HiLD 2024: 2nd Workshop on High-dimensional Learning Dynamics

When Are Bias-Free ReLU Networks Like Linear Networks?

Yedi Zhang YEDI@GATSBY.UCL.AC.UK
Gatsby Unit, University College London

Andrew Saxe A.SAXE@UCL.AC.UK
Gatsby Unit & SWC, University College London

Peter E. Latham PEL@GATSBY.UCL.AC.UK
Gatsby Unit, University College London

Abstract

We investigate the expressivity and learning dynamics of bias-free ReLU networks. We firstly show that two-layer bias-free ReLU networks have limited expressivity: the only odd function two-layer bias-free ReLU networks can express is a linear one. We then show that, under symmetry conditions on the data, these networks have the same learning dynamics as linear networks. This allows us to give closed-form time-course solutions to certain two-layer bias-free ReLU networks, which has not been done for nonlinear networks outside the lazy learning regime. While deep bias-free ReLU networks are more expressive than their two-layer counterparts, they still share a number of similarities with deep linear networks. These similarities enable us to leverage insights from linear networks, leading to a novel understanding of bias-free ReLU networks. Overall, our results show that some properties established for bias-free ReLU networks arise due to equivalence to linear networks, and suggest that including bias or considering asymmetric data are avenues to engage with nonlinear behaviors.

1. Introduction

Theorists make simplifications to real-world models because simplified models are mathematically more tractable, yet discoveries made in them may hold in general. For instance, linear models have illuminated double descent [1] and benign overfitting [7] in practical neural networks. In this paradigm, understanding the consequences of a simplification is critical, since it informs us which discoveries in simple models extend to complex ones. Here we consider a specific simplification that is common in theoretical work on ReLU networks [4, 15, 28, 43, 50]: the removal of the bias terms. We investigate how removing the bias affects the expressivity and the learning dynamics of ReLU networks and identify scenarios where bias-free ReLU networks are like linear networks.

For expressivity, we show that two-layer bias-free (leaky) ReLU networks cannot express odd functions except linear functions. This was proven for input uniformly distributed on a sphere [8], but we prove it for arbitrary input and use a simpler approach. We then consider deep bias-free (leaky) ReLU networks and show a depth separation result. For learning dynamics, we show that two-layer bias-free (leaky) ReLU networks have the same learning dynamics as linear networks when trained on symmetric datasets with square loss or logistic loss starting from small initialization. Our symmetry condition on the dataset incorporates several previous results as special cases [29, 37]. Finally, we empirically find that deep bias-free ReLU networks form low-rank weights similar to those in deep linear networks when the target function is linear.

By revealing regimes in which bias-free ReLU networks are like linear networks, we offer a novel perspective on bias-free ReLU networks. This perspective draws from linear networks, which enjoy much richer theoretical understanding [3, 6, 20, 39, 40] than ReLU networks. In particular, we are able to give exact time-course solutions to certain two-layer ReLU networks in closed form, which has never been done for nonlinear networks outside the lazy learning regime. Our findings suggest that the bias terms in the network and the structures in the data play an essential role in learning nonlinear tasks with ReLU networks.

Our contributions are the following: (i) Section 3 proves the limited expressivity of bias-free (leaky) ReLU networks, and shows a depth separation result between two-layer and deep bias-free ReLU networks; (ii) Section 4.1 gives general conditions for when the two-layer bias-free (leaky) ReLU networks evolve like a linear network throughout training, shows why this equivalence occurs for both square loss and logistic loss, and gives closed-form time-course solutions to ReLU networks in this regime; (iii) Section 4.2 reports the similarities between deep bias-free ReLU networks and deep linear networks, and finds a low-rank structure in the weights.

1.1. Related Work

Prior work has reported that two-layer bias-free ReLU networks behave like linear networks in certain settings. We discuss two closest works here and provide additional related work in Appendix A.

Basri et al. [8] proved, using harmonic analysis, that two-layer bias-free ReLU networks can neither learn nor express odd nonlinear functions when input is uniformly distributed on a sphere. We show a more general result with a simpler proof. Our Theorem 1 handles arbitrary input, includes both ReLU and leaky ReLU networks, and the proof only involves rewriting the (leaky) ReLU activation function as the sum of a linear function and an absolute value function.

Lyu et al. [29] proved two-layer bias-free leaky ReLU networks trained with logistic loss converge to a linear, max-margin classifier on linearly separable tasks with a data augmentation procedure. The learning dynamics they studied is the same as that of a linear network, although this equivalence was not mentioned in the paper. We relax the assumption on the task from being linearly separable to being odd. And we show that two-layer bias-free ReLU and leaky ReLU networks evolve like linear networks on these datasets with square or logistic loss. We thus offer an additional perspective: convergence to the max-margin solution is not a speciality of leaky ReLU networks, but a property of linear networks, which extends to ReLU networks when they evolve like linear networks. The equivalence between ReLU and linear networks can be undesirable when the target model is nonlinear. Furthermore, we identify a practical challenge: the data augmentation procedure of [29] can cause the ReLU network to fail in learning a linearly non-separable task — a task the network might have succeeded on without the data augmentation.

2. Preliminaries

Notation. We use bold symbols to denote vectors and matrices. Double-pipe brackets $\|\cdot\|$ denote the L2 norm of a vector or the Frobenius norm of a matrix. Angle brackets $\langle\cdot\rangle$ denote taking the average over the dataset.

A two-layer bias-free (Leaky) ReLU network with H hidden neurons is defined as

$$f(\boldsymbol{x};\boldsymbol{W})=\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})=\sum_{h=1}^{H}w_{2h}\sigma(\boldsymbol{w}_{1h}\boldsymbol{x}),\quad \text{where } \sigma(z)=\max(z,\alpha z),\alpha\in[0,1]. \tag{1}$$

Here $\boldsymbol{x} \in \mathbb{R}^D$ is the input, $\boldsymbol{W}_1 \in \mathbb{R}^{H \times D}$ is the first-layer weight, $\boldsymbol{W}_2 \in \mathbb{R}^{1 \times H}$ is the second-layer weight, and $\boldsymbol{W}$ denotes weights of both layers collectively. It is a linear network when $\alpha = 1$ and can be written as $f(\boldsymbol{x}) = \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x}$. We also denote the linear network as $f^{\text{lin}}\left(\boldsymbol{x}; \boldsymbol{W}^{\text{lin}}\right) = \boldsymbol{W}_2^{\text{lin}} \boldsymbol{W}_1^{\text{lin}} \boldsymbol{x}$ when we need to distinguish it from (leaky) ReLU networks.

A deep bias-free network of depth L is $f(\boldsymbol{x}) = \boldsymbol{h}_L$ where $\boldsymbol{h}_L$ is recursively defined as $\boldsymbol{h}_l = \boldsymbol{W}_l \sigma(\boldsymbol{h}_{l-1}), 2 \leq l \leq L$, and $\boldsymbol{h}_1 = \boldsymbol{W}_1 \boldsymbol{x}$.

We consider the rich regime [48] in which the network is initialized with small random weights. The network is trained with full-batch gradient flow on a dataset $\{\boldsymbol{x}_\mu, y_\mu\}_{\mu=1}^P$ consisting of P samples. We study the square loss $\mathcal{L} = \left\langle (y - f(\boldsymbol{x}))^2 \right\rangle /2$ and the logistic loss $\mathcal{L}_{\text{LG}} = \left\langle \ln\left(1 + e^{y f(\boldsymbol{x})}\right) \right\rangle$. The gradient flow dynamics are given in Appendix C.

3. Network Expressivity

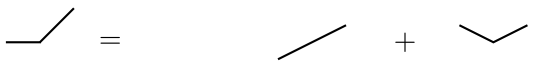
We first examine the expressivity of bias-free ReLU networks. Although standard ReLU networks are universal approximators [23, 36], bias-free ReLU networks are not since they can only express positively homogeneous functions, i.e., $g(a\boldsymbol{x}) = a g(\boldsymbol{x}) \, \forall a > 0$. Moreover, Section 3.1 shows that two-layer bias-free ReLU networks cannot express any odd function except linear functions. Section 3.2 shows that deep bias-free ReLU networks are more expressive than two-layer ones, but are still limited to positively homogeneous functions.

3.1. Two-Layer Bias-Free (Leaky) ReLU Networks

Theorem 1 *Two-layer bias-free (leaky) ReLU networks can only express a linear function plus a positively homogeneous even function.*

Proof . An arbitrary two-layer (leaky) ReLU network can be written as

$$\sum_{h=1}^{H} w_{2h} \sigma(\boldsymbol{w}_{1h} \boldsymbol{x}) = \sum_{h=1}^{H} w_{2h} \left[\frac{1+\alpha}{2} \boldsymbol{w}_{1h} \boldsymbol{x} + \frac{1-\alpha}{2} |\boldsymbol{w}_{1h} \boldsymbol{x}| \right], \tag{2}$$



which is a linear function plus a positively homogeneous even function. ■

Corollary 2 *The only odd function that bias-free two-layer (leaky) ReLU networks can express is the linear function.*

3.2. Deep Bias-Free (Leaky) ReLU Networks

Similarly to two-layer bias-free ReLU networks, deep bias-free ReLU networks can express only positively homogeneous functions. However, in contrast to two-layer bias-free ReLU networks, deep bias-free ReLU networks can express some odd nonlinear functions. For instance, for two-dimensional input $\boldsymbol{x} = [x_1, x_2]^\top$, the function below is odd, nonlinear, and can be implemented by a three-layer bias-free ReLU network,

$$g(\boldsymbol{x}) = \sigma(\sigma(x_1) - \sigma(x_2)) - \sigma(\sigma(-x_1) - \sigma(-x_2)), \quad \text{where } \sigma(z) = \max(z, 0). \tag{3}$$

Thus, we have a depth separation result for bias-free ReLU networks: there exist odd nonlinear functions, such as $g(\boldsymbol{x})$ above, that two-layer bias-free ReLU networks cannot express but deep bias-free ReLU networks can.

4. Learning Dynamics

4.1. Two-Layer Bias-Free (Leaky) ReLU Networks

Section 3.1 has proved that two-layer bias-free (Leaky) ReLU networks cannot express odd functions except linear functions. We now show that under the Condition 3 on the dataset, two-layer bias-free (Leaky) ReLU networks not only find a linear solution but also have the same learning dynamics as a two-layer linear network.

Condition 3 *The dataset satisfies the following two symmetry conditions:*

1. *The empirical input data distribution is even:* $p(\boldsymbol{x}) = p(-\boldsymbol{x})$.
2. *The target model is odd:* $y(\boldsymbol{x}) = -y(-\boldsymbol{x})$.

Under Condition 3, the dynamics of two-layer bias-free (leaky) ReLU networks and linear networks initialized with small weights can both be approximated by a linear differential equation in the early phase of learning. Thus weights in the ReLU network form the same rank-one structure as weights in a linear network, as specified in Theorem 13. At the end of the early phase, the weights are rank-one with small errors. To simplify the analysis, Assumption 4 assumes the weights are exactly rank-one, which is justified by the initialization approaching 0 and the width approaching infinity. Simulations in Figure 1(*b*) also show that errors are indeed small in the finite case.

Assumption 4 *The two-layer bias-free network's weights satisfy* $\boldsymbol{W}_1 = \boldsymbol{W}_2^\top \boldsymbol{r}^\top$ *where* $\boldsymbol{r}$ *is an arbitrary unit vector. The second-layer weights* $\boldsymbol{W}_2$ *have equal L2 norms for their positive and negative elements.*

Under Condition 3 and Assumption 4, the learning dynamics of two-layer bias-free (leaky) ReLU networks reduces to Equation (34), which is the same as that of linear networks given in Equation (8) modulo scale factors. Thus, apart from the fact that learning is $(\alpha+1)/2$ times slower and the weights are $\sqrt{2/(\alpha+1)}$ times larger, the ReLU network has the same learning dynamics as its linear counterpart. We formally state this equivalence below.

