Complexity Matters: Dynamics of Feature Learning in the Presence of Spurious Correlations

GuanWen Qiu, Da Kuang, Surbhi Goel
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104, USA
{guanwenq, kuangda, surbhig}@seas.upenn.edu

Abstract

Existing research often posits spurious features as *easier* to learn than core features in neural network optimization, but the nuanced impact of their relative simplicity remains under-explored. In this paper, we propose a theoretical framework and associated synthetic dataset grounded in boolean function analysis. Our framework allows for fine-grained control on both the relative complexity (compared to core features) and correlation strength (with respect to the label) of spurious features. Experimentally, we observe that the presence of stronger spurious correlations or simpler spurious features leads to a slower rate of learning for the core features in networks when trained with (stochastic) gradient descent. Perhaps surprisingly, we also observe that spurious features are not forgotten even when the network has perfectly learned the core features. We give theoretical justifications for these observations for the special case of learning with parity features on a one-layer hidden network. Our findings justify the success of retraining the last layer for accelerating core feature convergence and identify limitations of debiasing algorithms that exploit early learning of spurious features. We corroborate our findings through experiments on real-world vision datasets, thereby validating the practical relevance of our framework.

1 Introduction

There is increasing evidence [10, 49, 11, 43, 23] indicating that neural networks inherently tend to learn *spurious* features in classification tasks. These features, while correlated with the data label, are non-causal and lead to enhanced training and in-distribution performance. However, this inherent tendency overlooks core or invariant features that are crucial for robustness against distribution shifts. This phenomenon is attributed to the relative simplicity of spurious features compared to core features, reflecting a *simplicity bias* in neural network training [10, 37, 32, 27, 45], where networks inherently prefer simpler features over more complex, yet essential ones. Interestingly, recent empirical work [18, 14] has shown that despite this bias and the compromised predictive performance, standard neural network training does in fact learn the harder core features in its representation, as long as the spurious correlation is not perfect. However, a fine-grained understanding of the impact of "simplicity" of the spurious features on the learning of the robust features has remained unexplored. Moreover, a precise definition of simplicity that accounts for computational aspects of learning is lacking.

In our work, we characterize the impact of the relative complexity of spurious features and their correlation strength with the true label on the dynamics of core feature learning in neural networks trained with (stochastic) gradient descent. To ground our exploration, we introduce a versatile framework and corresponding synthetic datasets based on the rich theory of boolean functions (see Appendix A.2 for a quick review). We quantify simplicity/complexity using the *computational time*

Mathematics of Modern Machine Learning Workshop at NeurIPS 2023.

of learning the different features (represented as boolean functions) by gradient-based training, and subsequently study the dynamics of gradient-based learning on these datasets. We focus on two types of boolean functions: *parity* and *staircase* functions [1]. Our key findings are:

- **Simpler spurious features lead to slower core feature convergence**. We find that the presence of spurious features slows down the convergence rate of core feature learning. Moreover, *easier* spurious features often lead to *slower* convergence compared to more complex spurious features. However, even a spurious feature with similar or slightly lower complexity than the core feature can significantly impair the convergence rate. This slower convergence leads to diminished performance in scenarios with limited data.
- **Core features are also partly learned with spurious features**. We find that for staircase functions, which most closely mimic how real-world data behaves, at the stage when spurious features are fully learned, the network has also learned part of the core features. The extent to which the core features are learned depends on their relative complexity. This observation challenges the effectiveness of widely adopted machine learning algorithms that heavily depend on early learning of shortcut features and make an assumption of a clear-cut separation between the learning phases of core and spurious features [20, 21, 39, 28, 46].
- **Spurious features are memorized**. We observe that even when the network learns the core features, the spurious features, particularly those with lower complexity compared to the core features, tend to stay memorized in the representation. This addresses a question posed in [12].

2 Boolean Features Dataset

We encapsulate features by boolean functions of the input variables and the hardness of the feature by the computational complexity of the corresponding boolean feature, that is, time taken to learn using gradient-based approaches. We create two boolean features on a set of variables: the core feature which completely predicts the label, and a spurious feature which predicts the label for λ fraction of the samples, but with smaller complexity. More formally, consider two boolean functions $f_c : \{+1, -1\}^c \to \{+1, -1\}, f_s : \{+1, -1\}^s \to \{+1, -1\}$, which we call core and spurious function respectively. Here $c, s \in \mathbb{N}$. We also have a constant $\lambda \in [0, 1]$ that represents the confounder strength of the dataset, and $u \in \mathbb{N}$ which specify the number of random or independent variables. We first form two distributions, and then combine them to form the distribution $\mathcal{D}_\lambda$ as follows:

- $\mathcal{D}_{\mathsf{same}}$**:** Uniformly select two vectors x_c, x_u from the set $\{+1, -1\}^c$ and $\{+1, -1\}^u$ respectively, and set the label $y = f_c(x_c)$. Now, select vector x_s randomly from the set $\{x_s \in \{+1, -1\}^s | f_s(x_s) = f_c(x_c)\}$ such that it shares the same label as x_c under the function f_s. $([x_c, x_s, x_u], y)$ then gives us one sample from $\mathcal{D}_{\mathsf{same}}$.
- $\mathcal{D}_{\mathsf{diff}}$**:** Sample x_c, x_u, y as in $\mathcal{D}_{\mathsf{same}}$. Now, sample x_s uniformly from $\{x_s \in \{+1, -1\}^s | f_s(x_s) = -f_c(x_c)\}$ such that it disagrees with y under the function f_s.
- $\mathcal{D}_\lambda$: With probability λ draw a sample from $\mathcal{D}_{\mathsf{same}}$ and with $1 - \lambda$ draw a sample from $\mathcal{D}_{\mathsf{diff}}$.[1]

Note that the above framework allows us to control both strength of spurious features using λ and the complexity in comparison with core features with the choice of the boolean features. See Appendix A.1 for a comparison to other datasets created to investigate learning with spurious features.

Boolean functions: Parity and Staircase. We focus on two specific scenarios: one where both core and spurious functions are parity functions i.e $f(x) = x_1x_2x_3...x_d$, and another where both take the form of leap 1 threshold staircase functions $f(x) = sign(x_1 + x_1x_2 + x_1x_2x_3 + ... + x_1x_2...x_d)$.It is noteworthy that threshold staircase functions under classification settings shows a similar learning trajectory to the regression case in [1] (Figure 2), illustrating the function is fitted from lower to higher degree. For parity functions, it is well-understood that the computational complexity of gradient descent training [6] depends exponentially on the degree. In contrast, for staircase functions, the exponential dependence is on the *leap* complexity (which is 1 for our setting) and the dependence on degree is only linear. We make this selection with strategic intent: (1) despite having the same degree, both have very different training dynamics, (2) the parity case facilitates theoretical analysis, and (3) the learning process on a staircase function exhibits a loss curve that closely mirrors that

[1]When either of the feature is unbiased, for $\lambda > 0.5$, the distribution is equivalent to a mixture of $2(1 - \lambda)\mathcal{D}_{\mathsf{unif}}(x) + (2\lambda - 1)\mathcal{D}_{\mathsf{same}}(x)$ where $\mathcal{D}_{\mathsf{unif}}$ is the uniform distribution on x_s, x_c, x_u and $y = f_c(x_c)$. See Appendix A.3 for the proof.

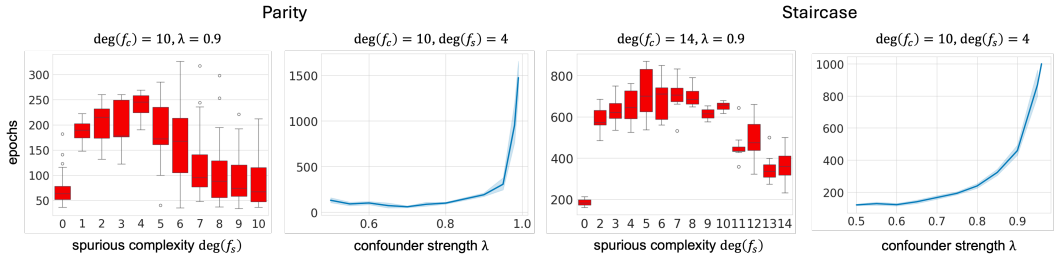


Figure 1: Influence of confounder strength and complexity of spurious correlation on learning of core features. The y-axis shows the number of epochs required to reach 0.95 core correlation.

of real-world datasets. This similarity likely stems from the inherent hierarchical nature observed in real-world data, which is effectively captured by the staircase function model [46, 44].

Related work: Spurious Datasets. Numerous works have proposed different theoretical and experimental setups to study spurious correlation. It is noteworthy that spurious features are often used interchangeably with 'shortcut' or 'easier' features. However, different works have drastically different notions to encapsulate the easiness of a feature. For example, [37] examines features along different dimensions, quantifying simplicity by the number of linear segments needed for perfectly separating the data. [41, 47, 35] encapsulate both spurious and core features as 1-bit vectors, gauging simplicity by the amount or variance of noise applied to each feature. Despite our framework bearing resemblance to previously proposed notions of simplicity, we distinguish ourselves by: (1) using non-linear features for spurious and core features, (2) focusing on the computational aspects of learning to characterize complexity rather than representational properties, and (3) capturing the hierarchical learning aspect of neural networks. See Appendix A.1 for more detail on the background.

3 Empirical Evaluation

Here, we provide a thorough evaluation of a two-layer neural network (width 100) optimized using Stochastic Gradient Descent with the cross entropy loss on the boolean features dataset. The exact experimental setup can be found in the Appendix C.1. We mainly focus on two metrics to measure feature learning: (1) Core and spurious correlation: correlation between the model and core or spurious feature measured by $\mathbb{E}_{x\sim\mathcal{D}_{\text{unif}}}[f(x)f_m(x)]$ where f_m is the model and f is either f_s or f_c, (2) Decoded core and spurious correlation [18, 12, 33, 3]: we first retrain the last layer to fit either the spurious or core function. Then measure the corresponding correlation as (1). The latter captures the information the representation has about the core/spurious features.

(R1) Simpler spurious features and higher correlation strength slow down the convergence rate of core feature learning (Figure 1). We observe an upside U-shaped phenomenon for the dependence on complexity of spurious feature and convergence time, with lower complexity features slowing down convergence. Even when the spurious feature is nearly as hard as the core feature, the model's performance still suffers from the presence of the spurious feature. In addition, the slower convergence of learning core feature implies worse end performance on limited size datasets (see table 1). The learning process appears to be relatively insensitive to the confounder strength until a certain point, beyond which there is a sudden and substantial increase in the computational time required to learn the core feature [2].

