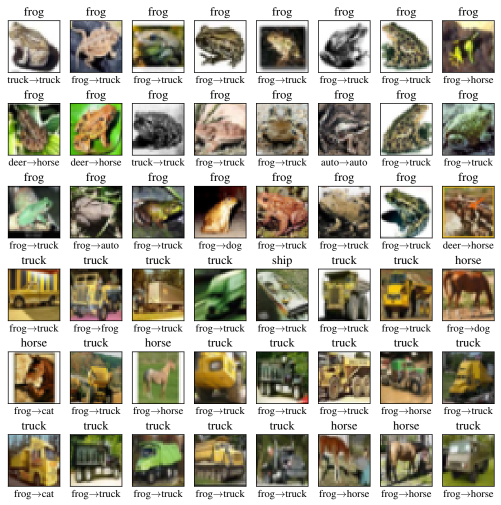
(a) Step 100 to 101


frog deer deer deer deer frog frog deer
auto→auto plane→plane plane→plane plane→plane plane→ship plane→ship plane→plane plane→plane
frog frog deer frog frog deer deer deer
plane→plane plane→ship plane→ship plane→ship plane→plane plane→plane plane→plane ship→ship
bird frog frog deer deer airplane deer deer
truck→truck truck→truck auto→auto plane→plane plane→plane frog→frog plane→plane auto→ship
ship ship ship ship ship ship ship ship
plane→plane plane→plane plane→plane plane→plane ship→ship plane→plane truck→truck plane→plane
ship ship ship ship ship ship ship ship
plane→ship plane→plane plane→plane plane→plane plane→plane auto→ship auto→ship plane→plane
ship ship ship ship ship ship ship ship
plane→ship auto→auto plane→plane plane→ship plane→plane plane→plane plane→plane plane→plane

(b) Step 250 to 251


ship ship bird bird ship bird bird bird
ship→auto auto→auto ship→auto plane→truck auto→auto plane→truck auto→auto ship→auto
ship airplane airplane airplane airplane airplane bird bird
auto→auto auto→auto plane→auto auto→auto truck→auto plane→auto ship→truck horse→horse
ship deer airplane airplane ship ship airplane ship
truck→truck plane→auto auto→auto ship→truck ship→auto ship→auto auto→auto plane→auto
automobile automobile automobile automobile automobile automobile automobile automobile
ship→ship plane→auto plane→auto ship→auto plane→plane ship→auto ship→ship plane→ship
automobile truck automobile truck horse automobile truck automobile
ship→ship plane→truck ship→auto plane→ship deer→horse plane→plane plane→auto plane→plane
automobile automobile automobile dog automobile automobile automobile horse
plane→auto plane→plane plane→ship frog→dog plane→auto plane→auto plane→plane deer→horse