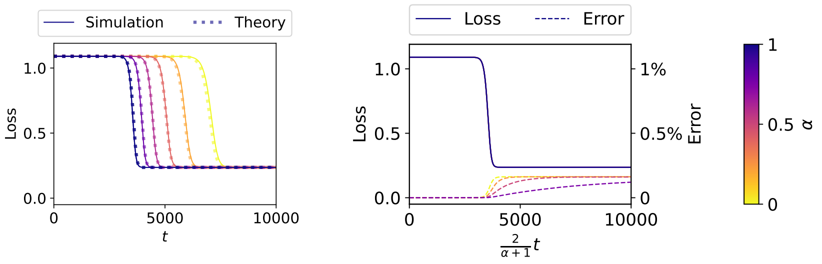
Theorem 5 *A two-layer (leaky) ReLU network and a linear network are trained with square or logistic loss starting from weights which differ by a scale factor,* $\boldsymbol{W}(0) = \sqrt{2/(\alpha+1)}\,\boldsymbol{W}^{\text{lin}}(0)$. *Under Condition 3 on the dataset and Assumption 4 on the weights at time* $t = 0$, *we have that* $\forall\, t \geq 0$, *Assumption 4 remains valid and:*

1. *The (leaky) ReLU network implements the same linear function as the linear network with scaled time*

$$f(\boldsymbol{x}; \boldsymbol{W}(t)) = f^{\text{lin}}\left(\boldsymbol{x}; \boldsymbol{W}^{\text{lin}}\left(\frac{\alpha+1}{2}t\right)\right). \tag{4}$$

2. *The weights in the (leaky) ReLU network are the same as scaled weights in the linear network*

$$\boldsymbol{W}(t) = \sqrt{\frac{2}{\alpha+1}}\boldsymbol{W}^{\text{lin}}\left(\frac{\alpha+1}{2}t\right). \tag{5}$$



(*a*) Loss. (*b*) Loss & error with rescaled time.

Figure 1: Two-layer bias-free (leaky) ReLU networks that evolve like a linear network. (a) Loss curves with different leaky ReLU parameter α (note $\alpha = 1$ is a linear network). The simulations match the theoretical solutions in Equation (38). The loss converges to global minimum, which is not zero due to the restricted expressivity of two-layer bias-free ReLU networks. (b) The simulated loss curves are plotted against a rescaled time axis; they collapse to one curve, demonstrating the networks are implementing the same linear function as in Equation (4). The error, defined as $\left\| \sqrt{\frac{\alpha+1}{2}} \boldsymbol{W} \left(\frac{2}{\alpha+1} t \right) - \boldsymbol{W}^{\text{lin}}(t) \right\| / \left\| \boldsymbol{W}^{\text{lin}}(t) \right\|$, is less than 1%, demonstrating that the weights in the (leaky) ReLU network are close to the weights in the linear network as in Equation (5). The errors are not exactly zero because the initialization is small but nonzero in the simulations. Hyperparameters in Appendix G.

We validate Theorem 5 and the plausibility of Assumption 4 with numerical simulations in Figure 1.

If the input covariance is white, we can further write down the exact time-course solution in closed form for two-layer bias-free (leaky) ReLU networks by adopting the solutions of linear networks [11, Theorem 3.1], as specified in Corollary 14.

Since the time evolution of two-layer bias-free (leaky) ReLU networks is the same as that of linear networks (modulo scale factors), their converged weights will also be the same. For learning with square loss, linear networks converge to the ordinary least squares solution [39]. For linearly separable binary classification with logistic loss, linear networks converge to the max-margin solution [41]. Thus two-layer bias-free (leaky) ReLU networks also converge to these solutions when they behave like linear networks; see Appendix E.4.

Corollary 6 *Under the same conditions as Theorem 5, the two-layer bias-free (leaky) ReLU network converges to a linear solution $f(\boldsymbol{x}; \boldsymbol{W}(\infty)) = \boldsymbol{w}^{*\top}\boldsymbol{x}$. For square loss, $\boldsymbol{w}^*$ is the ordinary least squares solution, $\boldsymbol{\Sigma}^{-1}\boldsymbol{\beta}$, which is the global minimum. For linearly separable binary classification with logistic loss, $\boldsymbol{w}^*$ is the max-margin (hard margin SVM) solution.*

4.2. Deep Bias-Free ReLU Networks

The equivalence between two-layer ReLU and linear networks stated in Theorem 5 does not extend to deep ReLU networks. Nonetheless, we find deep bias-free ReLU networks can form low-rank weights similar to those in deep linear networks when the empirical input distribution is even, the target function is linear, and the initialization is small. For the first and last layers, the linear and ReLU networks form rank-one weights of the same structure. For intermediate layers, weights of the linear network are rank-one, while weights of the ReLU network are rank-two and all weights are approximately non-negative as shown in Figure 3. Deep ReLU networks that have formed such low-rank weights will retain the low-rank structure and implement a linear map; see Appendix F.

Acknowledgement

The authors are grateful to Peter Orbanz, Spencer Frei, Ingo Steinwart, Rodrigo Carrasco-Davis, Lukas Braun, Clémentine Dominé, and William Dorrell for helpful discussions. The authors thank the following funding sources: Gatsby Charitable Foundation to YZ, AS, and PEL; Sainsbury Wellcome Centre Core Grant from Wellcome (219627/Z/19/Z) to AS; a Sir Henry Dale Fellowship from the Wellcome Trust and Royal Society (216386/Z/19/Z) to AS; Wellcome Trust (110114/Z/15/Z) to PEL.

References

[1] Madhu S. Advani, Andrew M. Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.08.022. URL https://www.sciencedirect.com/science/article/pii/S0893608020303117.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/allen-zhu19a.html.

[3] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 244–253. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/arora18a.html.

[4] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 322–332. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/arora19a.html.

[5] Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1NvflqAdoom.

[6] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90014-2. URL https://www.sciencedirect.com/science/article/pii/0893608089900142.

[7] Peter L. Bartlett, Philip M. Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020. doi: 10.1073/pnas.1907378117. URL https://www.pnas.org/doi/abs/10.1073/pnas.1907378117.

[8] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5ac8bb8a7d745102a978c5f8ccdb61b8-Paper.pdf.

[9] Etienne Boursier and Nicolas Flammarion. Early alignment in two-layer networks training is a two-edged sword. *arXiv preprint arXiv:2401.10791*, 2024.

[10] Etienne Boursier, Loucas Pillaud-Vivien, and Nicolas Flammarion. Gradient flow dynamics of shallow relu networks for square loss and orthogonal inputs. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20105–20118. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/7eeb9af3eb1f48e29c05e8dd3342b286-Paper-Conference.pdf.

[11] Lukas Braun, Clémentine Dominé, James Fitzgerald, and Andrew Saxe. Exact learning dynamics of deep linear networks with prior knowledge. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 6615–6629. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/2b3bb2c95195130977a51b3bb251c40a-Paper-Conference.pdf.

[12] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. SGD learns over-parameterized networks that provably generalize on linearly separable data. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJ33wwxRb.

[13] Dmitry Chistikov, Matthias Englert, and Ranko Lazic. Learning a neuron by a shallow relu network: Dynamics and implicit bias for correlated inputs. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 23748–23760. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4af24e6ce753c181e703f3f0be3b5e20-Paper-Conference.pdf.

[14] Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/fe131d7f5a6b38b23cc967316c13dae2-Paper.pdf.

[15] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eK3i09YQ.

[16] Spencer Frei, Niladri S. Chatterji, and Peter L. Bartlett. Random feature amplification: Feature learning and generalization in neural networks. *Journal of Machine Learning Research*, 24 (303):1–49, 2023. URL http://jmlr.org/papers/v24/22-1132.html.

[17] Spencer Frei, Gal Vardi, Peter Bartlett, and Nathan Srebro. Benign overfitting in linear classifiers and leaky relu networks from kkt conditions for margin maximization. In Gergely Neu and Lorenzo Rosasco, editors, *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pages 3173–3228. PMLR, 12–15 Jul 2023. URL https://proceedings.mlr.press/v195/frei23a.html.

[18] Spencer Frei, Gal Vardi, Peter Bartlett, Nathan Srebro, and Wei Hu. Implicit bias in leaky reLU networks trained on high-dimensional data. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JpbLyEI5EwW.

[19] Spencer Frei, Gal Vardi, Peter Bartlett, and Nati Srebro. The double-edged sword of implicit bias: Generalization vs. robustness in relu networks. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 8885–8897. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1c26c389d60ec419fd24b5fee5b35796-Paper-Conference.pdf.

[20] Kenji Fukumizu. Effect of batch learning in multilayer neural networks. *Gen*, 1(04):1E–03, 1998.

[21] T. H. Gronwall. Note on the derivatives with respect to a parameter of the solutions of a system of differential equations. *Annals of Mathematics*, 20(4):292–296, 1919. ISSN 0003486X. URL http://www.jstor.org/stable/1967124.

[22] David Holzmüller and Ingo Steinwart. Training two-layer relu networks with gradient descent is inconsistent. *Journal of Machine Learning Research*, 23(181):1–82, 2022. URL http://jmlr.org/papers/v23/20-830.html.

[23] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

[24] Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJflg30qKX.

[25] Yiwen Kou, Zixiang Chen, and Quanquan Gu. Implicit bias of gradient descent for two-layer relu and leaky relu networks on nearly-orthogonal data. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 30167–30221. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/602f5c1b803c53b2aaf0b3864bf3383a-Paper-Conference.pdf.

[26] Thien Le and Stefanie Jegelka. Training invariances and the low-rank phenomenon: beyond linear networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=XEW8CQgArno.

[27] Tao Luo, Zhi-Qin John Xu, Zheng Ma, and Yaoyu Zhang. Phase diagram for two-layer relu neural networks at infinite-width limit. *Journal of Machine Learning Research*, 22(71):1–47, 2021. URL http://jmlr.org/papers/v22/20-1123.html.

[28] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeLIgBKPS.

[29] Kaifeng Lyu, Zhiyuan Li, Runzhe Wang, and Sanjeev Arora. Gradient descent on two-layer nets: Margin maximization and simplicity bias. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12978–12991. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/6c351da15b5e8a743a21ee96a86e25df-Paper.pdf.

[30] Hartmut Maennel, Olivier Bousquet, and Sylvain Gelly. Gradient descent quantizes relu network features. *arXiv preprint arXiv:1803.08367*, 2018.

[31] Xuran Meng, Difan Zou, and Yuan Cao. Benign overfitting in two-layer relu convolutional neural networks for xor data. *arXiv preprint arXiv:2310.01975*, 2023.

[32] Hancheng Min, Enrique Mallada, and Rene Vidal. Early neuron alignment in two-layer relu networks with small initialization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=QibPzdVrRu.

[33] Leonardo Petrini, Francesco Cagnetta, Eric Vanden-Eijnden, and Matthieu Wyart. Learning sparse features can lead to overfitting in neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 9403–9416. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3d3a9e085540c65dd3e5731361f9320e-Paper-Conference.pdf.

[34] Leonardo Petrini, Francesco Cagnetta, Eric Vanden-Eijnden, and Matthieu Wyart. Learning sparse features can lead to overfitting in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2023(11):114003, nov 2023. doi: 10.1088/1742-5468/ad01b9. URL https://dx.doi.org/10.1088/1742-5468/ad01b9.

[35] Mary Phuong and Christoph H Lampert. The inductive bias of relu networks on orthogonally separable data. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=krz7T0xU9Z_.

[36] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8: 143–195, 1999. doi: 10.1017/S0962492900002919.

[37] Roei Sarussi, Alon Brutzkus, and Amir Globerson. Towards understanding learning in neural networks with linear teachers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9313–9322. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/sarussi21a.html.

[38] Andrew Saxe, Shagun Sodhani, and Sam Jay Lewallen. The neural race reduction: Dynamics of abstraction in gated networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19287–19309. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/saxe22a.html.

[39] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6120.