Table 1: Trained model's core correlation/decoded core correlation on Domino datasets of varying complexity. Core feature is CIFAR truck/automobile.

	Pretrained	*Random*
MNIST 01	0.71/0.86	0.47/0.69
MNIST 79	0.72/0.88	0.59/0.82
Fashion MNIST	0.75/0.86	0.70/0.85

(R2) Spurious correlation increases first and then decreases once core features catch up. However, decoded spurious correlation remains high throughout (Figure 2). We observe that once the spurious features are learned, they are memorized in the last layer. Feature learning for spurious

[2]Note that our experiments show that even when confounder strength is as high as 0.99, the model still fits the core function perfectly eventually. Refer to Appendix C.2.2 for more details

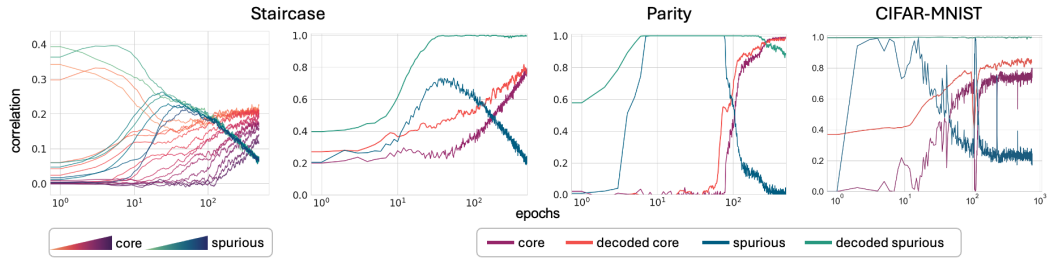


Figure 2: Core/spurious correlation and decoded correlation dynamics of different datasets. Leftmost figure shows the fourier coefficients of both the spurious and core function are fitted from low (light color) to high (deep color) for the staircase function. All of the experiments have $\lambda = 0.9$

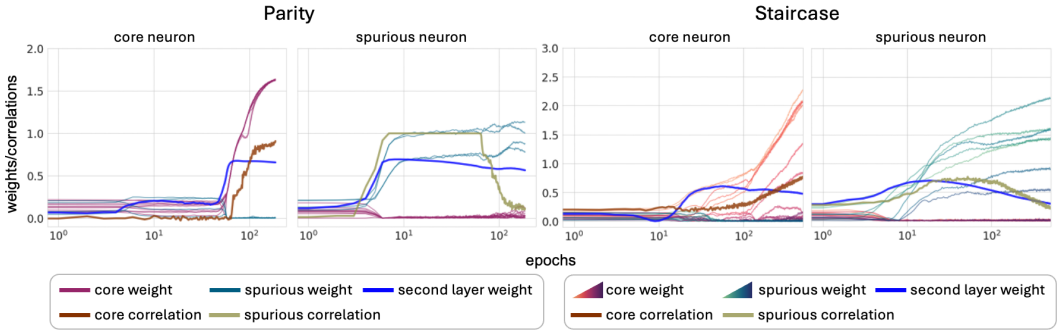


Figure 3: Weights on spurious and core coordinates in "spurious" and "core" neurons throughout training. Spurious neuron remain spurious while core neurons eventually emerge.

features ends roughly at the point when the spurious correlation starts to decrease. In addition, we see that the last layer retraining boosts core feature correlation and the boost is most significant in the early stage of training when either the correlation strength is higher or the spurious features are simpler. The benefit of retraining the last layer diminishes as we train the model further (see Figure 11 in the Appendix).

(R3) Spurious and core features are learned by two separate sub-networks (Figure 3). There exists a classification of neurons into two groups, "spurious neurons" which have larger weights on the spurious index and "core neurons" which have larger weights on the core index in the late stage of learning. For both parity and staircase tasks, almost all spurious neurons remain focused on spurious coordinates, while core neurons, at the start, do not focus on spurious coordinates and gradually develop an emphasis on core coordinates. A similar trend was observed by [24] in the context of grokking. See Appendix C.3 for more detail. We have also noticed that the number of spurious neurons depends on both the confounder strength and complexity of the spurious feature. See Appendix 2 for the statistics of the number of spurious neuron. An interesting finding is that the number of spurious neurons seems to be correlated with the time required to learn the core feature.

(R4) Popular debiasing algorithms fail in more general settings. In popular spurious datasets e.g. image Domino dataset[37], waterbirds[34], Color-MNIST [4], the spurious feature is much easier to learn than the core feature. Various debiasing algorithms [20, 21, 39, 28, 46] heavily rely on the early learning of the easy spurious features to identify and up-weight the minority group when the spurious attribute is not provided. They inherently assume a clear demarcation between the learning phases of spurious features and core features caused by heavy confounder strength and simplicity of the spurious feature, in particular, they implicitly assume that there should be a distinct temporal point at which the model is correlated much more with the spurious features than the core features. However, our findings indicate that this demarcation may remain ambiguous throughout training, especially for our spurious staircase dataset. Consequently, these debiasing algorithms will have a hard time correctly distinguishing minority groups from other groups and potentially introduce unwanted bias in the model. To quantitatively establish this, we use Jaccard score $J(A, B) = \frac{A \cap B}{A \cup B}$ to measure the quality of the inferred minority group from the true minority group throughout training (see the right

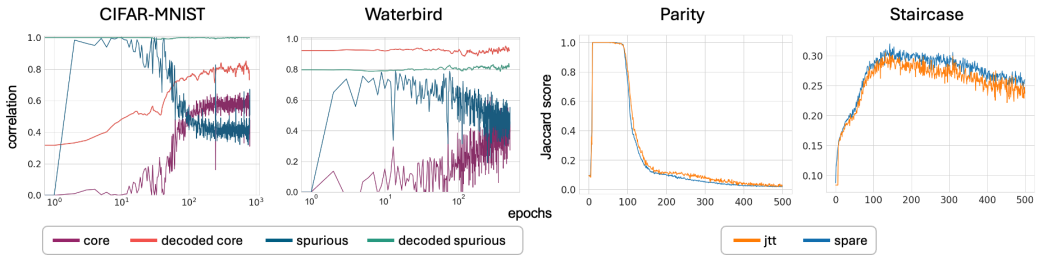


Figure 4: The left two figures demonstrate the learning dynamics of widely-used spurious datasets (under $\lambda = 0.95$), highlighting an early spike in spurious correlation. The third figure reveals that JTT [20] and Spare [47] algorithms effectively infer the minority group for parity case. While for the spurious staircase experiment conducted with limited dataset shown in the fourth plot ($\lambda = 0.9, deg(f_s) = 10, deg(f_c) = 14$), the highest Jaccard score remains below 0.5, indicating a complete breakdown of the algorithms. See C.2.2 for more detail.

two plots in Figure 4). It should be noted that these algorithms would improve the performance of the model only if at some point the Jaccard score is significantly higher than 0.5 because this is a necessary condition for the minority group to be up-weighted in the second stage of training. Our experiment detail and a quick summary of these algorithms are listed in Appendix C.4.

4 Theoretical Explanations

We do not attempt to provide an end-to-end result of the observed dynamics of feature learning on the spurious parity case, which seems very challenging given the unresolved Fourier-gap conjecture [6]. We instead give an insightful, but non-rigorous, theoretical justification based on Fourier-gaps of spurious and core features relative to the independent variables at initialization, and subsequently, after the spurious feature has been learned. See Appendix B for the calculations.

Lemma 1 (informal)**.** *Let $\xi_k = \widehat{\mathsf{Maj}}([k])$ be the k-th Fourier coefficient of the Majority function. At initialization, there is a set of neurons such that the population gradient gap on the variables compared to the independent variables[3] is:*

1. ***Spurious Variable**: $-(\lambda - \frac{1}{2})(\xi_{s-1} - \xi_{s+1})$,*
2. ***Core Variable**: $-\frac{1}{2}(\xi_{c-1} - \xi_{c+1})$.*

We know that $|\xi_k| \approx \Theta\left(n^{-(k-1)/2}\right)$ (where $n = c + s + u$) is monotonically decreasing with k, and thus we see the population gradient gap is higher for the spurious feature than the core feature. This suggests that the spurious feature would be learned first, as is true from our observations.

Lemma 2 (informal)**.** *Suppose the function has become fully correlated with the spurious feature, that is, the model is only making error on $\mathcal{D}_{\text{diff}}$, then there is a set of neurons (that are still close to initialization) for which the gap in population gradient compared to the independent variables is:*

1. ***Spurious Variable**: $-\frac{1}{2}(1 - \lambda)(\xi_{s+1} - \xi_{s-1})$,*
2. ***Core Variable**: $-\frac{1}{2}(1 - \lambda)(\xi_{c-1} - \xi_{c+1})$.*

Once the spurious feature is learned, the gradient on the spurious coordinates is going to reduce. In contrast, the core feature will continue to increase. Note that in this second phase, the gradient signal on the core is scaled down by $1 - \lambda$, which implies slower convergence with increasing correlation strength. Furthermore, the second phase will start after the spurious feature is learned, hence simpler spurious features would lead to a longer phase with lower gradient signal, and slower convergence.

5 Acknowledgements

We thank Ben L. Edelman for insightful discussions and the anonymous reviewers for their feedback.

[3]These quantities are negative because they refer to the gradient and when we update the weights using gradient descent, the sign will cancel to have a positive contribution on the weights.

References

[1] E. Abbe, E. Boix-Adsera, and T. Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for SGD learning of sparse functions on two-layer neural networks, Feb. 2022. arXiv:2202.08658 [cs, stat].

[2] E. Abbe, E. Boix-Adsera, and T. Misiakiewicz. SGD learning on neural networks: leap complexity and saddle-to-saddle dynamics, Aug. 2023. arXiv:2302.11055 [cs, stat].

[3] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes, Nov. 2018. arXiv:1610.01644 [cs, stat].

[4] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant Risk Minimization, Mar. 2020. arXiv:1907.02893 [cs, stat].

[5] R. J. N. Baldock, H. Maennel, and B. Neyshabur. Deep Learning Through the Lens of Example Difficulty, June 2021. arXiv:2106.09647 [cs, stat].