(c) Step 500 to 501

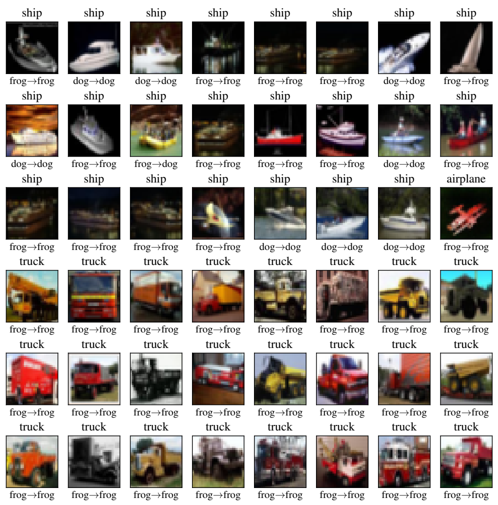

frog frog frog frog frog frog frog frog
truck→truck frog→truck frog→truck frog→truck frog→truck frog→truck frog→truck frog→horse
frog frog frog frog frog frog frog frog
deer→horse deer→horse truck→truck frog→truck frog→truck auto→auto frog→truck frog→truck
frog frog frog frog frog frog frog frog
frog→truck frog→auto frog→truck frog→dog frog→truck frog→truck frog→truck deer→horse
truck truck truck truck ship truck truck horse
frog→truck frog→frog frog→truck frog→truck frog→truck frog→truck frog→truck frog→dog
horse truck horse truck truck truck truck truck
frog→cat frog→truck frog→horse frog→truck frog→truck frog→truck frog→horse frog→truck
truck truck truck truck truck horse horse truck
frog→cat frog→truck frog→truck frog→truck frog→truck frog→horse frog→horse frog→horse

(d) Step 750 to 751

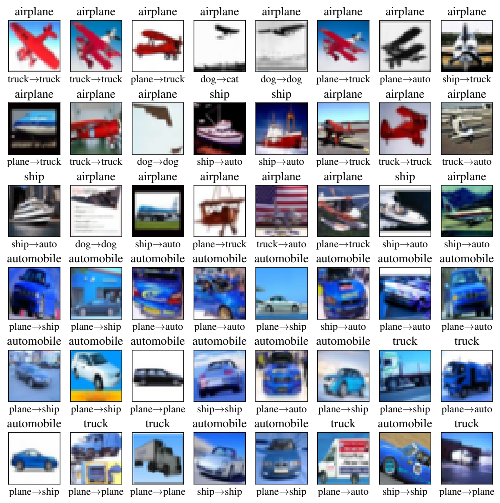
Figure 49: **(VGG-11, seed 2)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).

frog frog deer deer deer deer deer deer
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane
deer deer deer deer frog frog deer deer
auto→auto plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane
deer frog deer deer deer deer deer deer
plane→plane plane→plane plane→plane auto→auto auto→auto auto→auto plane→plane plane→plane
airplane airplane airplane airplane airplane airplane airplane airplane
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane
airplane airplane airplane airplane airplane airplane airplane airplane
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane
airplane airplane airplane airplane airplane airplane airplane airplane
plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane plane→plane

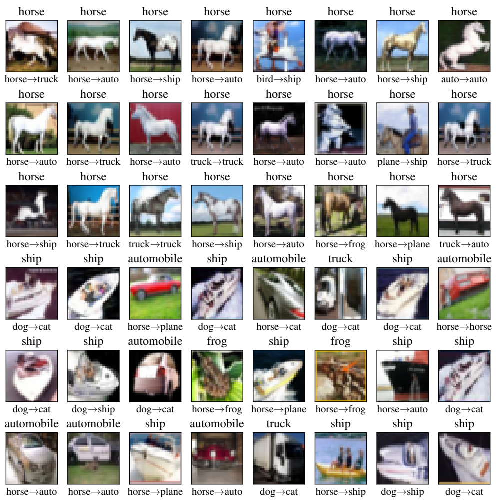
(a) Step 100 to 101


ship ship ship ship ship ship ship ship
frog→frog dog→dog dog→dog frog→frog frog→frog frog→frog dog→dog frog→frog
ship ship ship ship ship ship ship ship
dog→dog frog→frog frog→dog frog→frog frog→frog frog→frog dog→dog frog→frog
ship ship ship ship ship ship ship airplane
frog→frog frog→frog frog→frog frog→frog dog→dog dog→dog dog→dog frog→frog
truck truck truck truck truck truck truck truck
frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog
truck truck truck truck truck truck truck truck
frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog
truck truck truck truck truck truck truck truck
frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog frog→frog