[40] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019. doi: 10.1073/pnas.1820226116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1820226116.

[41] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19 (70):1–57, 2018. URL http://jmlr.org/papers/v19/18-188.html.

[42] Nadav Timor, Gal Vardi, and Ohad Shamir. Implicit regularization towards rank minimization in relu networks. In Shipra Agrawal and Francesco Orabona, editors, *Proceedings of The 34th International Conference on Algorithmic Learning Theory*, volume 201 of *Proceedings of Machine Learning Research*, pages 1429–1459. PMLR, 20 Feb–23 Feb 2023. URL https://proceedings.mlr.press/v201/timor23a.html.

[43] Gal Vardi and Ohad Shamir. Implicit regularization in relu networks with the square loss. In Mikhail Belkin and Samory Kpotufe, editors, *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 4224–4258. PMLR, 15–19 Aug 2021. URL https://proceedings.mlr.press/v134/vardi21b.html.

[44] Gal Vardi, Gilad Yehudai, and Ohad Shamir. Gradient methods provably converge to non-robust networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20921–20932. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/83e6913572ba09b0ab53c64c016c7d1a-Paper-Conference.pdf.

[45] Gang Wang, Georgios B. Giannakis, and Jie Chen. Learning relu networks on linearly separable data: Algorithm, optimality, and generalization. *IEEE Transactions on Signal Processing*, 67 (9):2357–2370, 2019. doi: 10.1109/TSP.2019.2904921.

[46] Mingze Wang and Chao Ma. Understanding multi-phase optimization dynamics and rich non-linear behaviors of reLU networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=konBXvt2iS.

[47] Yifei Wang and Mert Pilanci. The convex geometry of backpropagation: Neural network gradient flows converge to extreme points of the dual convex program. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=5QhUE1qiVC6.

[48] Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 3635–3673. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/woodworth20a.html.

[49] Zhiwei Xu, Yutong Wang, Spencer Frei, Gal Vardi, and Wei Hu. Benign overfitting and grokking in relu networks for xor cluster data. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BxHgpC6FNv.

[50] Xiao Zhang, Yaodong Yu, Lingxiao Wang, and Quanquan Gu. Learning one-hidden-layer relu networks via gradient descent. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1524–1534. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/zhang19g.html.

[51] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine learning*, 109:467–492, 2020.

Appendix A. Additional Related Work

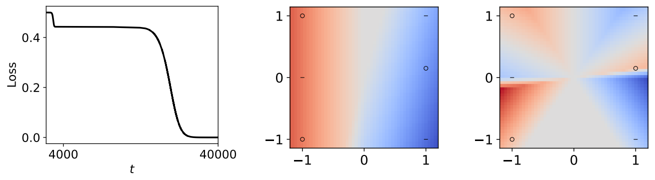
Connections Between ReLU & Linear Networks. Section 1.1 has discussed [8, 29] in detail. In addition, [37] discovered that two-layer bias-free leaky ReLU networks converge to a decision boundary that is very close to linear when the teacher model is linear. Their theoretical results assume that the second layer is fixed while we train all layers of the network. We also consider more general datasets that are not restricted to the linear teacher model. [38] studied gated deep linear networks and found they closely approximate a two-layer bias-free ReLU network trained on an XOR task. But they did not generalize the connection between gated linear networks and ReLU networks beyond the XOR case. [9] gave an example dataset with three scalar input data points, in which two-layer bias-free (leaky) ReLU networks converge to the linear, ordinary least square estimator. [22] studied two-layer leaky ReLU networks with bias and found that they perform linear regression on certain data distributions because the bias fails to move far away from their initialization at zero. However, [9, 22] have only dealt with one-dimensional inputs.

Implicit / Simplicity Bias. Many works have studied the implicit bias or simplicity bias of two-layer bias-free ReLU networks under various assumptions on the dataset. [12, 29, 37, 45, 46] considered linearly separable binary classification tasks. [32, 35, 47] studied orthogonally separable classification (i.e., where for every pair of labeled examples $(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)$ we have $\boldsymbol{x}_i^\top \boldsymbol{x}_j > 0$ if $y_i = y_j$ and $\boldsymbol{x}_i^\top \boldsymbol{x}_j \leq 0$ if otherwise). [10, 17, 18, 25] studied binary classification with exactly or nearly orthogonal input (i.e., where $\boldsymbol{x}_i^\top \boldsymbol{x}_j = 0$ if $i \neq j$). Orthogonal input is a sufficient condition for linear separability for binary classification tasks. [16, 31, 49] studied XOR-like datasets. [19, 44] studied datasets with adversarial noise. We add to this line of research by studying a case with extreme simplicity bias, i.e., behaving like linear networks.

Low-Rank Weights. [30] is the seminal work on the low-rank weights in two-layer ReLU networks trained from small initialization. They described the phenomenon as "quantizing", where the first layer weight vectors align with a small number of directions in the early phase of training. [27] identified when two-layer bias-free ReLU networks form low-rank weights in terms of the initialization and the network width. [42] provided cases where gradient flow on two-layer and deep ReLU networks provably minimize or not minimize the ranks of weight matrices. [18, 25] computed the numerical rank of the converged weights of two-layer bias-free ReLU networks for nearly orthogonal datasets, and found that weights in leaky ReLU networks have rank at most two and weights in ReLU networks have a numerical rank upper bounded by a constant. [13] showed two-layer bias-free ReLU networks are implicitly biased to learn the network of minimal rank under the assumption that training points are correlated with the teacher neuron. [9, 32] studied the early phase learning dynamics to understand how the low-rank weights form. [33, 34] conducted experiments on practical datasets to show that two-layer bias-free ReLU networks learn sparse features, which can be detrimental and lead to overfitting. [26] generalize the low-rank phenomenon in linear and ReLU networks to arbitrary non-homogeneous networks whose last few layers contain linear fully-connected and linear ResNet blocks.

Appendix B. Discussion

Perturbed Symmetric Dataset. We have shown an exact equivalence between two-layer bias-free (leaky) ReLU networks and linear networks in Theorem 5 under symmetry Condition 3 on the dataset. In practice, no datasets satisfy Condition 3 precisely. However, two-layer bias-free ReLU networks may still struggle to fit a dataset that approximately satisfies Condition 3. We present a simple



(*a*) Loss (*b*) $t = 4000$ (*c*) $t = 40000$

Figure 2: Two-layer bias-free ReLU network trained on a dataset that approximately satisfies the symmetry Condition 3. The right middle data point is slight asymmetric. (a) Loss curve. (b,c) The network output is plotted in color during and at the end of training. The circles are data points with $+1$ labels and the lines are data points with -1 labels.

dataset with six data points, as shown in Figure 2. Because this dataset does not satisfy Condition 3 precisely, the two-layer bias-free ReLU can find a nonlinear solution and the loss converges to zero as shown in Figure 2(*a*). However, the loss undergoes a long plateau when the network is stuck in an approximately linear solution as shown in Figure 2(*b*). Moreover, as shown in Figure 2(*c*), the decision boundaries at convergence are close to the data points, and thus possibly non-robust.

Implication of Bias Removal. We have studied the implications of removing bias in ReLU networks in terms of the expressivity and learning dynamics. One common argument in studies of bias-free ReLU networks is that we can stack the input $\boldsymbol{x}$ with an additional one, i.e., $[\boldsymbol{x}, 1]$. Then results derived for bias-free networks could extend to networks with bias and the removal of bias might thus have a minor implication. This argument is valid for some studies [2, 51], but not all. For example, [41] found that two-layer bias-free ReLU networks trained with logistic loss converge to the max-margin classifier on linearly separable datasets. As clarified by [41], this technical result still holds if the inputs are stacked with an additional one. However, the max-margin solution for the dataset with stacked inputs is not the max-margin solution for the original dataset. Thus, the convergence to max-margin solution result does not directly extend to ReLU networks with bias.

Appendix C. Gradient Flow Differential Equations

C.1. Two-Layer Networks

C.1.1. TWO-LAYER (LEAKY) RELU NETWORKS

The gradient flow dynamics of a two-layer (leaky) ReLU network trained with square loss are given by

$$\tau \dot{\boldsymbol{W}}_1 = \left\langle \sigma'(\boldsymbol{W}_1\boldsymbol{x}) \odot \boldsymbol{W}_2^\top \left(y - \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})\right) \boldsymbol{x}^\top \right\rangle, \tag{6a}$$

$$\tau \dot{\boldsymbol{W}}_2 = \left\langle \left(y - \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})\right) \sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle. \tag{6b}$$

where $\odot$ is the Hadamard product, σ' is the derivative of σ, τ is the time constant.

The gradient flow dynamics of a two-layer (leaky) ReLU network trained with logistic loss are

$$\tau \dot{\boldsymbol{W}}_1 = \left\langle \frac{\sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \odot \boldsymbol{W}_2^\top \boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x}) \boldsymbol{x}^\top}{e^{y \boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x})} + 1} \right\rangle, \tag{7a}$$

$$\tau \dot{\boldsymbol{W}}_2 = \boldsymbol{W}_2 \left\langle \frac{\sigma(\boldsymbol{W}_1 \boldsymbol{x}) \sigma(\boldsymbol{W}_1 \boldsymbol{x})^\top}{e^{y \boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x})} + 1} \right\rangle. \tag{7b}$$

C.1.2. TWO-LAYER LINEAR NETWORKS

For linear networks, $\sigma(z) = z$, the gradient flow dynamics with square loss can be written

$$\tau \dot{\boldsymbol{W}}_1^{\text{lin}} = \boldsymbol{W}_2^{\text{lin}\top} \left(\boldsymbol{\beta}^\top - \boldsymbol{W}_2^{\text{lin}} \boldsymbol{W}_1^{\text{lin}} \boldsymbol{\Sigma} \right), \tag{8a}$$

$$\tau \dot{\boldsymbol{W}}_2^{\text{lin}} = \left(\boldsymbol{\beta}^\top - \boldsymbol{W}_2^{\text{lin}} \boldsymbol{W}_1^{\text{lin}} \boldsymbol{\Sigma} \right) \boldsymbol{W}_1^{\text{lin}\top}, \tag{8b}$$

where $\boldsymbol{\Sigma}$ denotes the input data covariance and $\boldsymbol{\beta}$ denotes the input-output correlation

$$\boldsymbol{\Sigma} = \left\langle \boldsymbol{x}\boldsymbol{x}^\top \right\rangle, \quad \boldsymbol{\beta} = \langle y\boldsymbol{x} \rangle, \tag{9}$$

The gradient flow dynamics with logistic loss are

$$\tau \dot{\boldsymbol{W}}_1^{\text{lin}} = \boldsymbol{W}_2^{\text{lin}\top} \boldsymbol{W}_2^{\text{lin}} \boldsymbol{W}_1^{\text{lin}} \left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{y \boldsymbol{W}_2^{\text{lin}} \boldsymbol{W}_1^{\text{lin}} \boldsymbol{x}} + 1} \right\rangle, \tag{10a}$$

$$\tau \dot{\boldsymbol{W}}_2^{\text{lin}} = \boldsymbol{W}_2^{\text{lin}} \boldsymbol{W}_1^{\text{lin}} \left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{y \boldsymbol{W}_2^{\text{lin}} \boldsymbol{W}_1^{\text{lin}} \boldsymbol{x}} + 1} \right\rangle \boldsymbol{W}_1^{\text{lin}\top}. \tag{10b}$$

C.2. Deep Networks

The gradient flow dynamics of a deep neural network with square loss is

$$\tau \dot{\boldsymbol{W}}_l = \left\langle \frac{\partial \boldsymbol{h}_L}{\partial \boldsymbol{h}_l} (y - \boldsymbol{h}_L) \frac{\partial \boldsymbol{h}_l}{\partial \boldsymbol{W}_l} \right\rangle. \tag{11}$$

For deep linear networks, the gradient flow dynamics can be written as

$$\tau \dot{\boldsymbol{W}}_l^{\text{lin}} = \left(\prod_{i=l+1}^{L} \boldsymbol{W}_i^{\text{lin}} \right)^\top \left(\boldsymbol{\beta}^\top - \prod_{i=1}^{L} \boldsymbol{W}_i^{\text{lin}} \boldsymbol{\Sigma} \right) \left(\prod_{i=1}^{l-1} \boldsymbol{W}_i^{\text{lin}} \right)^\top, \tag{12}$$

where $\prod_i \boldsymbol{W}_i$ represents the ordered product of matrices with the largest index on the left and smallest on the right.