[6] B. Barak, B. L. Edelman, S. Goel, S. Kakade, E. Malach, and C. Zhang. Hidden Progress in Deep Learning: SGD Learns Parities Near the Computational Limit, Jan. 2023. arXiv:2207.08799 [cs, math, stat].

[7] A. Daniely and E. Malach. Learning Parities with Neural Networks, July 2020. arXiv:2002.07400 [cs, stat].

[8] C. Duchene, H. Jamet, P. Guillaume, and R. Dehak. A benchmark for toxic comment classification on Civil Comments dataset, Jan. 2023. arXiv:2301.11125 [cs, eess].

[9] B. L. Edelman, S. Goel, S. Kakade, E. Malach, and C. Zhang. Pareto Frontiers in Neural Feature Learning: Data, Compute, Width, and Luck, Sept. 2023. arXiv:2309.03800 [cs, stat] version: 1.

[10] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut Learning in Deep Neural Networks. *Nature Machine Intelligence*, 2(11):665–673, Nov. 2020. arXiv:2004.07780 [cs, q-bio].

[11] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness, Nov. 2022. arXiv:1811.12231 [cs, q-bio, stat].

[12] K. L. Hermann and A. K. Lampinen. What shapes feature representations? Exploring datasets, architectures, and training, Oct. 2020. arXiv:2006.12433 [cs, stat].

[13] B. Y. Idrissi, M. Arjovsky, M. Pezeshki, and D. Lopez-Paz. Simple data balancing achieves competitive worst-group-accuracy, Feb. 2022. arXiv:2110.14503 [cs].

[14] P. Izmailov, P. Kirichenko, N. Gruver, and A. G. Wilson. On Feature Learning in the Presence of Spurious Correlations, Oct. 2022. arXiv:2210.11369 [cs, stat].

[15] S. Joshi, Y. Yang, Y. Xue, W. Yang, and B. Mirzasoleiman. Towards Mitigating Spurious Correlations in the Wild: A Benchmark and a more Realistic Dataset, Sept. 2023. arXiv:2306.11957 [cs].

[16] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis. Decoupling Representation and Classifier for Long-Tailed Recognition, Feb. 2020. arXiv:1910.09217 [cs].

[17] D. S. Kermany, M. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, J. Pei, M. Y. L. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zheng, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. N. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, and K. Zhang. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*, 172(5):1122–1131.e9, Feb. 2018. Publisher: Elsevier.

[18] P. Kirichenko, P. Izmailov, and A. G. Wilson. Last Layer Re-Training is Sufficient for Robustness to Spurious Correlations, June 2023. arXiv:2204.02937 [cs, stat].

[19] Y. Lee, H. Yao, and C. Finn. Diversify and Disambiguate: Learning From Underspecified Data, Feb. 2023. arXiv:2202.03418 [cs, stat].

[20] E. Z. Liu, B. Haghgoo, A. S. Chen, A. Raghunathan, P. W. Koh, S. Sagawa, P. Liang, and C. Finn. Just Train Twice: Improving Group Robustness without Training Group Information, Sept. 2021. arXiv:2107.09044 [cs, stat].

[21] S. Liu, X. Zhang, N. Sekhar, Y. Wu, P. Singhal, and C. Fernandez-Granda. Avoiding spurious correlations via logit correction, Feb. 2023. arXiv:2212.01433 [cs, eess, stat].

[22] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep Learning Face Attributes in the Wild, Sept. 2015. arXiv:1411.7766 [cs] version: 3.

[23] T. McCoy, E. Pavlick, and T. Linzen. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics.

[24] W. Merrill, N. Tsilivis, and A. Shukla. A Tale of Two Circuits: Grokking as Competition of Sparse and Dense Subnetworks, Mar. 2023. arXiv:2303.11873 [cs].

[25] D. Morwani, J. Batra, P. Jain, and P. Netrapalli. Simplicity Bias in 1-Hidden Layer Neural Networks, Feb. 2023. arXiv:2302.00457 [cs, stat].

[26] N. Murali, A. M. Puli, K. Yu, R. Ranganath, and K. Batmanghelich. Shortcut Learning Through the Lens of Early Training Dynamics, Feb. 2023. arXiv:2302.09344 [cs].

[27] P. Nakkiran, G. Kaplun, D. Kalimeris, T. Yang, B. L. Edelman, F. Zhang, and B. Barak. SGD on Neural Networks Learns Functions of Increasing Complexity, May 2019. arXiv:1905.11604 [cs, stat].

[28] J. Nam, H. Cha, S. Ahn, J. Lee, and J. Shin. Learning from Failure: De-biasing Classifier from Biased Classifier. In *Advances in Neural Information Processing Systems*, volume 33, pages 20673–20684. Curran Associates, Inc., 2020.

[29] R. O'Donnell. Analysis of Boolean Functions, May 2021. arXiv:2105.10386 [cs, math].

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, Dec. 2019. arXiv:1912.01703 [cs, stat].

[31] M. Pezeshki, S.-O. Kaba, Y. Bengio, A. Courville, D. Precup, and G. Lajoie. Gradient Starvation: A Learning Proclivity in Neural Networks, Nov. 2021. arXiv:2011.09468 [cs, math, stat].

[32] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville. On the Spectral Bias of Neural Networks, May 2019. arXiv:1806.08734 [cs, stat].

[33] E. Rosenfeld, P. Ravikumar, and A. Risteski. Domain-Adjusted Regression or: ERM May Already Learn Features Sufficient for Out-of-Distribution Generalization, Oct. 2022. arXiv:2202.06856 [cs].

[34] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization, Apr. 2020. arXiv:1911.08731 [cs, stat].

[35] S. Sagawa, A. Raghunathan, P. W. Koh, and P. Liang. An Investigation of Why Overparameterization Exacerbates Spurious Correlations, Aug. 2020. arXiv:2005.04345 [cs, stat].

[36] L. Scimeca, S. J. Oh, S. Chun, M. Poli, and S. Yun. Which Shortcut Cues Will DNNs Choose? A Study from the Parameter-Space Perspective, Feb. 2022. arXiv:2110.03095 [cs, stat].

[37] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli. The Pitfalls of Simplicity Bias in Neural Networks, Oct. 2020. arXiv:2006.07710 [cs, stat].

[38] D. Teney, E. Abbasnejad, S. Lucey, and A. v. d. Hengel. Evading the Simplicity Bias: Training a Diverse Set of Models Discovers Solutions with Superior OOD Generalization, Sept. 2022. arXiv:2105.05612 [cs].

[39] P. A. Utama, N. S. Moosavi, and I. Gurevych. Towards Debiasing NLU Models from Unknown Biases. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7597–7610, Online, Nov. 2020. Association for Computational Linguistics.

[40] G. Valle-Pérez, C. Q. Camargo, and A. A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions, Apr. 2019. arXiv:1805.08522 [cs, stat].

[41] Z. Wen and Y. Li. Toward Understanding the Feature Learning Process of Self-supervised Contrastive Learning, July 2021. arXiv:2105.15134 [cs, stat].

[42] A. Williams, N. Nangia, and S. R. Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference, Feb. 2018. arXiv:1704.05426 [cs] version: 4.

[43] K. Xiao, L. Engstrom, A. Ilyas, and A. Madry. Noise or Signal: The Role of Image Backgrounds in Object Recognition, June 2020. arXiv:2006.09994 [cs].

[44] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma. Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks. *Communications in Computational Physics*, 28(5):1746–1767, June 2020. arXiv:1901.06523 [cs, stat].

[45] Y. Xue, S. Joshi, E. Gan, P.-Y. Chen, and B. Mirzasoleiman. Which Features are Learnt by Contrastive Learning? On the Role of Simplicity Bias in Class Collapse and Feature Suppression, May 2023. arXiv:2305.16536 [cs, stat].

[46] Y. Yaghoobzadeh, S. Mehri, R. Tachet, T. J. Hazen, and A. Sordoni. Increasing Robustness to Spurious Correlations using Forgettable Examples, Feb. 2021. arXiv:1911.03861 [cs].

[47] Y. Yang, E. Gan, G. K. Dziugaite, and B. Mirzasoleiman. Identifying Spurious Biases Early in Training through the Lens of Simplicity Bias, May 2023. arXiv:2305.18761 [cs].

[48] M. Zhang, N. S. Sohoni, H. R. Zhang, C. Finn, and C. Ré. Correct-N-Contrast: A Contrastive Approach for Improving Robustness to Spurious Correlations, Mar. 2022. arXiv:2203.01517 [cs].

[49] C. Zhou, X. Ma, P. Michel, and G. Neubig. Examining and Combating Spurious Features under Distribution Shift, June 2021. arXiv:2106.07171 [cs].

A Appendix

A.1 Related Work

Our work touches upon several general aspects of deep learning and machine learning. However, the following is by no means an exhaustive list.

Learning with Spurious Features. Learning under spurious correlation can be interpreted as an Out-Of-Distribution (OOD) or group imbalance task, as spurious features divide the dataset into imbalanced groups. Two cases arise: (1) when the spurious attribute is given, popular methods like [34, 13] can be applied, (2) when the spurious label is unknown during training, various algorithms have been proposed to exploit the phenomenon of simplicity bias [40, 37, 27], which posits that spurious features are learned by the model in the early stages of learning, to upweight underrepresented groups. A representative method of this type is the "Just Train Twice Algorithm"[20], where a model is first trained to upweight "easy" samples. Another line of work focuses on underspecified tasks where the spurious features are fully correlated with the label [38, 19].