(b) Step 250 to 251

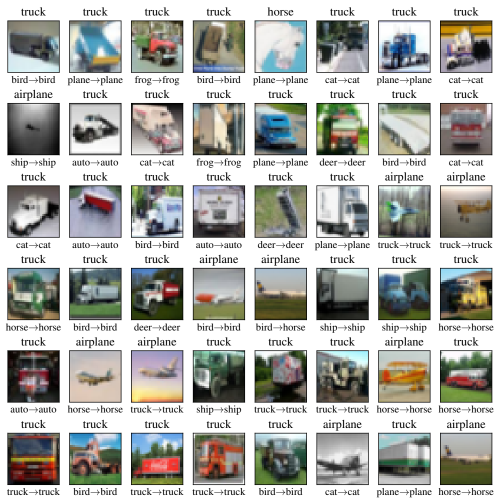

airplane airplane airplane airplane airplane airplane airplane airplane
truck→truck truck→truck plane→truck dog→cat dog→dog plane→truck plane→auto ship→truck
airplane airplane airplane ship ship airplane airplane airplane
plane→truck truck→truck dog→dog ship→auto ship→auto plane→truck truck→truck truck→auto
ship airplane airplane airplane airplane airplane ship airplane
ship→auto dog→dog ship→auto plane→truck truck→auto plane→truck ship→auto ship→auto
automobile automobile automobile automobile automobile automobile automobile automobile
plane→ship plane→ship plane→auto plane→auto plane→ship ship→auto plane→auto plane→auto
automobile automobile automobile automobile automobile automobile truck truck
plane→ship plane→ship plane→plane ship→ship plane→auto plane→ship plane→ship plane→auto
automobile truck truck automobile automobile truck automobile truck
plane→ship plane→plane plane→plane ship→ship plane→ship plane→auto ship→ship plane→plane

(c) Step 500 to 501

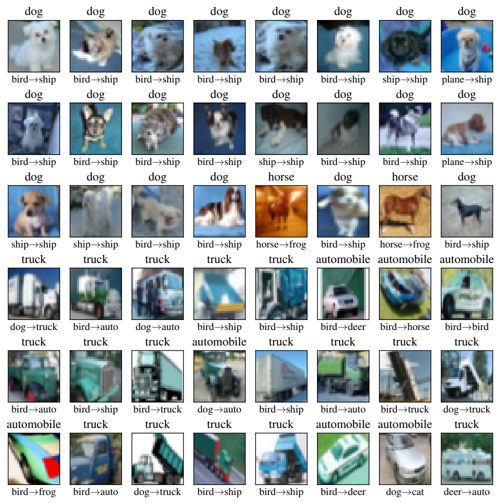

horse horse horse horse horse horse horse horse
horse→truck horse→auto horse→ship horse→auto bird→ship horse→auto horse→ship auto→auto
horse horse horse horse horse horse horse horse
horse→auto horse→truck horse→auto truck→truck horse→auto horse→auto plane→ship horse→truck
horse horse horse horse horse horse horse horse
horse→ship horse→truck truck→truck horse→ship horse→auto horse→frog horse→plane truck→auto
ship ship automobile ship automobile truck ship automobile
dog→cat dog→cat horse→plane dog→cat horse→cat dog→cat dog→cat horse→horse
ship ship automobile frog ship frog ship ship
dog→cat dog→ship dog→cat horse→frog horse→plane horse→frog horse→auto dog→cat
automobile automobile ship automobile truck ship ship ship
horse→auto horse→auto horse→plane horse→auto dog→cat horse→ship dog→ship dog→cat

(d) Step 750 to 751

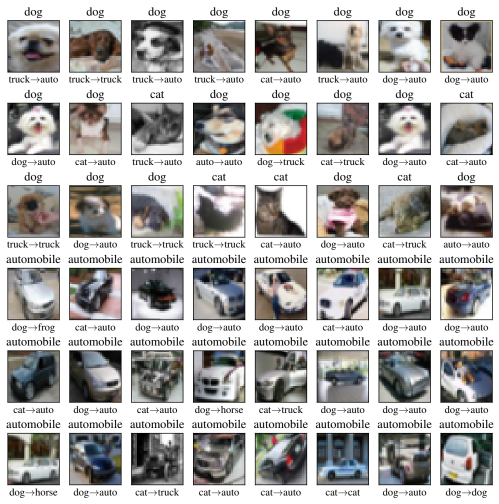
Figure 50: **(VGG-11, seed 3)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).