Appendix D. Useful Lemmas

Grönwall's Inequality [21] is a common tool to obtain error bounds when considering approximate differential equations.

Lemma 7 (Grönwall's Inequality) *Let I denote an interval of the real line of the form $[a, \infty)$ or $[a, b]$ or $[a, b)$ with $a < b$. Let α, β and u be real-valued functions defined on I. Assume that β and u are continuous and that the negative part of α is integrable on every closed and bounded subinterval of I. If β is non-negative and u satisfies the integral inequality*

$$u(t) \leq \alpha(t) + \int_a^t \beta(s)u(s)ds, \quad \forall t \in I,$$

then

$$u(t) \leq \alpha(t) + \int_a^t \alpha(s)\beta(s)e^{\int_s^t \beta(r)dr}ds, \quad t \in I. \tag{13}$$

Terms in the gradient flow Equation (6) can be bounded by the norm of the weights and the trace of the input covariance matrix.

Lemma 8 $\left\|\left\langle \sigma(\boldsymbol{W}_1\boldsymbol{x})\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle\right\| \leq \|\boldsymbol{W}_1\|^2 \operatorname{Tr} \boldsymbol{\Sigma}$.

Proof

$$\begin{aligned}
\left\|\left\langle \sigma(\boldsymbol{W}_1\boldsymbol{x})\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle\right\| &\leq \left\langle \|\sigma(\boldsymbol{W}_1\boldsymbol{x})\|^2 \right\rangle \\
&\leq \left\langle \|\boldsymbol{W}_1\boldsymbol{x}\|^2 \right\rangle \\
&\leq \left\langle \|\boldsymbol{W}_1\|^2\|\boldsymbol{x}\|^2 \right\rangle \\
&= \|\boldsymbol{W}_1\|^2 \operatorname{Tr} \boldsymbol{\Sigma}
\end{aligned}$$

■

Lemma 9 $\|\langle \sigma(\boldsymbol{W}_1\boldsymbol{x})\sigma'(\boldsymbol{W}_1\boldsymbol{x})\rangle\| \leq \|\boldsymbol{W}_1\| \operatorname{Tr} \boldsymbol{\Sigma}$.

Proof

$$\begin{aligned}
\left\|\left\langle \sigma(\boldsymbol{W}_1\boldsymbol{x})\sigma'(\boldsymbol{W}_1\boldsymbol{x}) \right\rangle\right\| &\leq \left\langle \|\sigma(\boldsymbol{W}_1\boldsymbol{x})\|\|\sigma'(\boldsymbol{W}_1\boldsymbol{x})\| \right\rangle \\
&\leq \left\langle \|\boldsymbol{W}_1\boldsymbol{x}\|\|\boldsymbol{x}\| \right\rangle \\
&\leq \left\langle \|\boldsymbol{W}_1\|\|\boldsymbol{x}\|^2 \right\rangle \\
&= \|\boldsymbol{W}_1\| \operatorname{Tr} \boldsymbol{\Sigma}
\end{aligned}$$

■

The key implication of Condition 3 we exploit is that the input covariance matrix and the input-output correlation averaged over any half space is equal to those averaged over the entire space.

Lemma 10 *Let set $\mathbb{S}^+$ be an arbitrary half space divided by a hyperplane with normal vector $\boldsymbol{r}$, namely $\mathbb{S}^+ = \{\boldsymbol{x} \in \mathbb{R}^D | \boldsymbol{r}^\top \boldsymbol{x} > 0\}$. Under the first condition in Condition 3, we have $\forall \boldsymbol{r}$*

$$\left\langle \boldsymbol{x}\boldsymbol{x}^\top \right\rangle_{\mathbb{S}^+} = \boldsymbol{\Sigma}. \tag{14}$$

Under Condition 3, we have $\forall \boldsymbol{r}$

$$\langle \boldsymbol{x}y(\boldsymbol{x})\rangle_{\mathbb{S}^+} = \boldsymbol{\beta} \tag{15}$$

Recall that $\boldsymbol{\Sigma}$ and $\boldsymbol{\beta}$ are averages over the entire space as defined in Equation (9).

Proof . Define $\mathbb{S}^- = \{\boldsymbol{x} \in \mathbb{R}^D | \boldsymbol{r}^\top \boldsymbol{x} < 0\}$. Because Condition 3 states that $p(\boldsymbol{x})$ is even, we have $\int_{\mathbb{S}^+} p(\boldsymbol{x}) d\boldsymbol{x} = \int_{\mathbb{S}^-} p(\boldsymbol{x}) d\boldsymbol{x} = 1/2$. Because $\boldsymbol{x}\boldsymbol{x}^\top$ is an even function about $\boldsymbol{x}$, its average in $\mathbb{S}^+$ is equal to its average in the other half space $\mathbb{S}^-$,

$$\left\langle \boldsymbol{x}\boldsymbol{x}^\top \right\rangle_{\mathbb{S}^+} \equiv \frac{1}{2} \int_{\mathbb{S}^+} \boldsymbol{x}\boldsymbol{x}^\top p(\boldsymbol{x}) d\boldsymbol{x} = \frac{1}{2} \int_{\mathbb{S}^-} \boldsymbol{x}\boldsymbol{x}^\top p(\boldsymbol{x}) d\boldsymbol{x}.$$

Thus the average in $\mathbb{S}^+$ is equal to the average in the entire space,

$$\begin{aligned}\boldsymbol{\Sigma} &\equiv \int \boldsymbol{x}\boldsymbol{x}^\top p(\boldsymbol{x}) d\boldsymbol{x} \\ &= \frac{1}{2} \int_{\mathbb{S}^+} \boldsymbol{x}\boldsymbol{x}^\top p(\boldsymbol{x}) d\boldsymbol{x} + \frac{1}{2} \int_{\mathbb{S}^-} \boldsymbol{x}\boldsymbol{x}^\top p(\boldsymbol{x}) d\boldsymbol{x} \\ &= \int_{\mathbb{S}^+} \boldsymbol{x}\boldsymbol{x}^\top p(\boldsymbol{x}) d\boldsymbol{x}.\end{aligned}$$

The same holds for $\boldsymbol{x}y(\boldsymbol{x})$. ■

Lemma 11 *Under Condition 3, the first terms in the differential Equation (6) can be simplified to*

$$\left\langle \sigma'(\boldsymbol{W}_1\boldsymbol{x}) \odot \boldsymbol{W}_2^\top y \boldsymbol{x}^\top \right\rangle = \frac{\alpha+1}{2} \boldsymbol{W}_2^\top \boldsymbol{\beta}^\top, \tag{16a}$$

$$\left\langle y \sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle = \frac{\alpha+1}{2} \boldsymbol{\beta}^\top \boldsymbol{W}_1^\top. \tag{16b}$$

Proof . Let us consider the h-th row of the matrix $\langle \sigma'(\boldsymbol{W}_1\boldsymbol{x}) \odot \boldsymbol{W}_2^\top y \boldsymbol{x}^\top \rangle$, which is

$$\left\langle \sigma'(\boldsymbol{w}_{1h}\boldsymbol{x}) w_{2h} y \boldsymbol{x}^\top \right\rangle = \frac{1}{2} \left\langle \alpha w_{2h} y \boldsymbol{x}^\top \right\rangle_{\boldsymbol{w}_{1h}\boldsymbol{x}<0} + \frac{1}{2} \left\langle w_{2h} y \boldsymbol{x}^\top \right\rangle_{\boldsymbol{w}_{1h}\boldsymbol{x}>0} = \frac{\alpha+1}{2} w_{2h} \boldsymbol{\beta}^\top,$$

where the second equality uses Equation (15). Since this holds for all rows, we arrive at Equation (16a).

Let us consider the h-th element of the row vector $\langle y\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \rangle$, which is

$$\langle y\sigma(\boldsymbol{w}_{1h}\boldsymbol{x}) \rangle = \frac{1}{2} \langle \alpha y \boldsymbol{w}_{1h}\boldsymbol{x} \rangle_{\boldsymbol{w}_{1h}\boldsymbol{x}<0} + \frac{1}{2} \langle y \boldsymbol{w}_{1h}\boldsymbol{x} \rangle_{\boldsymbol{w}_{1h}\boldsymbol{x}>0} = \frac{\alpha+1}{2} \boldsymbol{w}_{1h} \boldsymbol{\beta},$$

where the second equality uses Equation (15). Since this holds for all elements, we arrive at Equation (16b). ■

Appendix E. Two-Layer Network Learning Dynamics on Symmetric Datasets

E.1. Early Phase

In the early phase of learning, the network output is small compared with the target output since the initialization is small. We specify how small the norm of the weights is in Lemma 12.

Lemma 12 *Denote the larger L2 norm of the weights in a two-layer network as* $u(t) = \max\{\|\boldsymbol{W}_1(t)\|, \|\boldsymbol{W}_2(t)\|\}$. *For two-layer linear, ReLU, or leaky ReLU networks trained with square loss from small initialization (that is* $u(0) \ll 1$*),* $u(t)$ *is bounded by*

$$u(t) \leq u(0)e^{(s+\operatorname{Tr}\boldsymbol{\Sigma})t/\tau}, \tag{17}$$

for time $t < \frac{\tau}{s+\operatorname{Tr}\boldsymbol{\Sigma}} \ln \frac{1}{u(0)}$.

Proof . For two-layer linear, ReLU, or leaky ReLU networks, the learning dynamics are given in general in Equation (6). Using the inequality in Lemma 8, we can bound the dynamics of u as

$$\begin{aligned}
\tau \frac{d}{dt}u^2 = \tau\frac{d}{dt}\|\boldsymbol{W}_2\|^2 &= (\alpha+1)\boldsymbol{\beta}^\top \boldsymbol{W}_1^\top \boldsymbol{W}_2^\top - 2\boldsymbol{W}_2 \left\langle \sigma(\boldsymbol{W}_1\boldsymbol{x})\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle \boldsymbol{W}_2^\top \\
&\leq \left|(\alpha+1)\boldsymbol{\beta}^\top \boldsymbol{W}_1^\top \boldsymbol{W}_2^\top\right| + \left|2\boldsymbol{W}_2 \left\langle \sigma(\boldsymbol{W}_1\boldsymbol{x})\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle \boldsymbol{W}_2^\top\right| \\
&\leq 2\|\boldsymbol{\beta}\|\|\boldsymbol{W}_1\|\|\boldsymbol{W}_2\| + 2\|\boldsymbol{W}_2\|^2\|\boldsymbol{W}_1\|^2 \operatorname{Tr}\boldsymbol{\Sigma} \\
&\leq 2su^2 + 2u^4 \operatorname{Tr}\boldsymbol{\Sigma}.
\end{aligned}$$

For $u < 1$, we have

$$\tau\frac{d}{dt}u^2 \leq 2su^2 + 2u^4 \operatorname{Tr}\boldsymbol{\Sigma} < 2\left(s + \operatorname{Tr}\boldsymbol{\Sigma}\right)u^2.$$

Via Lemma 7 Grönwall's Inequality, we obtain

$$u^2 \leq u(0)^2 e^{2(s+\operatorname{Tr}\boldsymbol{\Sigma})t/\tau} \quad \Rightarrow \quad u(t) \leq u(0)e^{(s+\operatorname{Tr}\boldsymbol{\Sigma})t/\tau}.$$

This holds for

$$t < \frac{\tau}{s+\operatorname{Tr}\boldsymbol{\Sigma}} \ln\frac{1}{u(0)}.$$

■

Since the weights are small in the early phase, we can approximate the early phase dynamics with only the first terms in Equation (6), that is

$$\tau\dot{\boldsymbol{W}}_1 \approx \left\langle \sigma'(\boldsymbol{W}_1\boldsymbol{x}) \odot \boldsymbol{W}_2^\top y\boldsymbol{x}^\top \right\rangle = \frac{\alpha+1}{2}\boldsymbol{W}_2^\top\boldsymbol{\beta}^\top, \tag{18}$$

$$\tau\dot{\boldsymbol{W}}_2 \approx \left\langle y\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle = \frac{\alpha+1}{2}\boldsymbol{\beta}^\top\boldsymbol{W}_1^\top, \tag{19}$$

where the equalities hold under Condition 3 as proved by Lemma 11. We solve the approximate early phase dynamics and prove that the approximation introduces small errors in Theorem 13.