Simplicity of Spurious Features. The terms "spurious" and "shortcut" features, which are "easier" to learn than core features, [10] are often used interchangeably in deep learning literature. However,

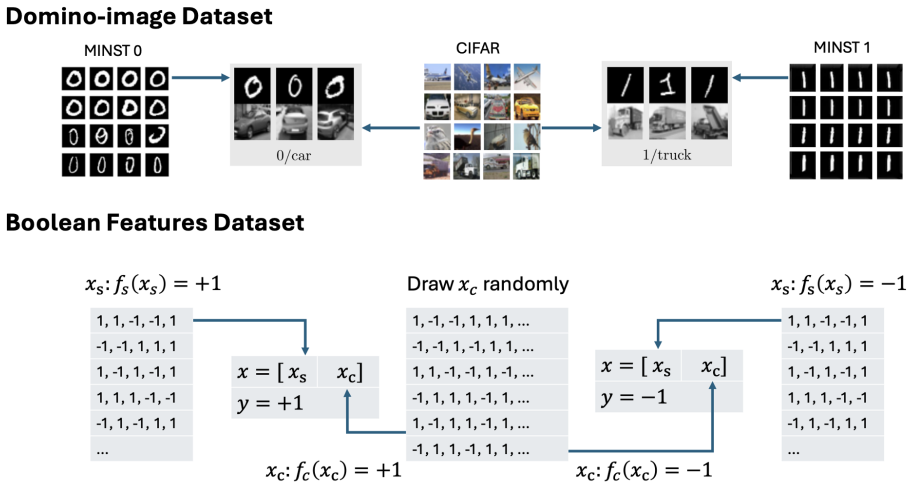


Figure 5: Comparison between our dataset $\mathcal{D}_{\text{same}}$ and the corresponding part of MNIST-CIFAR dataset. We use $f_s = \chi$ as an example.

the constructs of "simplicity" and "features" differ considerably across studies. For example, [37] examine features along different dimensions, quantifying simplicity by the number of linear segments needed for perfectly separating the data. [41, 47, 35] encapsulate both spurious and core features as 1-bit vectors, gauging simplicity by the amount or variance of noise applied to each feature. In a similar manner, [45] use different strength or magnitude of feature vector to encapsulate feature with different simplicity. [25] represent features as equally-sized vectors and assess simplicity through a model's predictive accuracy when relying on spurious features in a low-rank subspace. [31] focus on features as principal components of a model's neural tangent random feature and show an inclination for features with higher strength. [36] study the loss landscape in the presence of spurious features revealing that models randomly select features with varying likelihoods. [26] conjecture that prediction depth [5] is a suitable measure of feature difficulty. The closest setup to ours is binary feature dataset by [12] that consider binary linear spurious features with parity-like core features. Despite our framework bearing resemblance to previously proposed notions of simplicity, we distinguish ourselves by: (1) using non-linear features for spurious and core features, (2) focusing on the computational aspects of learning to characterize complexity rather than representational properties. (3) providing a closer resemblance to real data set through capturing the hierarchical learning aspect of neural network.

Datasets for Studying Spurious Correlations. Numerous datasets have been employed to study learning under spurious correlation. These include synthetic datasets such as WaterBird [34], Domino Image dataset [37], Color-MNIST [48], and a series of datasets proposed in [12]. It's important to note that these datasets are constructed in an ad-hoc manner, making it challenging to justify the complexity of the spurious features. Real datasets known to contain spurious correlations, such as CivilComments [8], MultiNLI[42], CelebA [22], and CXR [17], are also used to evaluate algorithms designed to mitigate shortcut features. A recent work [15] points out several problems of existing datasets that has been used to study spurious correlation and evaluating algorithm performances. Our observation provide further support for their claims (see C.2.2).

It is worth noting that almost all algorithms assume a balanced validation dataset for extensive hyperparameter tuning, as observed in [14]. A key work related to our research is [18, 14], where it is demonstrated that core features can be decoded from the last layer with state-of-the-art performance using validation data. This finding suggests that much of the work in this field focuses on optimizing classifier head weights. Retraining the last layer has also been explored widely and shown to be highly efficient in other settings, such as long-tail learning [16], probing inner representations of a model [3], and out-of-distribution learning [33].

Neural Feature Learning for Boolean functions. The problem of learning Boolean functions has long been a fundamental challenge in computational learning theory. A body of work has focused

on studying the mechanism of learning the parity function with neural networks in great detail [24, 7, 9, 6]. Another important class of functions, referred to as "staircase" functions by [1, 2], is explored in our study and used to construct features that resemble real-world features.

A.2 Boolean Function Analysis Background

See [29] for a comprehensive review for boolean function analysis. We only include the most important tools here.

Lemma 3. *Any boolean function $f : \{+1, -1\}^n \to \mathbb{R}$ can be decomposed into an orthogonal basis*

$$f(x) = \prod_{S\in[n]} \widehat{f}(S)\chi_S(x)$$

where $\chi_S(x) = \prod_{i\in S} x_i$ and note we have $E[\chi_S(x)\chi_{S'}(x)] = 0$ for all $S' \neq S$. Thus we have $\widehat{f}(S) = E[f(x)\chi_S(x)]$.

A.3 Properties of Boolean spurious distribution

Lemma 4. *Given a spurious boolean distribution defined previously, if either of the feature is unbiased. Then the distribution is equivalent to a mixture of $2(1-\lambda)\mathcal{D}_{\mathsf{unif}} + (2\lambda-1)\mathcal{D}_{\mathsf{same}}$.*

Proof. This is equivalent to saying that given any booleaбyn vector x, $P_D(x) = 2(1-\lambda)P_{\mathsf{unif}}(x) + (1-2\lambda)P_{\mathsf{same}}(x)$. Note that $P_D(x) = \lambda P_{\mathsf{same}}(x) + (1-\lambda)P_{\mathsf{diff}}(x) = (2\lambda-1)P_{\mathsf{same}}(x) + (1-\lambda)(P_{\mathsf{diff}}(x) + P_{\mathsf{same}}(x))$. Thus we only need to argue that $P_{\mathsf{diff}}(x) + P_{\mathsf{same}}(x) = 2P_{\mathsf{unif}}(x)$ forms a uniform distribution. And we have

$$\begin{aligned}
&P_{\mathsf{diff}}(x) + P_{\mathsf{same}}(x)\\
=&\frac{P_{x\sim U}(x=x, f_s(x)=f_c(x))}{P(f_s(x)=f_c(x))} + \frac{P_{x\sim U}(x=x, f_s(x)\neq f_c(x))}{P(f_s(x)\neq f_c(x))}\\
=&I[f_s(x)=f_c(x)]\frac{P_{x\sim U}(x=x)}{P(f_s(x)=f_c(x))} + I[f_s(x)\neq f_c(x)]\frac{P_{x\sim U}(x=x)}{P(f_s(x)\neq f_c(x))}\\
=&I[f_s(x)=f_c(x)]\frac{P_{x\sim U}(x=x)}{P(f_s(x)=1)P(f_c(x)=1)+P(f_s(x)=-1)P(f_c(x)=-1)}\\
+&I[f_s(x)\neq f_c(x)]\frac{P_{x\sim U}(x=x)}{P(f_s(x)=1)P(f_c(x)=-1)+P(f_s(x)=-1)P(f_c(x)=1)}\\
=&I[f_s(x)=f_c(x)]2P_{x\sim U}(x=x) + I[f_s(x)\neq f_c(x)]2P_{x\sim U}(x=x)\\
=&2P_{x\sim U}(x=x)
\end{aligned}$$

□

B Calculation of Gradient Gaps

B.1 Setting

Recall the definition of the boolean task we defined in the draft. We define 1. x the concatenation of three vectors x_s, x_c, x_u. The length of x_s is s and the length of x_c is c with $s << c$. x_u here denote a length u random boolean vector which does not have any correlation with the label. Thus the length of x is $n = s + c + u$ Without loss of generality, we additionally require c, s to be even length and u to be odd length. 2. f_s and f_c are parity functions defined as $\chi^S(x_s) = \prod_{i\in[s]} x_i$ and $\chi^C(x_c) = \prod_{i\in[c]}$. So $\deg(f_s) = s, \deg(f_c) = c$. 3. We then form a spurious distribution as defined in the main paper with confounder strength λ.

B.1.1 Model

We consider a 1 hidden layer ReLU neural network with r neurons

$$f(x) = \sum_{i=1}^{r} a_i\sigma(w_i^\top x + b_i)$$

where $a_i \in \mathbb{R}, w_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. We use hinge loss

$$\ell(y, \widehat{y}) = \max(0, 1 - y\widehat{y})$$

where $\widehat{y}$ is the output of the model and y is the true label.

B.1.2 Initialization

Let us consider the following initialization scheme as in [6]

1. For all $1 \le i \le r/2$, randomly initialize

   $w_i^{(0)} \sim \mathsf{Unif}(+1, -1^n), a_i^{(0)} \sim \mathsf{Unif}(+1, -1), b_i^{(0)} \sim \mathsf{Unif}(-1 + 1/k, -1 + 2/k, ..., 1 - 1/k)$
2. For all $r/2 < i \le r$, initialize

   $$w_i^{(0)} = w_{i-r/2}^{(0)}, a_i^{(0)} = -a_{i-r/2}^{(0)}, b_i^{(0)} = b_{i-r/2}^{(0)}$$

Key properties of this initialization scheme are: (1) It is unbiased, since the model output is 0 on all inputs at initialization, and (2) Biases b are set such that they enable computing parity linearly once we have the correct coordinates identified. For the informal lemma in the main paper, we assume w_i is the all 1s vector.

We will first analyze the gradients at initialization, then after spurious feature is learned.

B.2 Population Gradient Gap at Initialization

Notice that our initalization makes the model output 0 on all x. Thus $l(y, \widehat{y}) = 0$ and then we have $\nabla_{\widehat{y}} l'(y, \widehat{y}) = -y$.

We can now formulate the population gradient at initialization. Without loss of generality, we will assume $\lambda > 0.5$, then $\mathcal{D}_\lambda$ is a mixture such that w.p $2(1-\lambda)$ we draw a sample x from the uniform distribution $\mathsf{Unif}(\{+1, -1\}^n)$. And with $2\lambda - 1$, we draw a sample from $\mathcal{D}_{\mathsf{same}}$ where we first draw $x_c \sim \mathsf{Unif}(\{+1, -1\}^c)$ and then draw a $x_s \sim \mathsf{Unif}\{x_s | \chi_S(x_s) = \chi_C(x_c)\}$.

Then the population gradient for weight $w_{i,j}$ is

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}_\lambda}[\nabla_{w_{i,j}} l(f(x; \theta_0), y)] \\
&\quad = \mathbb{E}_{\mathcal{D}_\lambda}[-y \nabla_{w_{i,j}} f(x; \theta_0)] \\
&\quad = \mathbb{E}_{\mathcal{D}_\lambda}[-y a_i \mathbb{1}\{w_i^\top x + b_i > 0\} x_j] \\
&\quad = 2(1-\lambda)\mathbb{E}_{\mathcal{D}_{\mathsf{unif}}}[-y a_i \mathbb{1}\{w_i^\top x + b_i > 0\} x_j] + (2\lambda - 1)\mathbb{E}_{D_s}[-y a_i \mathbb{1}\{w_i^\top x + b_i > 0\} x_j]
\end{aligned} \tag{1}$$

We will study the two terms separately.