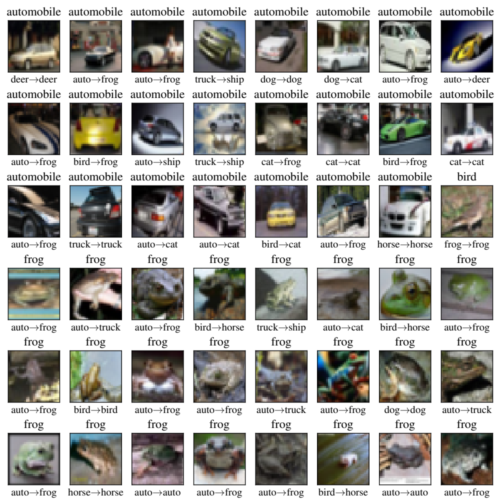
(a) Step 100 to 101



(b) Step 250 to 251

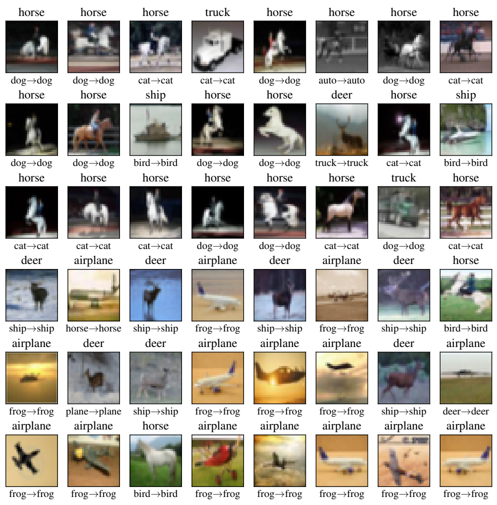


(c) Step 500 to 501

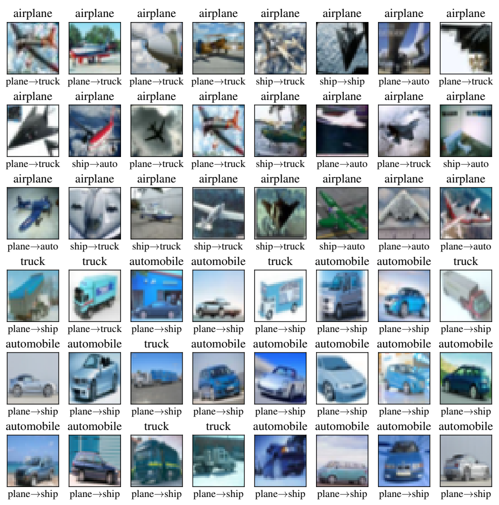


(d) Step 750 to 751

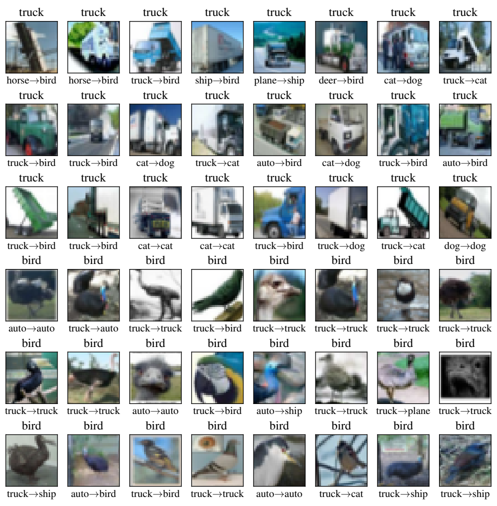
Figure 51: **(ViT, seed 1)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).