Theorem 13 *Assume the initial norm of both layers are equally small, that is* $\|\boldsymbol{W}_1(0)\| = \|\boldsymbol{W}_2(0)\| = w_{\text{init}}$ *is small. For time* $t < \frac{\tau}{s+\operatorname{Tr}\boldsymbol{\Sigma}} \ln\frac{1}{w_{\text{init}}}$*, the solution to Equation (6) is exponential growth along one direction with small errors*

$$\boldsymbol{W}_1(t) = e^{\frac{\alpha+1}{2\tau}st}\boldsymbol{r}_1\bar{\boldsymbol{\beta}}^\top + O(w_{\text{init}}), \quad \boldsymbol{W}_2(t) = e^{\frac{\alpha+1}{2\tau}st}\boldsymbol{r}_1^\top + O(w_{\text{init}}). \tag{20}$$

where $s = \|\boldsymbol{\beta}\|, \bar{\boldsymbol{\beta}} = \boldsymbol{\beta}/s$*, and* $\boldsymbol{r}_1$ *is determined by random initialization* $\boldsymbol{r}_1 = \left(\boldsymbol{W}_1(0)\bar{\boldsymbol{\beta}} + {\boldsymbol{W}_2}^\top(0)\right)/2$.

Proof . We first consider the approximate learning dynamics:

$$\tau \dot{\widetilde{\boldsymbol{W}}}_1 = \frac{\alpha+1}{2} \widetilde{\boldsymbol{W}}_2^\top \boldsymbol{\beta}^\top, \quad \tau \dot{\widetilde{\boldsymbol{W}}}_2 = \frac{\alpha+1}{2} \boldsymbol{\beta}^\top \widetilde{\boldsymbol{W}}_1^\top. \tag{21}$$

This is a linear dynamical system with an analytical solution available. We re-write it as:

$$\tau \frac{d}{dt} \widetilde{\boldsymbol{W}} = \frac{\alpha+1}{2} \boldsymbol{M} \widetilde{\boldsymbol{W}}, \quad \text{where } \boldsymbol{M} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{\beta} \\ \boldsymbol{\beta}^\top & 0 \end{bmatrix}, \widetilde{\boldsymbol{W}} = \begin{bmatrix} \widetilde{\boldsymbol{W}}_1^\top \\ \widetilde{\boldsymbol{W}}_2 \end{bmatrix}. \tag{22}$$

Since matrix $\boldsymbol{M}$ only has two nonzero eigenvalues $\pm s$, the solution to Equation (22) is

$$\begin{aligned} \widetilde{\boldsymbol{W}}(t) = & \frac{1}{2} e^{\frac{\alpha+1}{2\tau} st} \begin{bmatrix} \bar{\boldsymbol{\beta}} \\ 1 \end{bmatrix} \left(\bar{\boldsymbol{\beta}}^\top \boldsymbol{W}_1^\top(0) + \boldsymbol{W}_2(0) \right) \\ & + \frac{1}{2} e^{-\frac{\alpha+1}{2\tau} st} \begin{bmatrix} \bar{\boldsymbol{\beta}} \\ -1 \end{bmatrix} \left(\bar{\boldsymbol{\beta}}^\top \boldsymbol{W}_1^\top(0) - \boldsymbol{W}_2(0) \right) + \begin{bmatrix} \left(\boldsymbol{I} - \bar{\boldsymbol{\beta}} \bar{\boldsymbol{\beta}}^\top \right) \boldsymbol{W}_1^\top(0) \\ 0 \end{bmatrix}. \end{aligned} \tag{23}$$

Note that only the first term in Equation (23) is growing.

We then consider the exact learning dynamics given by Equation (6) and prove its solution is close to $\widetilde{\boldsymbol{W}}$. The dynamics of the difference between the exact and approximate dynamics are

$$\tau \frac{d}{dt} \left(\widetilde{\boldsymbol{W}}_1 - \boldsymbol{W}_1 \right) = \frac{\alpha+1}{2} \left(\widetilde{\boldsymbol{W}}_2 - \boldsymbol{W}_2 \right)^\top \boldsymbol{\beta}^\top + \boldsymbol{W}_2{}^\top \boldsymbol{W}_2 \left\langle \sigma(\boldsymbol{W}_1 \boldsymbol{x}) \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \right\rangle \tag{24a}$$

$$\tau \frac{d}{dt} \left(\widetilde{\boldsymbol{W}}_2 - \boldsymbol{W}_2 \right) = \frac{\alpha+1}{2} \boldsymbol{\beta}^\top \left(\widetilde{\boldsymbol{W}}_1 - \boldsymbol{W}_1 \right)^\top + \boldsymbol{W}_2 \left\langle \sigma(\boldsymbol{W}_1 \boldsymbol{x}) \sigma(\boldsymbol{W}_1 \boldsymbol{x})^\top \right\rangle. \tag{24b}$$

We re-write Equation (24) as

$$\tau \frac{d}{dt} \delta \boldsymbol{W} = \frac{\alpha+1}{2} \boldsymbol{M} \delta \boldsymbol{W} + \boldsymbol{\epsilon}, \tag{25}$$

We can bound the norm of the two terms of $\boldsymbol{\epsilon}$ via Lemmas 8 and 9

$$\begin{aligned} \left\| \boldsymbol{W}_2{}^\top \boldsymbol{W}_2 \left\langle \sigma(\boldsymbol{W}_1 \boldsymbol{x}) \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \right\rangle \right\| &\leq u^3 \operatorname{Tr} \boldsymbol{\Sigma}, \\ \| \boldsymbol{W}_2 \left\langle \sigma(\boldsymbol{W}_1 \boldsymbol{x}) \sigma(\boldsymbol{W}_1 \boldsymbol{x})^\top \right\rangle \| &\leq u^3 \operatorname{Tr} \boldsymbol{\Sigma}. \end{aligned}$$

We can then substitute in Equation (17) and obtain

$$\|\boldsymbol{\epsilon}\| \leq \sqrt{2} u^3 \operatorname{Tr} \boldsymbol{\Sigma} < \sqrt{2} u_0^3 e^{3(s + \operatorname{Tr} \boldsymbol{\Sigma}) t/\tau} \operatorname{Tr} \boldsymbol{\Sigma}.$$

We now bound the norm of $\boldsymbol{W} - \widetilde{\boldsymbol{W}}$:

$$\begin{aligned} \left\| \boldsymbol{W} - \widetilde{\boldsymbol{W}} \right\| &= \left\| \int_0^t \frac{\alpha+1}{2} \boldsymbol{M} \left(\boldsymbol{W} - \widetilde{\boldsymbol{W}} \right) + \boldsymbol{\epsilon} dt \right\| \\ &\leq \int_0^t \|\boldsymbol{M}\| \left\| \boldsymbol{W} - \widetilde{\boldsymbol{W}} \right\| + \|\boldsymbol{\epsilon}\| dt \\ &\leq \int_0^t \left(\sqrt{2} s \left\| \boldsymbol{W} - \widetilde{\boldsymbol{W}} \right\| + \sqrt{2} u_0^3 e^{3(s + \operatorname{Tr} \boldsymbol{\Sigma}) t/\tau} \operatorname{Tr} \boldsymbol{\Sigma} \right) dt \\ &\leq \frac{\sqrt{2} u_0^3 \operatorname{Tr} \boldsymbol{\Sigma}}{3(s + \operatorname{Tr} \boldsymbol{\Sigma})} \left(e^{3(s + \operatorname{Tr} \boldsymbol{\Sigma}) t/\tau} - 1 \right) + \sqrt{2} \int_0^t s \left\| \boldsymbol{W} - \widetilde{\boldsymbol{W}} \right\| dt. \end{aligned}$$

Via Lemma 7 Grönwall's Inequality, we obtain

$$\begin{aligned}\left\|\boldsymbol{W}-\widetilde{\boldsymbol{W}}\right\| &\leq \frac{\sqrt{2}\operatorname{Tr}\boldsymbol{\Sigma}u_0^3}{3(s+\operatorname{Tr}\boldsymbol{\Sigma})}\left[e^{3(s+\operatorname{Tr}\boldsymbol{\Sigma})t/\tau}-1+\int_0^t\left(e^{3(s+\operatorname{Tr}\boldsymbol{\Sigma})t'/\tau}-1\right)\sqrt{2}se^{\sqrt{2}st'}dt'\right]\\ &=\frac{\sqrt{2}\operatorname{Tr}\boldsymbol{\Sigma}u_0^3}{3(s+\operatorname{Tr}\boldsymbol{\Sigma})}\left[e^{3(s+\operatorname{Tr}\boldsymbol{\Sigma})t/\tau}+\frac{\sqrt{2}s\left(e^{[3(s+\operatorname{Tr}\boldsymbol{\Sigma})+\sqrt{2}s]t/\tau}-1\right)}{3(s+\operatorname{Tr}\boldsymbol{\Sigma})+\sqrt{2}s}-e^{\sqrt{2}st}\right].\end{aligned}$$

When $t<\frac{\tau}{s+\operatorname{Tr}\boldsymbol{\Sigma}}\ln\frac{1}{u_0}$, we have $\left\|\boldsymbol{W}-\widetilde{\boldsymbol{W}}\right\|<C_1u_0^2$ for some constant C_1.

We are now ready to bound the difference between the exact solution and an exponential function along one direction

$$\begin{aligned}&\boldsymbol{W}-e^{\frac{\alpha+1}{2\tau}st}\begin{bmatrix}\bar{\boldsymbol{\beta}}\\1\end{bmatrix}\boldsymbol{r}_1^\top\\ =&\left(\boldsymbol{W}-\widetilde{\boldsymbol{W}}\right)+\left(\widetilde{\boldsymbol{W}}-e^{-\frac{\alpha+1}{2\tau}st}\begin{bmatrix}\bar{\boldsymbol{\beta}}\\1\end{bmatrix}\boldsymbol{r}_1^\top\right)\\ =&\left(\boldsymbol{W}-\widetilde{\boldsymbol{W}}\right)+\frac{1}{2}e^{-\frac{\alpha+1}{2\tau}st}\begin{bmatrix}\bar{\boldsymbol{\beta}}\\-1\end{bmatrix}\left(\bar{\boldsymbol{\beta}}^\top\boldsymbol{W}_1^\top(0)-\boldsymbol{W}_2(0)\right)+\begin{bmatrix}\left(\boldsymbol{I}-\bar{\boldsymbol{\beta}}\bar{\boldsymbol{\beta}}^\top\right)\boldsymbol{W}_1^\top(0)\\0\end{bmatrix}.\end{aligned}$$

The first term arises from our approximation of dropping the cubic terms in the dynamics. Its norm is bounded by $C_1w_{\text{init}}^2$. The second term arises from initialization, which is $O(w_{\text{init}})$. Via triangle inequality, the norm of the total error is order $O(w_{\text{init}})$.

$$\left\|\boldsymbol{W}-e^{\frac{\alpha+1}{2\tau}st}\begin{bmatrix}\bar{\boldsymbol{\beta}}\\1\end{bmatrix}\boldsymbol{r}_1^\top\right\|<C_1w_{\text{init}}^2+C_2w_{\text{init}}<Cw_{\text{init}}.$$

■

Theorem 13 can imply two messages. Firstly, the (leaky) ReLU network has the same time-course solution as its linear counterpart except a scale factor determined by α, which is consistent with Theorem 5. Secondly, the (leaky) ReLU and linear networks form rank-one weights with small errors in the early phase. We will exploit the rank-one weights to reduce the learning dynamics to Equation (34).