Population Gradient on uniform distribution. Set $g_{i,j} = \mathbb{E}_{\mathcal{D}_{\mathsf{unif}}}[-y a_i \mathbb{1}\{w_i^\top x + b_i > 0\} x_j]$. As long as $w_i \in \{-1, 1\}^n$, from [6], we have

1. For $j \in [c]$:

   $$g_{i,j} = -\frac{1}{2} a_i \xi_{c-1} \cdot \chi_{[c]\setminus\{j\}}(w_i)$$
2. For $j \in [s] \cup [u]$:

   $$g_{i,j} = -\frac{1}{2} a_i \xi_{c+1} \cdot \chi_{[c]\cup\{j\}}(w_i)$$

where $\xi_k = \widehat{\mathsf{Maj}}(S)$ with $|S| = k$. Thus we have for the first term $g_{i,j}^u = 2(1 - \lambda)\mathbb{E}_{\mathcal{D}_{\mathsf{unif}}}[-y a_i \mathbb{1}\{w_i^\top x + b_i > 0\} x_j]$

1. For $j \in [c]$:

   $$g_{i,j}^u = -(1-\lambda) a_i \xi_{c-1} \cdot \chi_{[c]\setminus\{j\}}(w_i)$$
2. For $j \in [s] \cup [u]$:

   $$g_{i,j}^u = -(1-\lambda) a_i \xi_{c+1} \cdot \chi_{[c]\cup\{j\}}(w_i)$$

Population Gradient on label aligned distribution $\mathcal{D}_{\mathsf{same}}$. Note for the second term of 1,

$$
\begin{aligned}
&(2\lambda-1)\mathbb{E}_{x_c,x_s\sim D_s}[-ya_i\mathbb{1}\{w_i^\top x+b_i>0\}x_j]\\
&=(2\lambda-1)([P_{x_c\sim u}[\chi(x_c)=-1]\mathbb{E}_{x_c,x_s\sim u}[-ya_i\mathbb{1}\{w_i^\top x+b_i>0\}x_j|\chi(x_c)=-1,\chi(x_s)=-1]\\
&\quad+P_{x_c\sim u}[\chi(x_c)=1]\mathbb{E}_{x_c,x_s\sim u}[-ya_i\mathbb{1}\{w_i^\top x+b_i>0\}x_j|\chi(x_c)=1,\chi(x_s)=1])\\
&=(2\lambda-1)(\frac{1}{2}\cdot 4\cdot\mathbb{E}_{x_c,x_s\sim u}[-ya_i\mathbb{1}\{w_i^\top x+b_i>0\}x_j\mathbb{1}\{\chi(x_c)=1,\chi(x_s)=1\}]\\
&\quad+\frac{1}{2}\cdot 4\cdot\mathbb{E}_{x_c,x_s\sim u}[-ya_i\mathbb{1}\{w_i^\top x+b_i>0\}x_j\mathbb{1}\{\chi(x_c)=1,\chi(x_s)=1\})\\
&=-2a_i(2\lambda-1)\cdot\mathbb{E}_{x_c,x_s\sim u}[yx_j\mathbb{1}\{w_i^\top x+b_i>0\}\mathbb{1}\{\chi(x_c)=\chi(x_s)\}] \qquad (2)
\end{aligned}
$$

By the initialization scheme, we have $w_i^\top x$ as an integer and $|b|<1$. Thus

$$\mathbb{1}\{w_i^\top x+b_i>0\}=\mathbb{1}\{w_i^\top x>0\}$$

We will first ignore the yx_j component inside the expectation and study the boolean function $q(x)=\mathbb{1}\{w_i^\top x>0\}\mathbb{1}\{\chi(x_c)=\chi(x_s)\}$. Observe that

$$\mathbb{1}\{\chi(x_c)=\chi(x_s)\}=\mathbb{1}\{\chi(x_c)\chi(x_s)=1\}=\mathbb{1}\{\chi_{[c]\cup[s]}(x)=1\}.$$

This gives us

$$
\begin{aligned}
q(x)&=\mathbb{1}\{w_i^\top x>0\}\mathbb{1}\{\chi_{[c]\cup[s]}(x)=1\}\\
&=\frac{1+\mathsf{Maj}_n(w_i\odot x)}{2}\cdot\frac{1+\chi_{[c]\cup[s]}(x)}{2}\\
&=\frac{1}{4}\cdot\big(1+\chi_{[c]\cup[s]}(x)+\mathsf{Maj}_n(w_i\odot x)+\chi_{[c]\cup[s]}(x)\mathsf{Maj}_n(w_i\odot x)\big) \qquad (3)
\end{aligned}
$$

We can study the fourier spectrum of each of the term in 3 to construct the fourier spectrum of q.

1. For $q_1(x)=\chi_{[c]\cup[s]}(x)$, notice that $\widehat{q}(S)=0$ for all $S\subset[n]$ except when $S=[c]\cup[s]$ where $\widehat{q}(S)=1$.
2. For $q_2(x)=\mathsf{Maj}(w\odot x)$, notice that $\mathsf{Maj}(x)$ can be written in its fourier expansion as $\mathsf{Maj}(x)=\sum_{S\subseteq[n]}\widehat{\mathsf{Maj}_n}(S)\chi_S(x)$. This gives us

$$q_2(x)=\mathsf{Maj}(w_i\odot x)=\sum_{S\subseteq[n]}\widehat{\mathsf{Maj}_n}(S)\chi_S(w_i\odot x)=\sum_{S\subseteq[n]}\widehat{\mathsf{Maj}_n}(S)\chi_S(x)\chi_S(w_i).$$

   Thus we have $\widehat{q_2}(S)=\chi_S(w_i)\widehat{\mathsf{Maj}_n}(S)$.
3. For $q_3(x)=\chi_{[c]\cup[s]}(x)\mathsf{Maj}_n(w_i\odot x)$, we have

$$
\begin{aligned}
\widehat{q_3}(S)&=\mathbb{E}[\mathsf{Maj}_n(w_i\odot x)\chi_S(x)\chi_{[c]\cup[s]}(x)]\\
&=\mathbb{E}_x[\mathsf{Maj}_n(w_i\odot x)\chi_{([s]\cup[c])\Delta S}(x)]\\
&\quad-\widehat{q_2}(([c]\cup[s])\Delta S)\\
&=\chi_{([c]\cup[s])\Delta S}(w_i)\widehat{\mathsf{Maj}_n}(([c]\cup[s])\Delta S)
\end{aligned}
$$

By the orthogonality and linearity of the fourier basis, we thus have

$$\widehat{q}(S)=\frac{1}{4}(\chi_S(w_i)\widehat{\mathsf{Maj}_n}(S)+\chi_{([c]\cup[s])\Delta S}(w_i)\widehat{\mathsf{Maj}_n}(([c]\cup[s])\Delta S))$$

for $|S|>0$ and $S\neq[s]\cup[c]$. Now let us put it back in 2.

1. For random index $j\in[u]$,

$$
\begin{aligned}
g_{i,j}^s&=-2a_i(2\lambda-1)\cdot\mathbb{E}_{x_c,x_s\sim u}[yx_j\mathbb{1}\{w_i^\top x+b_i>0\}\mathbb{1}\{\chi(x_c)=\chi(x_s)\}]\\
&=-2a_i(2\lambda-1)\cdot\mathbb{E}_{x\sim u}[\chi_{[c]\cup j}(x)q(x)]\\
&=-2a_i(2\lambda-1)\cdot\widehat{q}([c]\cup j)\\
&=-\frac{1}{2}a_i(2\lambda-1)\cdot(\chi_{[c]\cup\{j\}}(w_i)\xi_{c+1}+\chi_{[s]\cup\{j\}}(w_i)\xi_{s+1})
\end{aligned}
$$

2. For core index $j \in [c]$, in a similar manner, we have

$$\begin{aligned} & g_{i,j}^s \\ &= -2a_i(2\lambda-1)\cdot \mathbb{E}_{x_c,x_s\sim u}[yx_j \mathbb{1}\{w_i^\top x + b_i > 0\}\mathbb{1}\{\chi(x_c)=\chi(x_s)\}] \\ &= -2a_i(2\lambda-1)\cdot \mathbb{E}_{x\sim u}[\chi_{[c]\setminus j}(x)q(x)] \\ &= -2a_i(2\lambda-1)\cdot \widehat{q}([c]\setminus j) \\ &= -\frac{1}{2}a_i(2\lambda-1)\cdot(\chi_{[c]\setminus\{j\}}(w_i)\xi_{c-1} + \chi_{[s]\cup\{j\}}(w_i)\xi_{s+1}) \end{aligned}$$

3. For spurious index $j \in [s]$, in a similar manner, we have

$$\begin{aligned} g_{i,j}^s &= -2a_i(2\lambda-1)\cdot \mathbb{E}_{x_c,x_s\sim u}[yx_j \mathbb{1}\{w_i^\top x + b_i > 0\}\mathbb{1}\{\chi(x_c)=\chi(x_s)\}] \\ &= -2a_i(2\lambda-1)\cdot \mathbb{E}_{x\sim u}[\chi_{[s]\setminus j}(x)q(x)] \\ &= -2a_i(2\lambda-1)\cdot \widehat{q}([s]\setminus j) \\ &= -\frac{1}{2}a_i(2\lambda-1)\cdot(\chi_{[s]\setminus\{j\}}(w_i)\xi_{s-1} + \chi_{[c]\cup\{j\}}(w_i)\xi_{c+1}) \end{aligned}$$

Putting it together. We summarize the final population gradient on each type of index.