horse horse horse truck horse horse horse horse
dog→dog dog→dog cat→cat cat→cat dog→dog auto→auto dog→dog cat→cat
horse horse ship horse horse deer horse ship
dog→dog dog→dog bird→bird dog→dog dog→dog truck→truck cat→cat bird→bird
horse horse horse horse horse horse truck horse
cat→cat cat→cat cat→cat dog→dog dog→dog cat→cat dog→dog cat→cat
deer airplane deer airplane deer airplane deer horse
ship→ship horse→horse ship→ship frog→frog ship→ship frog→frog ship→ship bird→bird
airplane deer deer airplane airplane airplane deer airplane
frog→frog plane→plane ship→ship frog→frog frog→frog frog→frog ship→ship deer→deer
airplane airplane horse airplane airplane airplane airplane airplane
frog→frog frog→frog bird→bird frog→frog frog→frog frog→frog frog→frog frog→frog

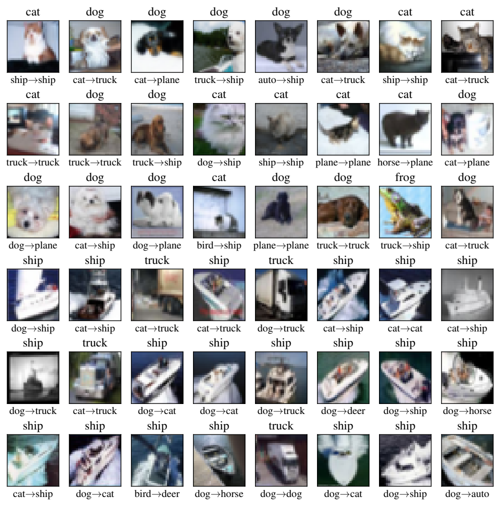
(a) Step 100 to 101


airplane airplane airplane airplane airplane airplane airplane airplane
plane→truck plane→truck plane→truck plane→truck ship→truck ship→ship plane→auto plane→truck
airplane airplane airplane airplane airplane airplane airplane airplane
plane→truck ship→auto plane→truck plane→truck ship→truck plane→auto plane→truck ship→auto
airplane airplane airplane airplane airplane airplane airplane airplane
plane→auto ship→truck ship→truck ship→truck ship→truck ship→auto plane→auto plane→auto
truck truck automobile automobile truck automobile automobile truck
plane→ship plane→truck plane→ship plane→ship plane→ship plane→ship plane→ship plane→ship
automobile automobile truck automobile automobile automobile automobile automobile
plane→ship plane→ship plane→ship plane→ship plane→ship plane→ship plane→ship plane→ship
automobile automobile truck truck automobile automobile automobile automobile
plane→ship plane→ship plane→ship plane→ship plane→ship plane→ship plane→ship plane→ship

(b) Step 250 to 251

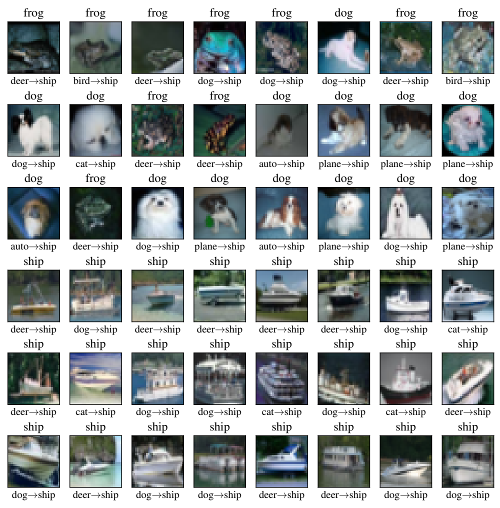

truck truck truck truck truck truck truck truck
horse→bird horse→bird truck→bird ship→bird plane→ship deer→bird cat→dog truck→cat
truck truck truck truck truck truck truck truck
truck→bird truck→bird cat→dog truck→cat auto→bird cat→dog truck→bird auto→bird
truck truck truck truck truck truck truck truck
truck→bird truck→bird cat→cat cat→cat truck→bird truck→dog truck→cat dog→dog
bird bird bird bird bird bird bird bird
auto→auto truck→auto truck→truck truck→bird truck→truck truck→truck truck→truck truck→truck
bird bird bird bird bird bird bird bird
truck→truck truck→truck auto→auto truck→bird auto→ship truck→truck truck→plane truck→truck
bird bird bird bird bird bird bird bird
truck→ship auto→bird truck→bird truck→truck auto→auto truck→cat truck→ship truck→ship