E.2. Late Phase

Proof of Theorem 5 (square loss).

Theorem 5 relies on Condition 3 and Assumption 4 and arrives at three statements: implementing the same function as in Equation (4), having the same weights as in Equation (5), and retaining rank-one weights as Assumption 4 retains valid. We prove them one by one.

Part 1: We first prove that the (leaky) ReLU network and the linear network implements the same linear function except scaling when their weights are rank-one, satisfying Assumption 4. Denote $\boldsymbol{W}_2=[\boldsymbol{W}_2^+,\boldsymbol{W}_2^-]$ where $\boldsymbol{W}_2^+$ are the positive elements in $\boldsymbol{W}_2$ and $\boldsymbol{W}_2^-$ are the negative elements in $\boldsymbol{W}_2$. For a (leaky) ReLU network with rank-one weights as in Assumption 4, we have

$$f(\boldsymbol{x};\boldsymbol{W})=\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})=\boldsymbol{W}_2\sigma\left(\boldsymbol{W}_2^\top\boldsymbol{r}^\top\boldsymbol{x}\right).$$

Notate the positive and negative half-space as

$$\mathbb{S}^+ = \left\{\boldsymbol{x} \in \mathbb{R}^D \middle| \boldsymbol{r}^\top \boldsymbol{x} \geq 0\right\}, \quad \mathbb{S}^- = \left\{\boldsymbol{x} \in \mathbb{R}^D \middle| \boldsymbol{r}^\top \boldsymbol{x} < 0\right\}. \tag{26}$$

For $\boldsymbol{x} \in \mathbb{S}^+$, we have

$$f(\boldsymbol{x}; \boldsymbol{W}) = \boldsymbol{r}^\top \boldsymbol{x} \boldsymbol{W}_2 \sigma(\boldsymbol{W}_2^\top) = \boldsymbol{r}^\top \boldsymbol{x} \begin{bmatrix} \boldsymbol{W}_2^+ & \boldsymbol{W}_2^- \end{bmatrix} \begin{bmatrix} \boldsymbol{W}_2^{+\top} \\ \alpha \boldsymbol{W}_2^{-\top} \end{bmatrix} = \boldsymbol{r}^\top \boldsymbol{x} \left(\|\boldsymbol{W}_2^+\|^2 + \alpha \|\boldsymbol{W}_2^-\|^2\right).$$

For $\boldsymbol{x} \in \mathbb{S}^-$, we have

$$f(\boldsymbol{x}; \boldsymbol{W}) = -\boldsymbol{r}^\top \boldsymbol{x} \boldsymbol{W}_2 \sigma(-\boldsymbol{W}_2^\top) = \boldsymbol{r}^\top \boldsymbol{x} \begin{bmatrix} \boldsymbol{W}_2^+ & \boldsymbol{W}_2^- \end{bmatrix} \begin{bmatrix} \alpha \boldsymbol{W}_2^{+\top} \\ \boldsymbol{W}_2^{-\top} \end{bmatrix} = \boldsymbol{r}^\top \boldsymbol{x} \left(\alpha \|\boldsymbol{W}_2^+\|^2 + \|\boldsymbol{W}_2^-\|^2\right).$$

Under Assumption 4, we have $\|\boldsymbol{W}_2^+\| = \|\boldsymbol{W}_2^-\|$. Hence, the (leaky) ReLU network implements

$$f(\boldsymbol{x}; \boldsymbol{W}) = \boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x}) = \frac{\alpha + 1}{2} \boldsymbol{r}^\top \boldsymbol{x} \|\boldsymbol{W}_2\|^2. \tag{27}$$

Under Assumption 4, the linear network implements

$$f^{\text{lin}}(\boldsymbol{x}; \boldsymbol{W}) = \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x} = \boldsymbol{W}_2 \boldsymbol{W}_2^\top \boldsymbol{r}^\top \boldsymbol{x} = \boldsymbol{r}^\top \boldsymbol{x} \|\boldsymbol{W}_2\|^2. \tag{28}$$

Comparing Equations (27) and (28), we find that when the weights satisfy Assumption 4, the (leaky) ReLU network implements the same function as the linear network except a constant scale

$$\boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x}) = \frac{\alpha + 1}{2} \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x}. \tag{29}$$

Part 2: We then look into the learning dynamics to prove that the weights in the (leaky) ReLU and the linear network are the same except scaling. Substituting Equation (29) into the dynamics, we obtain

$$\begin{aligned} \tau \dot{\boldsymbol{W}}_1 &= \frac{\alpha + 1}{2} \boldsymbol{W}_2^\top \boldsymbol{\beta}^\top - \left\langle \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \odot \boldsymbol{W}_2^\top \boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x}) \boldsymbol{x}^\top \right\rangle \\ &= \frac{\alpha + 1}{2} \boldsymbol{W}_2^\top \boldsymbol{\beta}^\top - \frac{\alpha + 1}{2} \left\langle \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \odot \boldsymbol{W}_2^\top \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x} \boldsymbol{x}^\top \right\rangle, \end{aligned} \tag{30a}$$

$$\begin{aligned} \tau \dot{\boldsymbol{W}}_2 &= \frac{\alpha + 1}{2} \boldsymbol{\beta}^\top \boldsymbol{W}_1^\top - \boldsymbol{W}_2 \left\langle \sigma(\boldsymbol{W}_1 \boldsymbol{x}) \sigma(\boldsymbol{W}_1 \boldsymbol{x})^\top \right\rangle \\ &= \frac{\alpha + 1}{2} \boldsymbol{\beta}^\top \boldsymbol{W}_1^\top - \frac{\alpha + 1}{2} \boldsymbol{W}_2 \boldsymbol{W}_1 \left\langle \boldsymbol{x} \sigma(\boldsymbol{W}_1 \boldsymbol{x})^\top \right\rangle. \end{aligned} \tag{30b}$$

We compute the second terms in the dynamics under Condition 3 and Assumption 4

$$\begin{aligned} &\left\langle \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \odot \boldsymbol{W}_2^\top \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x} \boldsymbol{x}^\top \right\rangle \qquad (31) \\ =& \frac{1}{2} \left\langle \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \odot \boldsymbol{W}_2^\top \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x} \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^+} + \frac{1}{2} \left\langle \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \odot \boldsymbol{W}_2^\top \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x} \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^-} \\ =& \frac{1}{2} \begin{bmatrix} \mathbf{1} \\ \alpha \mathbf{1} \end{bmatrix} \odot \boldsymbol{W}_2^\top \boldsymbol{W}_2 \boldsymbol{W}_1 \left\langle \boldsymbol{x} \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^+} + \frac{1}{2} \begin{bmatrix} \alpha \mathbf{1} \\ \mathbf{1} \end{bmatrix} \odot \boldsymbol{W}_2^\top \boldsymbol{W}_2 \boldsymbol{W}_1 \left\langle \boldsymbol{x} \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^-} \\ =& \frac{1}{2} \begin{bmatrix} \boldsymbol{W}_2^{+\top} \\ \alpha \boldsymbol{W}_2^{-\top} \end{bmatrix} \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{\Sigma} + \frac{1}{2} \begin{bmatrix} \alpha \boldsymbol{W}_2^{+\top} \\ \boldsymbol{W}_2^{-\top} \end{bmatrix} \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{\Sigma} \\ =& \frac{\alpha + 1}{2} \boldsymbol{W}_2^\top \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{\Sigma}, \qquad (32) \end{aligned}$$

and

$$\begin{aligned}
\left\langle \boldsymbol{x}\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle &= \frac{1}{2}\left\langle \boldsymbol{x}\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle_{\mathbb{S}^+} + \frac{1}{2}\left\langle \boldsymbol{x}\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top \right\rangle_{\mathbb{S}^-} \\
&= \frac{1}{2}\left\langle \boldsymbol{x}\boldsymbol{x}^\top \right\rangle_{\mathbb{S}^+} \boldsymbol{r}\left[\boldsymbol{W}_2^+ \quad \alpha\boldsymbol{W}_2^-\right] + \frac{1}{2}\left\langle \boldsymbol{x}\boldsymbol{x}^\top \right\rangle_{\mathbb{S}^-} \boldsymbol{r}\left[\alpha\boldsymbol{W}_2^+ \quad \boldsymbol{W}_2^-\right] \\
&= \frac{\alpha+1}{2}\boldsymbol{\Sigma}\boldsymbol{r}\boldsymbol{W}_2 \\
&= \frac{\alpha+1}{2}\boldsymbol{\Sigma}\boldsymbol{W}_1^\top.
\end{aligned} \tag{33}$$

Substituting Equations (31) and (33) into Equation (30), we obtain the reduced dynamics

$$\tau\dot{\boldsymbol{W}}_1 = \frac{\alpha+1}{2}\boldsymbol{W}_2^\top\boldsymbol{\beta}^\top - \left(\frac{\alpha+1}{2}\right)^2 \boldsymbol{W}_2{}^\top\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{\Sigma}, \tag{34a}$$

$$\tau\dot{\boldsymbol{W}}_2 = \frac{\alpha+1}{2}\boldsymbol{\beta}^\top\boldsymbol{W}_1^\top - \left(\frac{\alpha+1}{2}\right)^2 \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{\Sigma}\boldsymbol{W}_1{}^\top. \tag{34b}$$

If we scale the weights

$$\overline{\boldsymbol{W}}_1 = \sqrt{\frac{\alpha+1}{2}}\boldsymbol{W}_1, \quad \overline{\boldsymbol{W}}_2 = \sqrt{\frac{\alpha+1}{2}}\boldsymbol{W}_2, \tag{35}$$

the (leaky) ReLU network dynamics is the same as that of a linear network given in Equation (8) except the time constant

$$\frac{2\tau}{\alpha+1}\dot{\overline{\boldsymbol{W}}}_1 = \overline{\boldsymbol{W}}_2^\top\left(\boldsymbol{\beta}^\top - \overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\boldsymbol{\Sigma}\right), \quad \frac{2\tau}{\alpha+1}\dot{\overline{\boldsymbol{W}}}_2 = \left(\boldsymbol{\beta}^\top - \overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\boldsymbol{\Sigma}\right)\overline{\boldsymbol{W}}_1^\top.$$

Because Theorem 5 defines the initial condition $\overline{\boldsymbol{W}}(0) = \sqrt{\frac{\alpha+1}{2}}\boldsymbol{W}(0) = \boldsymbol{W}^{\text{lin}}(0)$, the weights in the linear network and the scaled weights in the (leaky) ReLU network start from the same initialization, obey the same dynamics, and consequently stay the same $\forall\, t \geq 0$

$$\begin{aligned}
\overline{\boldsymbol{W}}(t) &= \boldsymbol{W}^{\text{lin}}\left(\frac{\alpha+1}{2}t\right) \\
\Leftrightarrow \boldsymbol{W}(t) &= \sqrt{\frac{2}{\alpha+1}}\boldsymbol{W}^{\text{lin}}\left(\frac{\alpha+1}{2}t\right).
\end{aligned}$$

This proves Equation (5). Substituting Equation (5) into Equation (29) proves Equation (4)

$$\begin{aligned}
f(\boldsymbol{x};\boldsymbol{W}(t)) = \frac{\alpha+1}{2}\boldsymbol{W}(t)\boldsymbol{W}(t)\boldsymbol{x} &= \boldsymbol{W}_2^{\text{lin}}\left(\frac{\alpha+1}{2}t\right)\boldsymbol{W}_1^{\text{lin}}\left(\frac{\alpha+1}{2}t\right)\boldsymbol{x} \\
&\equiv f^{\text{lin}}\left(\boldsymbol{x};\boldsymbol{W}^{\text{lin}}\left(\frac{\alpha+1}{2}t\right)\right).
\end{aligned}$$