1. For random index $j \in [u]$,

$$\begin{aligned} g_{i,j} &= g_{i,j}^u + g_{i,j}^s \\ &= -(1-\lambda)a_i\xi_{c+1}\cdot\chi_{[c]\cup\{j\}}(w_i) - \frac{1}{2}a_i(2\lambda-1)\cdot(\chi_{[c]\cup\{j\}}(w_i)\xi_{c+1} + \chi_{[s]\cup\{j\}}(w_i)\xi_{s+1}) \\ &= -a_i\left(\frac{1}{2}\xi_{c+1}\cdot\chi_{[c]\cup\{j\}}(w_i) + \left(\lambda-\frac{1}{2}\right)\chi_{[s]\cup\{j\}}(w_i)\xi_{s+1}\right) \end{aligned}$$

2. For core index $j \in [c]$, in a similar manner, we have

$$\begin{aligned} g_{i,j} &= g_{i,j}^u + g_{i,j}^s \\ &= -(1-\lambda)a_i\xi_{c-1}\cdot\chi_{[c]\setminus\{j\}}(w_i) - \frac{1}{2}a_i(2\lambda-1)\cdot(\chi_{[c]\setminus\{j\}}(w_i)\xi_{c-1} + \chi_{[s]\cup\{j\}}(w_i)\xi_{s+1}) \\ &= -a_i\left(\frac{1}{2}\xi_{c-1}\cdot\chi_{[c]\setminus\{j\}}(w_i) + \left(\lambda-\frac{1}{2}\right)\chi_{[s]\cup\{j\}}(w_i)\xi_{s+1}\right) \end{aligned}$$

3. For spurious index $j \in [s]$, in a similar manner, we have

$$\begin{aligned} g_{i,j} &= g_{i,j}^u + g_{i,j}^s \\ &= -(1-\lambda)a_i\xi_{c+1}\cdot\chi_{[c]\cup\{j\}}(w_i) - \frac{1}{2}a_i(2\lambda-1)\cdot(\chi_{[s]\setminus\{j\}}(w_i)\xi_{s-1} + \chi_{[c]\cup\{j\}}(w_i)\xi_{c+1}) \\ &= -a_i\left(\frac{1}{2}\xi_{c+1}\cdot\chi_{[c]\cup\{j\}}(w_i) + \left(\lambda-\frac{1}{2}\right)\chi_{[s]\setminus\{j\}}(w_i)\xi_{s-1}\right) \end{aligned}$$

B.3 Population Gradient after spurious feature is learned

Proof. Following the same procedure we have

$$\begin{aligned} q(x) &= \mathbb{1}\{w_i^\top x > 0\}\mathbb{1}\{\chi_{[c]\cup[s]}(x) \neq 1\} \\ &= \frac{1+\mathsf{Maj}_n(w_i\odot x)}{2}\cdot\frac{1-\chi_{[c]\cup[s]}(x)}{2} \\ &= \frac{1}{4}\cdot\big(1-\chi_{[c]\cup[s]}(x) + \mathsf{Maj}_n(w_i\odot x) - \chi_{[c]\cup[s]}(x)\mathsf{Maj}_n(w_i\odot x)\big) \end{aligned}$$

Thus

$$\widehat{q}(S) = \frac{1}{4}(\chi_S(w_i)\widehat{\mathsf{Maj}_n}(S) - \chi_{([c]\cup[s])\Delta S}(w_i)\widehat{\mathsf{Maj}_n}(([c]\cup[s])\Delta S))$$

for $|S| > 0$ and $S \neq [s]\cup[c]$.

Now following the same procedure as B.2, we have

1. For random index $j \in [u]$,

$$g_{i,j}^s = -\frac{1}{2} a_i (1-\lambda) \cdot (\chi_{[c]\cup\{j\}}(w_i)\xi_{c+1} - \chi_{[s]\cup\{j\}}(w_i)\xi_{s+1})$$

2. For core index $j \in [c]$, in a similar manner, we have

$$g_{i,j}^s = -\frac{1}{2} a_i (1-\lambda) \cdot (\chi_{[c]\setminus\{j\}}(w_i)\xi_{c-1} - \chi_{[s]\cup\{j\}}(w_i)\xi_{s+1})$$

3. For spurious index $j \in [s]$, in a similar manner, we have

$$g_{i,j}^s = -\frac{1}{2} a_i (1-\lambda) \cdot (\chi_{[c]\cup\{j\}}(w_i)\xi_{c+1} - \chi_{[s]\setminus\{j\}}(w_i)\xi_{s-1})$$

□

B.4 Limitations of the Justification

Towards getting an end-to-end result, we would need to address several challenges beyond the Fourier gap. Firstly, we did not discuss *The magnitude of Gradient gap between spurious and core.* In particular, the weights will go down faster for the spurious coordinates compared to how fast they will increase in the core coordinates. This would lead to an interesting dynamic of when the spurious coordinates are starting to be forgotten, there is an opposite effect to remember them due to the errors we will make on the spuriously correlated example. Understanding this balancing will be paramount to explaining why the spurious features magnitude does not decrease. Secondly, we do not understand why the spurious and core sub-networks remain separate. Our techniques treat different neurons very similarly, so it is not clear how to separate them to specialize for one type of features. We believe these directions are very interesting to prove, and would give additional insight into feature learning in neural networks which has garnered significant interest in the last few years.

C Experiments

C.1 Experiments Settings

Dataset Description. In addition to the Parity and Staircase datasets, we utilized the Domino dataset, which incorporates Binary CIFAR-10 as the core feature and Binary MNIST01, Binary MNIST79, and Binary FASHION as spurious features. The specific datasets and their objectives are:

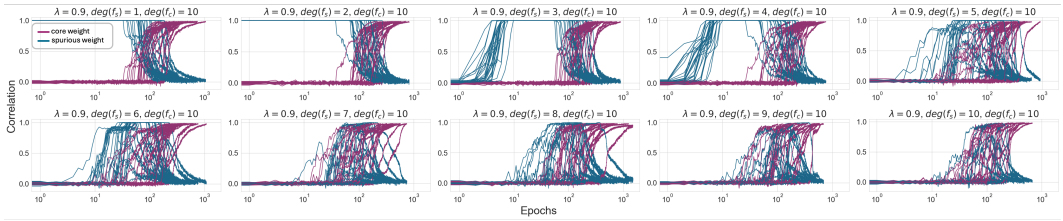
- Binary MNIST01: Differentiate between images representing numbers 0 and 1.
- Binary MNIST79: Differentiate between images representing numbers 7 and 9.
- Binary CIFAR-10: Distinguish between images representing a truck or an automobile.
- Binary Fashion MNIST: Distinguish between images representing a dress or a coat.

Furthermore, we incorporated the Waterbird dataset to capture real-world scenarios. Waterbirds encompasses a binary image classification challenge wherein the class represents the bird type (either landbird or waterbird). Notably, the background exhibits a spurious correlation with the class.

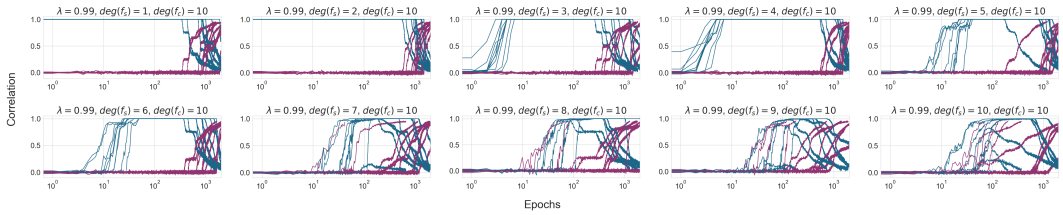
Model Details. Our observations remained consistent across a variety of architectures and hyperparameter combinations. For the Parity and Staircase datasets, we utilized a 1-layer MLP with ReLU activations and 100 neurons.

For the Domino and Waterbird datasets, we employed a ResNet-50-based architecture tailored for binary classification. The model was trained both from scratch and using `IMAGENET1K_V1` pre-trained weights.

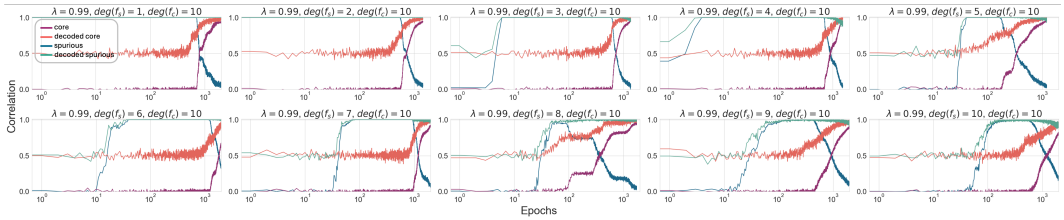
Training Procedure. Neural networks were initialized using a uniform distribution. Throughout our experiments, we opted for vanilla SGD with a momentum of 0.5, devoid of regularization. Both the MLPs and ResNets were trained with a batch size of 64. The learning rates were set to 0.0001 for MLPs and 0.001 for ResNets.



(a) $\lambda = 0.9$



(b) $\lambda = 0.99$



(c) $\lambda = 0.99$

Figure 6: **Online Parity Learning:** (a), (b) repeated experiments. (c) single experiment

C.2 Additional Experiments

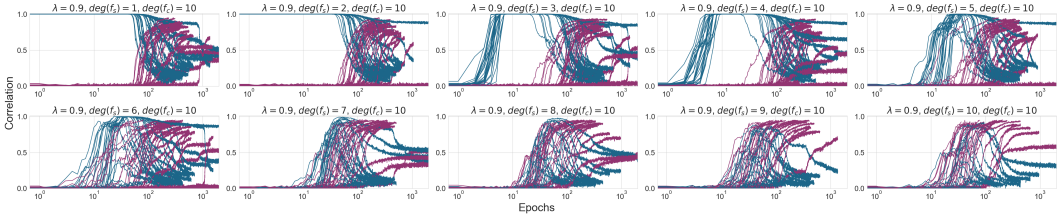
We divide this section into two sections to provide a comprehensive review of the influence of the two factors, complexity and confounder strength on learning either on a online setting learning or a finite dataset.

C.2.1 Complexity

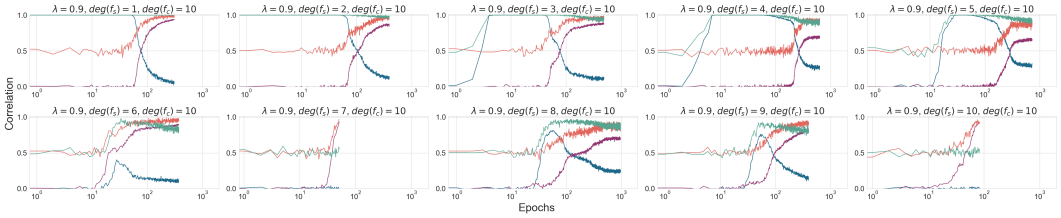
Parity Refer to Figures 6, 7. These figures illustrate the dynamic behavior of learning parity, showcasing considerable variability, especially when both λ and the complexity of the spurious function ($\deg(f_s)$) are elevated. In the context of learning parity with finite datasets, this phenomenon becomes even more pronounced, with numerous runs converging to a low core correlation value. For learning under finite dataset, it is worth highlighting that the ultimate performance of the network is heavily influenced by the randomness of initialization. Note here the total length of the feature vector is fixed to 20 so the computational complexity in learning core parity function stay fixed if $\lambda = 0.5$ for each case.