(c) Step 500 to 501

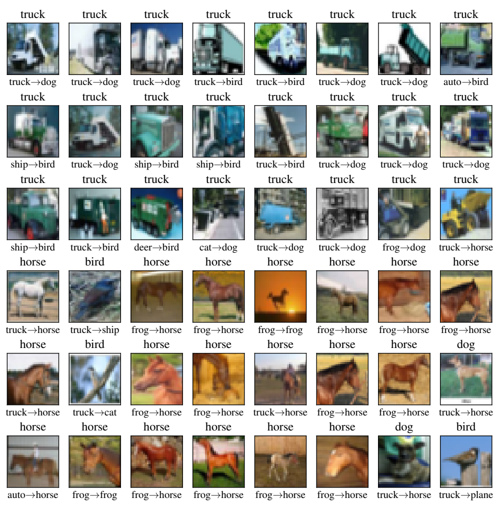

cat dog dog dog dog dog cat cat
ship→ship cat→truck cat→plane truck→ship auto→ship cat→truck ship→ship cat→truck
cat dog dog cat cat cat cat dog
truck→truck truck→truck truck→ship dog→ship ship→ship plane→plane horse→plane cat→plane
dog dog dog cat dog dog frog dog
dog→plane cat→ship dog→plane bird→ship plane→plane truck→truck truck→ship cat→truck
ship ship truck ship truck ship ship ship
dog→ship cat→ship cat→truck cat→truck dog→truck cat→ship cat→cat cat→ship
ship truck ship ship ship ship ship ship
dog→truck cat→truck dog→cat dog→cat dog→truck dog→deer dog→ship dog→horse
ship ship ship ship truck ship ship ship
cat→ship dog→cat bird→deer dog→horse dog→dog dog→cat dog→ship dog→auto

(d) Step 750 to 751

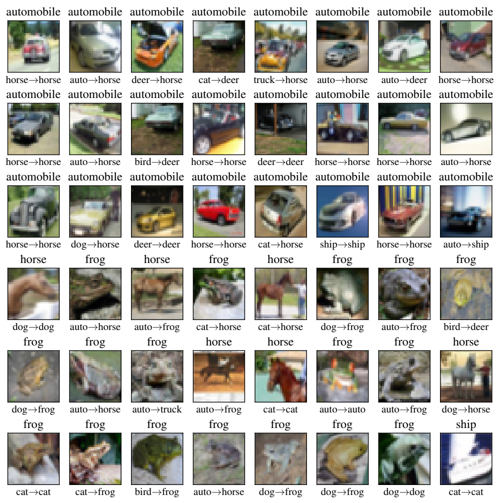
Figure 52: **(ViT, seed 2)** Images with the most positive (top 3 rows) and most negative (bottom 3 rows) change to training loss after steps 100, 250, 500, and 750. Each image has the true label (above) and the predicted label before and after the gradient update (below).


frog frog frog frog frog dog frog frog
deer→ship bird→ship deer→ship dog→ship dog→ship dog→ship deer→ship bird→ship
dog dog frog frog dog dog dog dog
dog→ship cat→ship deer→ship deer→ship auto→ship plane→ship plane→ship plane→ship
dog frog dog dog dog dog dog dog
auto→ship deer→ship dog→ship plane→ship auto→ship plane→ship dog→ship plane→ship
ship ship ship ship ship ship ship ship
deer→ship dog→ship deer→ship deer→ship deer→ship deer→ship dog→ship cat→ship
ship ship ship ship ship ship ship ship
deer→ship cat→ship dog→ship dog→ship cat→ship dog→ship cat→ship deer→ship
ship ship ship ship ship ship ship ship
dog→ship deer→ship dog→ship dog→ship deer→ship deer→ship dog→ship dog→ship