Part 3: We now show that Assumption 4 made at time $t = 0$ remains valid for $t > 0$, meaning that weights which start with rank-one structure remain rank-one. With Assumption 4 at time $t = 0$, the dynamics of the (leaky) ReLU network is described by Equation (34). This dynamics is the same as scaled dynamics in a linear network and thus satisfy the balancing property in linear networks [14, 24]

$$\frac{d}{dt}\left(\boldsymbol{W}_1\boldsymbol{W}_1^\top - \boldsymbol{W}_2^\top\boldsymbol{W}_2\right) = 0. \tag{36}$$

Under Assumption 4 at time $t = 0$, this quantity is zero at time $t = 0$ and will stay zero because of the balancing property

$$\forall\, t \geq 0: \quad \boldsymbol{W}_1(t)\boldsymbol{W}_1(t)^\top - \boldsymbol{W}_2(t)^\top\boldsymbol{W}_2(t) = \boldsymbol{W}_1(0)\boldsymbol{W}_1(0)^\top - \boldsymbol{W}_2(0)^\top\boldsymbol{W}_2(0) = \boldsymbol{0}.$$

Because rank $\left(\boldsymbol{W}_1\boldsymbol{W}_1^\top\right) = \text{rank}(\boldsymbol{W}_1)$, the balancing property enforces that $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ have equal rank. Since $\boldsymbol{W}_2$ is a vector, $\boldsymbol{W}_1$ has rank one. We can write a rank-one matrix as the outer-product between two vectors $\boldsymbol{W}_1 = \boldsymbol{v}\boldsymbol{r}^\top$. We can assume $\|\boldsymbol{r}\| = 1$ for convenience and get

$$\boldsymbol{W}_1\boldsymbol{W}_1^\top = \boldsymbol{v}\boldsymbol{r}^\top\boldsymbol{r}\boldsymbol{v}^\top = \boldsymbol{v}\boldsymbol{v}^\top = \boldsymbol{W}_2^\top\boldsymbol{W}_2 \quad \Rightarrow \quad \boldsymbol{v} = \pm\boldsymbol{W}_2^\top.$$

Because Assumption 4 specifies $\boldsymbol{W}_1 = \boldsymbol{W}_2^\top\boldsymbol{r}^\top$, then $\boldsymbol{v} = \boldsymbol{W}_2^\top$. To summarize, Assumption 4 at time $t = 0$ reduces the learning dynamics of the ReLU network to be similar to that of a linear network. The reduced dynamics satisfies the balancing property which enforces that the weights retain rank-one, satisfying Assumption 4 for all $t \geq 0$. ■

Proof of Theorem 5 (logistic loss).

Part 1: Same as the square loss case because Equation (29) holds regardless of the loss function.

Part 2: Substituting Equation (29) into Equation (7), we get

$$\tau\dot{\boldsymbol{W}}_1 = \frac{\alpha+1}{2}\left\langle \frac{\sigma'(\boldsymbol{W}_1\boldsymbol{x}) \odot \boldsymbol{W}_2^\top\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1}\right\rangle, \tag{37a}$$

$$\tau\dot{\boldsymbol{W}}_2 = \frac{\alpha+1}{2}\boldsymbol{W}_2\boldsymbol{W}_1\left\langle \frac{\boldsymbol{x}\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1}\right\rangle. \tag{37b}$$

Under Condition 3 and Assumption 4 Equation (37) can be simplified

$$\begin{aligned}
&\left\langle \frac{\sigma'(\boldsymbol{W}_1\boldsymbol{x}) \odot \boldsymbol{W}_2^\top\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1}\right\rangle \\
=&\frac{1}{2}\left\langle \frac{\sigma'(\boldsymbol{W}_1\boldsymbol{x}) \odot \boldsymbol{W}_2^\top\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1}\right\rangle_{\mathbb{S}^+} + \frac{1}{2}\left\langle \frac{\sigma'(\boldsymbol{W}_1\boldsymbol{x}) \odot \boldsymbol{W}_2^\top\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1}\right\rangle_{\mathbb{S}^-} \\
=&\frac{1}{2}\begin{bmatrix}\alpha\boldsymbol{W}_2^{+\top} \\ \boldsymbol{W}_2^{-\top}\end{bmatrix}\boldsymbol{W}_2\boldsymbol{W}_1\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1}\right\rangle_{\mathbb{S}^+} + \frac{1}{2}\begin{bmatrix}\boldsymbol{W}_2^{+\top} \\ \alpha\boldsymbol{W}_2^{-\top}\end{bmatrix}\boldsymbol{W}_2\boldsymbol{W}_1\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1}\right\rangle_{\mathbb{S}^-} \\
=&\frac{\alpha+1}{2}\boldsymbol{W}_2^\top\boldsymbol{W}_2\boldsymbol{W}_1\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1}\right\rangle,
\end{aligned}$$

and

$$
\begin{aligned}
&\left\langle \frac{\boldsymbol{x}\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1} \right\rangle \\
=&\frac{1}{2}\left\langle \frac{\boldsymbol{x}\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1} \right\rangle_{\mathbb{S}^+} + \frac{1}{2}\left\langle \frac{\boldsymbol{x}\sigma(\boldsymbol{W}_1\boldsymbol{x})^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1} \right\rangle_{\mathbb{S}^-} \\
=&\frac{1}{2}\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1} \right\rangle_{\mathbb{S}^+} \boldsymbol{r}\begin{bmatrix}\alpha\boldsymbol{W}_2^+ & \boldsymbol{W}_2^-\end{bmatrix} + \frac{1}{2}\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1} \right\rangle_{\mathbb{S}^-} \boldsymbol{r}\begin{bmatrix}\boldsymbol{W}_2^+ & \alpha\boldsymbol{W}_2^-\end{bmatrix} \\
=&\frac{\alpha+1}{2}\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1} \right\rangle \boldsymbol{W}_1{}^\top.
\end{aligned}
$$

Thus, the reduced dynamics of the two-layer (leaky) ReLU network are

$$
\begin{aligned}
\tau\dot{\boldsymbol{W}}_1 &= \left(\frac{\alpha+1}{2}\right)^2 \boldsymbol{W}_2{}^\top\boldsymbol{W}_2\boldsymbol{W}_1\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1} \right\rangle, \\
\tau\dot{\boldsymbol{W}}_2 &= \left(\frac{\alpha+1}{2}\right)^2 \boldsymbol{W}_2\boldsymbol{W}_1\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{\frac{\alpha+1}{2}y\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}}+1} \right\rangle \boldsymbol{W}_1{}^\top.
\end{aligned}
$$

If we scale the weights as Equation (35), the (leaky) ReLU network dynamics is the same as that of a linear network given in Equation (10) except the time constant

$$
\begin{aligned}
\frac{2\tau}{\alpha+1}\dot{\overline{\boldsymbol{W}}}_1 &= \overline{\boldsymbol{W}}_2{}^\top\overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{y\overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\boldsymbol{x}}+1} \right\rangle, \\
\frac{2\tau}{\alpha+1}\dot{\overline{\boldsymbol{W}}}_2 &= \overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\left\langle \frac{\boldsymbol{x}\boldsymbol{x}^\top}{e^{y\overline{\boldsymbol{W}}_2\overline{\boldsymbol{W}}_1\boldsymbol{x}}+1} \right\rangle \overline{\boldsymbol{W}}_1{}^\top.
\end{aligned}
$$

Through the same reasoning as the square loss case, Equations (4) and (5) are proved.

Part 3: Same as the square loss case. ■

E.3. Time-Course Solution

Corollary 14 *For learning with square loss, if the input covariance is white, $\boldsymbol{\Sigma} = \boldsymbol{I}$, the solution to Equation (4) is $f(\boldsymbol{x};\boldsymbol{W}) = \boldsymbol{w}(t)^\top\boldsymbol{x}$ with*

$$
\begin{aligned}
\boldsymbol{w}(t) = &\left(1+\frac{q_1}{q_2}e^{-2s\tilde{t}}\right)\left[\bar{\boldsymbol{\beta}}\left(1-\frac{q_1}{q_2}e^{-2s\tilde{t}}\right)+\frac{2}{q_2}\left(\boldsymbol{I}-\bar{\boldsymbol{\beta}}\bar{\boldsymbol{\beta}}^\top\right)\boldsymbol{r}e^{-s\tilde{t}}\right] \\
&\left[\frac{4}{q_2^2}\left(u_0^{-2}+\left(1-\left(\boldsymbol{r}^\top\bar{\boldsymbol{\beta}}\right)^2\right)\tilde{t}\right)e^{-2s\tilde{t}}+\frac{1}{s}\left(1+\frac{q_1^2}{q_2^2}e^{-2s\tilde{t}}\right)\left(1-e^{-2s\tilde{t}}\right)\right]^{-1}, \quad (38)
\end{aligned}
$$

where $\tilde{t}$ is a shorthand for rescaled time $\tilde{t} = \frac{\alpha+1}{2\tau}t$ and the constant quantities are $s = \|\boldsymbol{\beta}\|, \bar{\boldsymbol{\beta}} = \boldsymbol{\beta}/s, q_1 = 1-\boldsymbol{r}^\top\bar{\boldsymbol{\beta}}, q_2 = 1+\boldsymbol{r}^\top\bar{\boldsymbol{\beta}}, u_0 = \|\boldsymbol{W}_1(0)\|$.

Proof . Based on the equivalence stated in Theorem 5, we directly adopt the solution of linear networks [11, Theorem 3.1] to two-layer bias-free (leaky) ReLU networks. ■

The solution given in Equation (38) matches the simulations in Figure 1(*a*).

E.4. Global Minimum

Proof of Corollary 6.

The converged solution $\boldsymbol{w}^*$ is a direct consequence of the equivalence we showed in Theorem 5 and prior results of linear networks [39, 41].

We now show that for symmetric datasets satisfying Condition 3, the global minimum of a two-layer bias-free (leaky) ReLU network trained with square loss is a linear map. Based on Theorem 1, we can write a two-layer bias-free (leaky) ReLU network as a linear function plus an even function $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{w}^* + f_e(\boldsymbol{x})$ where $f_e(\cdot)$ denotes an even function. For datasets satisfying Condition 3, the square loss is

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\left\langle \left(y - \boldsymbol{x}^\top \boldsymbol{w}^* - f_e(\boldsymbol{x})\right)^2 \right\rangle_{p(\boldsymbol{x})} \\ &= \frac{1}{2}\left\langle \left(y - \boldsymbol{x}^\top \boldsymbol{w}^*\right)^2 - 2(y - \boldsymbol{A}\boldsymbol{x}) f_e(\boldsymbol{x}) + f_e(\boldsymbol{x})^2 \right\rangle_{p(\boldsymbol{x})} \\ &= \frac{1}{2}\left\langle \left(y - \boldsymbol{x}^\top \boldsymbol{w}^*\right)^2 \right\rangle_{p(\boldsymbol{x})} + \frac{1}{2}\left\langle f_e(\boldsymbol{x})^2 \right\rangle_{p(\boldsymbol{x})}.\end{aligned}$$

The square loss attains its minimum when both $\left\langle \left(y - \boldsymbol{x}^\top \boldsymbol{w}^*\right)^2 \right\rangle_{p(\boldsymbol{x})}$ and $\left\langle f_e(\boldsymbol{x})^2 \right\rangle_{p(\boldsymbol{x})}$ are minimized. The former is minimized when $\boldsymbol{w}^* = \boldsymbol{\Sigma}^{-1}\boldsymbol{\beta}$. The latter is minimized when $f_e(\boldsymbol{x}) = 0$. Hence, for symmetric datasets satisfying Condition 3, the two-layer bias-free (leaky) ReLU network achieves globally minimum square loss with the linear, ordinary least squares solution $f(\boldsymbol{x}) = \boldsymbol{w}^{*\top}\boldsymbol{x}$. ■

Appendix F. Deep ReLU Network

F.1. Low-Rank Weights

We empirically find that deep bias-free ReLU networks can form low-rank weights that are similar to those in deep linear networks when the empirical input distribution is even and the target function is linear.