Staircase Refer to Figures 8 and 9. In the case of the staircase task, the influence of simpler spurious features on convergence slowdown becomes more obvious. The learning dynamics remain consistently stable across repeated runs in the staircase task. Therefore, our focus shifts to analyzing the dynamics of a single experiment, as it offers more informative insights.

Domino See Figure 10. We adhere to the convention of employing three image datasets as spurious features: MNIST-01, MNIST-79, and Fashion dress-coat, arranged in ascending order of difficulty. Meanwhile, our core feature is CIFAR-truck-automobile. It is crucial to note that the semi-real datasets utilized in spurious correlation research are inherently noisy, meaning that the model cannot learn the core feature perfectly or achieve 0 generalization error, as highlighted in [18]. Furthermore, these datasets are limited in size, with only 10,000 images available for CIFAR-truck-automobile.

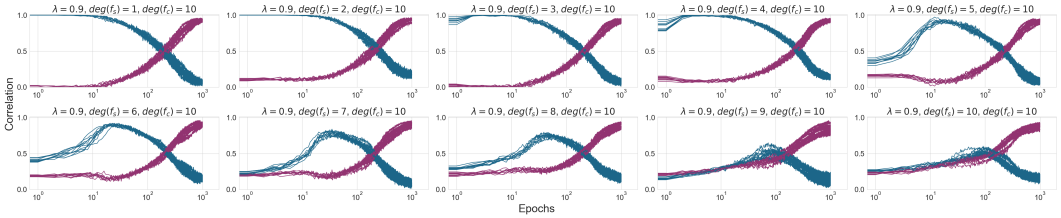


(a) $\lambda = 0.90$

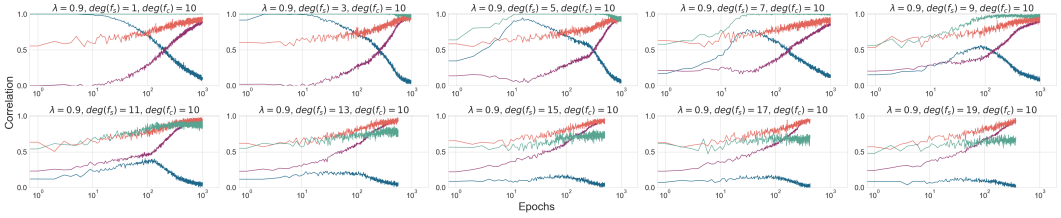


(b) $\lambda = 0.90$

Figure 7: **Finite Parity Learning with 20000 Sampled Points:** (a) repeated experiments. (b) single experiment



(a) $\lambda = 0.90$



(b) $\lambda = 0.90$

Figure 8: **Online Staircase:** (a) repeated experiments. (b) single experiment

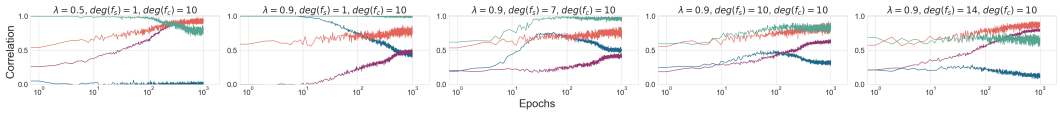


Figure 9: **Finite Staircase with 60000 Sampled Points:** $\lambda = 0.90$

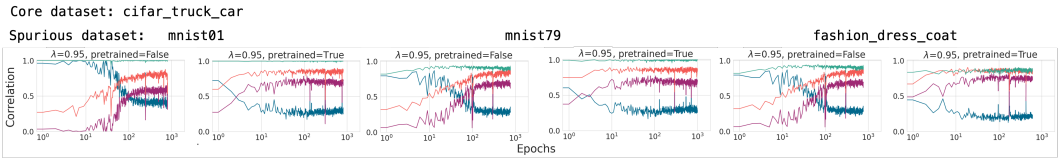
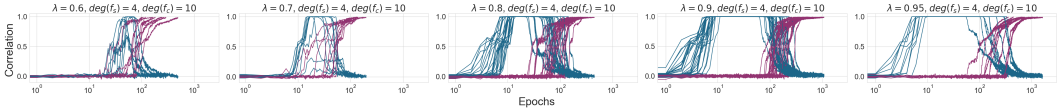
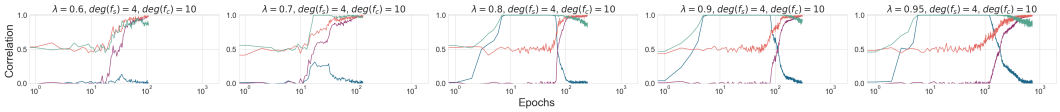


Figure 10: **Domino Dataset**



(a) $\deg(f_s) = 4$



(b) $\deg(f_s) = 4$

Figure 11: **Online Parity**: (a) repeated experiments. (b) single experiment

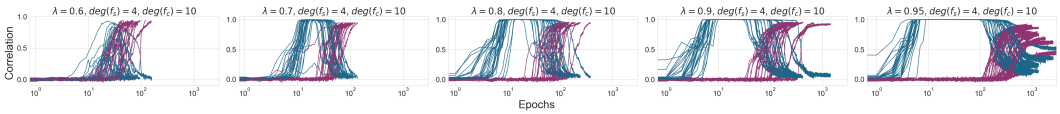
Previous studies have primarily focused on utilizing pretrained models to learn the spurious task. However, such an approach can obscure our understanding of feature learning dynamics, as pretrained models often achieve exceptionally high decoded core correlations from the outset as noted in [15]. Therefore, in our analysis, we present the learning dynamics for both pretrained and randomly initialized weights to provide a comprehensive perspective.

We see a interesting fact here is that pretrained model is robust to spurious feature as the end performance of the pretrained model is more insensitive to the presence of simpler spurious feature at higher λ.

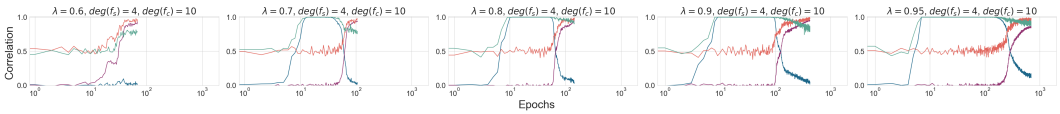
C.2.2 Confounder Strength

The impact of confounder strength on learning is more straight forward than the complexity. As confounder strength increases, the number of epochs needed for convergence also rises significantly. Notably, learning remains relatively insensitive to confounder strength until it reaches a threshold of 0.8, at which point we observe a notable increase in training epochs.The information of spurious feature i.e how well the spurious feature is memorized depends heavily on the confounder strength.

Parity For parity functions (see Figure 11), we see when confounder strength surpass 0.9, it converge much slower after the phase transition when compared to the experiment with lower λ. The slower convergence reflect on learning under finite dataset where the end performance of the model is impaired (see Figure 12).

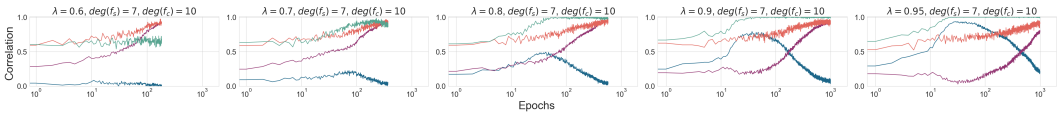


(a) $\deg(f_s) = 4$

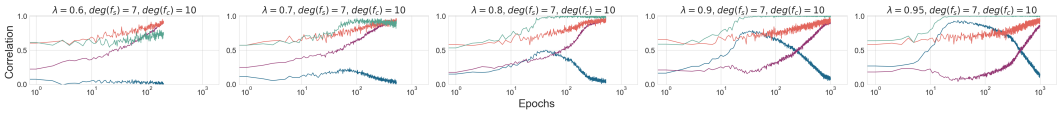


(b) $\deg(f_s) = 4$

Figure 12: **Finite Parity with 40000 Sampled Points**: (a)repeated experiments. (b) single experiment

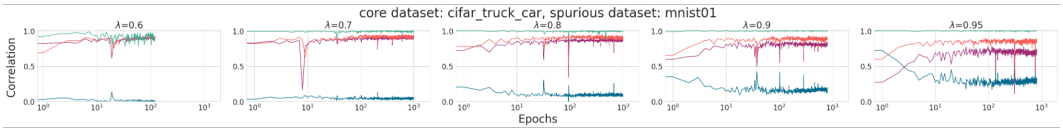


(a) Online: $\deg(f_s) = 4$

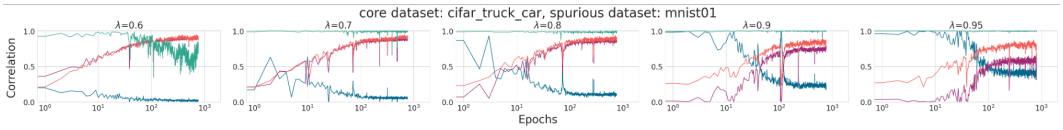


(b) Finite 120000 Sampled Points: $\deg(f_s) = 4$

Figure 13: **Staircase**



(a) Domino-Image: Pretrained Weights



(b) Domino-Image: Random Weights

Figure 14: Domino-Image

Staircase At higher confounder strength, the model has higher correlation to the spurious, simpler staircase function at the early stage of learning, which would also imply the spurious staircase function is memorized better by the model. We see higher λ cause harm to the end performance under finite dataset just as parity (see Figure 13).