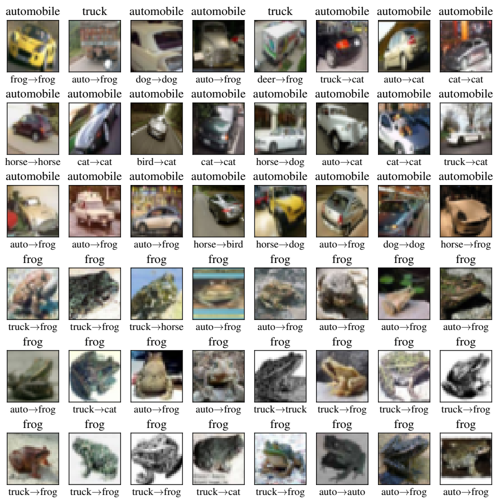
(a) Step 100 to 101


truck truck truck truck truck truck truck truck
truck→dog truck→dog truck→dog truck→bird truck→bird truck→dog truck→dog auto→bird
truck truck truck truck truck truck truck truck
ship→bird truck→dog ship→bird ship→bird truck→bird truck→dog truck→dog truck→dog
truck truck truck truck truck truck truck truck
ship→bird truck→bird deer→bird cat→dog truck→dog truck→dog frog→dog truck→horse
horse bird horse horse horse horse horse horse
truck→horse truck→ship frog→horse frog→horse frog→frog frog→horse frog→horse frog→horse
horse bird horse horse horse horse horse dog
truck→horse truck→cat frog→horse frog→horse truck→horse frog→horse frog→horse truck→horse
horse horse horse horse horse horse dog bird
auto→horse frog→frog frog→horse frog→horse frog→horse frog→horse truck→horse truck→plane

(b) Step 250 to 251


automobile automobile automobile automobile automobile automobile automobile automobile
horse→horse auto→horse deer→horse cat→deer truck→horse auto→horse auto→deer horse→horse
automobile automobile automobile automobile automobile automobile automobile automobile
horse→horse auto→horse bird→deer horse→horse deer→deer horse→horse horse→horse auto→horse
automobile automobile automobile automobile automobile automobile automobile automobile
horse→horse dog→horse deer→deer horse→horse cat→horse ship→ship horse→horse auto→ship
horse frog horse frog horse frog frog frog
dog→dog auto→horse auto→frog cat→horse cat→horse dog→frog auto→frog bird→deer
frog frog frog horse horse frog frog horse
dog→frog auto→horse auto→truck auto→frog cat→cat auto→auto auto→frog dog→horse
frog frog frog frog frog frog frog ship
cat→cat cat→frog bird→frog auto→horse dog→frog dog→frog dog→dog cat→cat

(c) Step 500 to 501


automobile truck automobile automobile truck automobile automobile automobile
frog→frog auto→frog dog→dog auto→frog deer→frog truck→cat auto→cat cat→cat
automobile automobile automobile automobile automobile automobile automobile automobile
horse→horse cat→cat bird→cat cat→cat horse→dog auto→cat cat→cat truck→cat
automobile automobile automobile automobile automobile automobile automobile automobile
auto→frog auto→frog auto→frog horse→bird horse→dog auto→frog dog→dog horse→frog
frog frog frog frog frog frog frog frog
truck→frog truck→frog truck→horse auto→frog auto→frog auto→frog auto→frog auto→frog
frog frog frog frog frog frog frog frog
auto→frog truck→cat auto→frog auto→frog truck→truck truck→frog truck→frog truck→frog
frog frog frog frog frog frog frog frog
truck→frog truck→frog truck→frog truck→cat truck→frog auto→auto auto→frog auto→frog

(d) Step 750 to 751

Figure 53: **(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).