Spurious Staircase can break algorithms that depends on early learning. Spurious Staircase can disrupt algorithms [20, 21, 39, 28, 46] reliant on early learning. In our finite staircase experiment, instances emerge where the model maintains nearly equal correlations with both spurious and core features throughout the learning process. Remarkably, even in these scenarios, spurious correlations have a detrimental impact on the final performance, as depicted in Figure 9. Algorithms that rely on early spurious feature learning assume a training phase where spurious correlations significantly outweigh core correlations. However, this assumption may not hold in many cases, posing a challenge for these algorithms in distinguishing samples influenced by the spurious feature from the majority or minority groups. Consequently, we address a critical question raised in [20]: when do these algorithms succeed, and when do they fail? We suspect that these algorithms may only perform well when the spurious feature is considerably simpler than the core feature or when the spurious correlation is exceptionally high.

Domino-Image, WaterBirds Refer to Figure 14. Surprisingly, our observations indicate that the pretrained model exhibits not only insensitivity to spurious features across a spectrum of complexities but also a remarkable resistance to higher λ values. Additionally, when compared to the initialization with random weights, models with pretrained weights consistently maintain low spurious correlations throughout the training process.

Regarding the waterbirds dataset, it is noteworthy that initialization with random weights fails to learn the core feature entirely, as reported in [18, 15]. Due to the absence of a controlled experiment for direct comparison, we leave the interpretation of the experimental results open to further investigation.

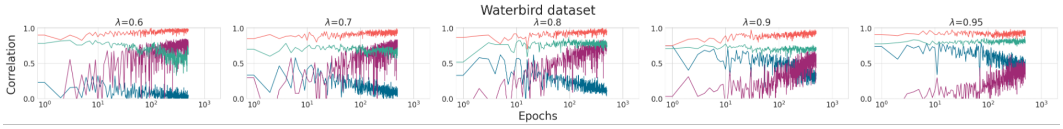


Figure 15: Waterbirds: Pretrained Weights

C.3 Core and Spurious Neurons

Refer to 16, 17. We show the dynamics of a random batch of spurious neurons and core neurons for both the parity and staircase spurious learning task. It can be seen that spurious neurons have higher weights on spurious coordinates throughout training. And core neurons which has significant weights on the core coordinates are specifically the neurons which does not have spurious weight spike at the start when the spurious feature is learned. We have also observed that the number of spurious neurons (see table 2) seems to be correlated with the convergence rate of core feature learning as shown in 1.

C.4 Quantitative evaluation of popular debiasing algorithms

We provide an overview of two algorithms, JTT [20] and SPARE [47]. Both operate under the assumption that the spurious or group attribute is not provided. They start by training a neural network on the dataset and apply early stopping. The goal is to use this model to identify the minority group that is induced by the spurious feature through exploiting simplicity bias of the model by doing early stopping. In the second stage, they treat the problem as one of class imbalance, employing strategies such as reweighting or resampling. The key difference between JTT and SPARE lies in their methods for identifying the minority group: JTT considers instances that the model predicts incorrectly as the minority, whereas SPARE uses clustering on the model's output representations.

These algorithms share a couple of significant drawbacks. The first is the challenge of identifying the optimal moment for early stopping of the model in the initial stage. The second is their reliance on the assumption that there is a distinct separation between the learning processes of spurious and core features, which is not always a given.

Our experiments employed the SpuCo library [15]. At each epoch, we paused the initial model's training to perform group inference, following which we calculated the Jaccard score to measure the accuracy of the inferred minority group against the actual minority group.

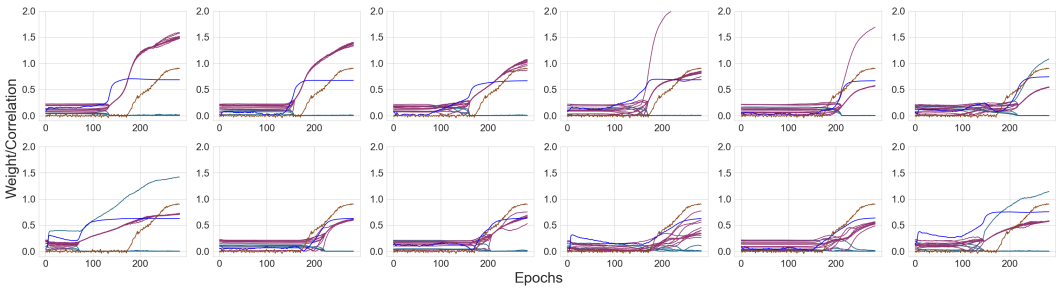
λ	# of Spurious Neurons	
	Parity	Staircase
0.5	0	0
0.6	0	0
0.7	8	2
0.8	10	10
0.9	20	16
0.94	14	16
0.98	18	27
1	24	26

$deg(f_s)$	# of Spurious Neurons	
	Parity	Staircase
1	8	4
3	15	8
5	21	15
7	15	19
9	0	32
11	0	27
13	0	18
15	0	18

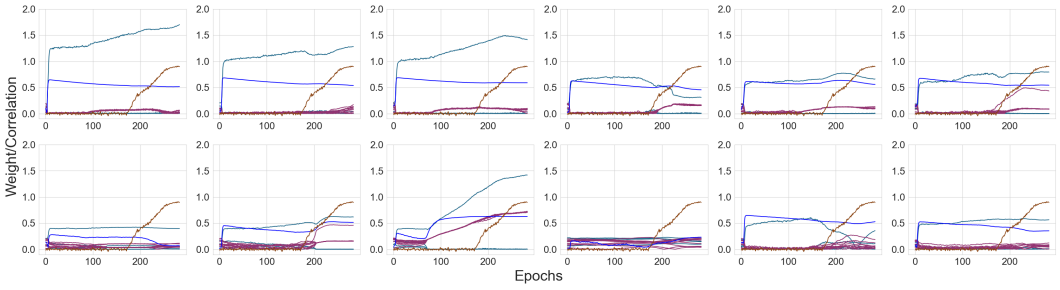
Table 2: Number of Spurious Neurons: The default setting is Parity:$deg(f_s) = 4, deg(f_c) = 10, \lambda = 0.9$ Staircase:$deg(f_s) = 7, deg(f_c) = 14, \lambda = 0.9$. We vary one of the parameters with other parameter fixed for each experiment.

C.5 Implementation, Hardware, and Computation Time

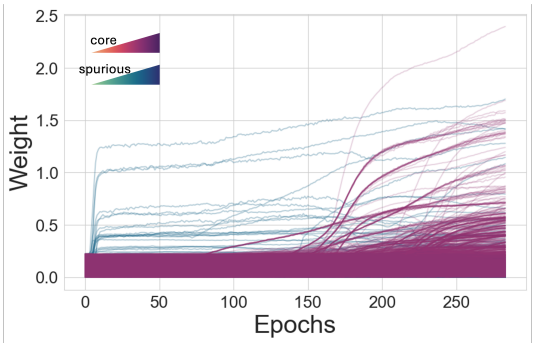
All training experiments were conducted using PyTorch[30]. While the majority of networks evaluated in our primary empirical findings are relatively compact, we trained a substantial number of models to validate the breadth of the "robust space" outcomes. These experiments utilized NVIDIA T4 and Quadro RTX 8000 GPUs, cumulatively consuming around 2,500 GPU hours.



(a) Parity: Core Neurons at $\lambda = 0.95$

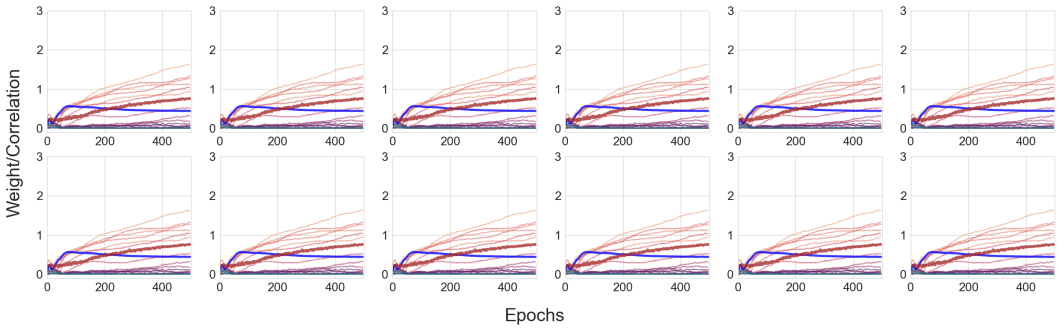


(b) Parity: Spurious Neurons at $\lambda = 0.95$

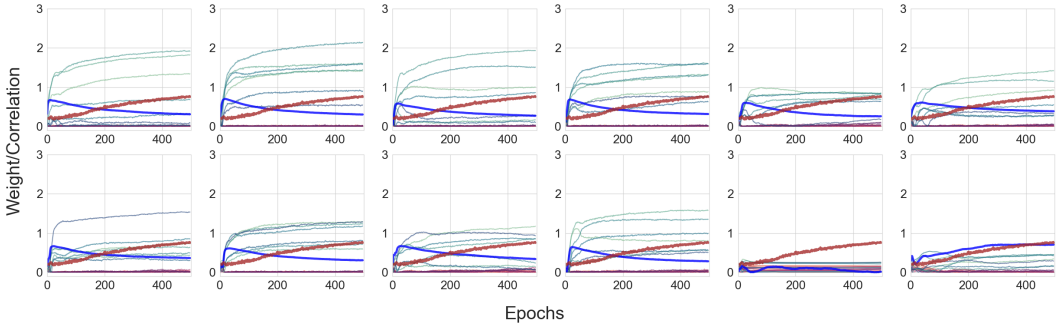


(c) Parity: All first layer neurons at $\lambda = 0.95$

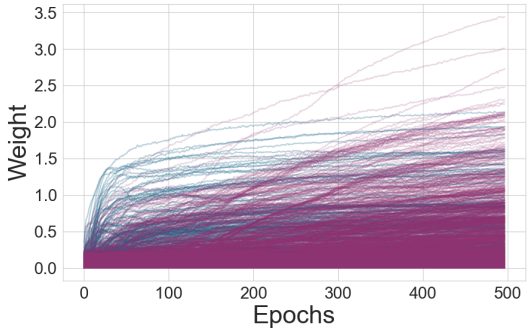
Figure 16: Dynamics of neurons on Parity task



(a) Staircase: Core Neurons at $\lambda = 0.90$



(b) Staircase: Spurious Neurons at $\lambda = 0.90$



(c) Staircase: All neurons at $\lambda = 0.90$

Figure 17: Dynamics of neurons on Staircase task