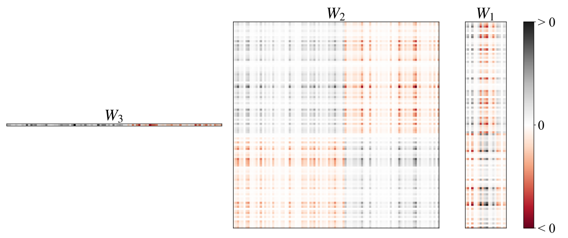
In a deep linear network, weights form a rank-one structure and adjacent layers are aligned when trained from small initialization [1, 5, 24]. The rank-one weight matrices can be written as outer-products of two vectors

$$\boldsymbol{W}_1^{\text{lin}} = u\boldsymbol{r}_1\boldsymbol{r}^\top = u\begin{bmatrix}\boldsymbol{r}_1^+ \\ \boldsymbol{r}_1^-\end{bmatrix}\boldsymbol{r}^\top, \tag{39a}$$

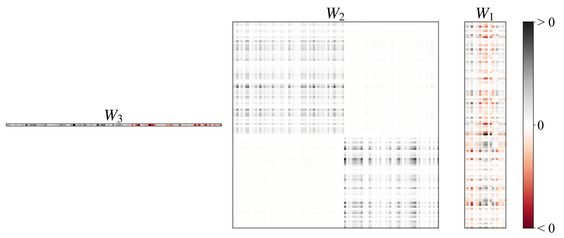
$$\boldsymbol{W}_l^{\text{lin}} = u\boldsymbol{r}_l\boldsymbol{r}_{l-1}^\top = u\begin{bmatrix}\boldsymbol{r}_l^+\boldsymbol{r}_{l-1}^{+\,\top} & \boldsymbol{r}_l^+\boldsymbol{r}_{l-1}^{-\,\top} \\ \boldsymbol{r}_l^-\boldsymbol{r}_{l-1}^{+\,\top} & \boldsymbol{r}_l^-\boldsymbol{r}_{l-1}^{-\,\top}\end{bmatrix}, \quad l = 2, \cdots, L-1, \tag{39b}$$

$$\boldsymbol{W}_L^{\text{lin}} = u\boldsymbol{r}_{L-1}^\top = u\begin{bmatrix}\boldsymbol{r}_{L-1}^{+\,\top} & \boldsymbol{r}_{L-1}^{-\,\top}\end{bmatrix}, \tag{39c}$$

where u represents the norm of each layer, and $\boldsymbol{r}, \boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_L$ are unit norm column vectors. The vectors $\boldsymbol{r}_l^+, \boldsymbol{r}_l^-$ denote the positive and negative elements in $\boldsymbol{r}_l$. The equal norm u of all layers is a consequence of small initialization [14]. Note that the weights can be written in blocks, as



(a) Weights in a 3-layer linear net as Eq. (39).



(b) Weights in a 3-layer ReLU net as Eq. (40).

Figure 3: Low-rank weights in deep linear and ReLU bias-free networks. A three-layer linear network and a three-layer ReLU network are trained on the same dataset starting from the same small random weights. The dataset has a linear target function and an even empirical input data distribution. We plot the weights when the loss has approached zero. $\boldsymbol{W}_1$, $\boldsymbol{W}_3$, and positive elements in $\boldsymbol{W}_2$ have approximately the same structure in the linear and ReLU networks. Elements of $\boldsymbol{W}_2$ that are negative in the linear network are approximately zero in the ReLU network. The neurons are permuted for better visualization.

Equation (39), only after permuting the positive and negative elements. We use this permuted notation for convenience only; no additional assumptions are required.

In a deep ReLU network, we find empirically that when the weights are trained from small initialization and the target function is linear, the weights form a rank-one and rank-two structure. The deep bias-free network's weights can be written approximately as

$$\boldsymbol{W}_1 = u\boldsymbol{r}_1\boldsymbol{r}^\top = u\begin{bmatrix}\boldsymbol{r}_1^+\\ \boldsymbol{r}_1^-\end{bmatrix}\boldsymbol{r}^\top, \tag{40a}$$

$$\boldsymbol{W}_l = u\begin{bmatrix}\sqrt{2}\boldsymbol{r}_l^+\boldsymbol{r}_{l-1}^{+\top} & \boldsymbol{0}\\ \boldsymbol{0} & \sqrt{2}\boldsymbol{r}_l^-\boldsymbol{r}_{l-1}^{-\top}\end{bmatrix}, \quad l = 2,\cdots,L-1, \tag{40b}$$

$$\boldsymbol{W}_L = u\boldsymbol{r}_{L-1}^\top = u\begin{bmatrix}\boldsymbol{r}_{L-1}^{+\top} & \boldsymbol{r}_{L-1}^{-\top}\end{bmatrix}. \tag{40c}$$

For the first and last layers, the weights in the deep ReLU network have the same rank-one structure as their linear counterpart. For the intermediate layers, weights in the deep ReLU network are rank-two. Moreover, positive weights in the ReLU network correspond to positive weights in the linear network and zero weights in the ReLU network correspond to negative weights in the linear network. We present an example of a three-layer linear and a bias-free ReLU network in Figure 3.

Assuming $\|\boldsymbol{r}_l^+\| = \|\boldsymbol{r}_l^-\|, (l = 1,2,\cdots,L-1)$, which is justified as the network width approaches infinity, the deep bias-free ReLU network with weights defined in Equation (40) implements a linear function

$$f(\boldsymbol{x};\boldsymbol{W}) = \boldsymbol{W}_L\boldsymbol{W}_{L-1}\cdots\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x}) = \frac{1}{2}\boldsymbol{W}_L\cdots\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}. \tag{41}$$

In the first equality, we drop the activation functions except the one between the first and second layers. This is because the output of a ReLU activation function, $\sigma(\boldsymbol{W}_1\boldsymbol{x})$, is non-negative, and so are the second layer weights, $\boldsymbol{W}_2$. Therefore, their product, $\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})$, is also non-negative. We thus have $\sigma(\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})) = \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})$. The same applies to all subsequent layers.

When the empirical input distribution is even and the target function is linear, the learning dynamics of the deep bias-free ReLU network with weights as Equation (40) reduces to (see Appendix F.2)

$$\tau \dot{\boldsymbol{W}}_l = \left(\prod_{l'=l+1}^{L} \boldsymbol{W}_{l'} \right)^\top \left(\frac{1}{2}\boldsymbol{\beta}^\top - \frac{1}{4} \prod_{l'=1}^{L} \boldsymbol{W}_{l'} \boldsymbol{\Sigma} \right) \left(\prod_{l'=l-1}^{L} \boldsymbol{W}_{l'} \right)^\top . \tag{42}$$

Except for constant coefficients, these equations are the same as that of the deep linear network given in Equation (12). We show, in Appendix F.2, that weights which have formed a low-rank structure as defined in Equation (40) maintain the structure over training.

We conjecture that the rank-one and rank-two weights in the deep ReLU network relate to the property of aligning between layers discovered in deep linear networks [24]. To take the second layer as an example, the linear network has rank-one weights, $\boldsymbol{W}_2 = u\boldsymbol{r}_2\boldsymbol{r}_1^\top$, so every row of $\boldsymbol{W}_2$ aligns with $\boldsymbol{r}_1$ and thus aligns with the input to the second layer, $\boldsymbol{W}_1\boldsymbol{x} = u\boldsymbol{r}_1\boldsymbol{r}^\top\boldsymbol{x}$. The ReLU network has rank-two weights, and the rows of $\boldsymbol{W}_2$ either align with $\begin{bmatrix}\boldsymbol{r}_1^{+\top} & \boldsymbol{0}\end{bmatrix}$ or $\begin{bmatrix}\boldsymbol{0} & \boldsymbol{r}_1^{-\top}\end{bmatrix}$. The former aligns with some inputs to the second layer $\sigma(\boldsymbol{W}_1\boldsymbol{x}) = u\begin{bmatrix}\boldsymbol{r}_1^+ \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{r}^\top\boldsymbol{x}$ (where $\boldsymbol{r}^\top\boldsymbol{x} > 0$). The latter aligns with $\sigma(\boldsymbol{W}_1\boldsymbol{x}) = u\begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{r}_1^-\end{bmatrix}\boldsymbol{r}^\top\boldsymbol{x}$ (where $\boldsymbol{r}^\top\boldsymbol{x} < 0$).

F.2. Learning Dynamics

According to Equation (40), all weight elements in the intermediate layers ($\boldsymbol{W}_{L-1}, \cdots, \boldsymbol{W}_3, \boldsymbol{W}_2$) are non-negative numbers. According to the definition of the ReLU activation function, $\sigma(\boldsymbol{W}_1\boldsymbol{x})$ yields a vector with non-negative numbers. Thus, $\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})$ yields a vector with non-negative numbers and we have $\sigma(\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})) = \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})$. Similarly, all subsequent ReLU activation functions can be ignored[1]. With weights satisfying Equation (40), a deep bias-free ReLU network reduces to

$$f(\boldsymbol{x};\boldsymbol{W}) \equiv \boldsymbol{W}_L\sigma(\cdots\sigma(\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x}))) = \boldsymbol{W}_L\cdots\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x}).$$

We stick to the notation for the positive and negative half-space defined in Equation (26). For $\boldsymbol{x} \in \mathbb{S}^+$, we have

$$f(\boldsymbol{x};\boldsymbol{W}) = u\boldsymbol{W}_L\cdots\boldsymbol{W}_2\begin{bmatrix}\boldsymbol{r}_1^+ \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{r}^\top\boldsymbol{x} = \cdots = \frac{u^{L-1}}{(\sqrt{2})^{L-2}}\boldsymbol{W}_L\begin{bmatrix}\boldsymbol{r}_{L-1}^+ \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{r}^\top\boldsymbol{x} = \left(\frac{u}{\sqrt{2}}\right)^L\boldsymbol{r}^\top\boldsymbol{x}.$$

For $\boldsymbol{x} \in \mathbb{S}^-$, we have

$$f(\boldsymbol{x};\boldsymbol{W}) = u\boldsymbol{W}_L\cdots\boldsymbol{W}_2\begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{r}_1^-\end{bmatrix}\boldsymbol{r}^\top\boldsymbol{x} = \cdots = \frac{u^{L-1}}{(\sqrt{2})^{L-2}}\boldsymbol{W}_L\begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{r}_{L-1}^-\end{bmatrix}\boldsymbol{r}^\top\boldsymbol{x} = \left(\frac{u}{\sqrt{2}}\right)^L\boldsymbol{r}^\top\boldsymbol{x}.$$

1. The activation functions can be ignored when calculating the network output but cannot be ignored when calculating the gradients. This is because for a ReLU function $\sigma(z) = \max(z,0)$ and a linear function $\phi(z) = z$, the function values are equal at zero $\sigma(0) = \phi(0)$ but the derivatives are not equal at zero $\sigma'(0) \neq \phi'(0)$.

Hence, the deep bias-free ReLU network implements a linear function $f(\boldsymbol{x};\boldsymbol{W}) = \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{x}$. Notice that a deep linear network with such weights implement

$$\boldsymbol{W}_L \cdots \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x} = u \boldsymbol{W}_L \cdots \boldsymbol{W}_2 \begin{bmatrix} \boldsymbol{r}_1^+ \\ \boldsymbol{r}_1^- \end{bmatrix} \boldsymbol{r}^\top \boldsymbol{x} = \cdots = \frac{u^{L-1}}{(\sqrt{2})^{L-2}} \boldsymbol{W}_L \boldsymbol{r}_{L-1} \boldsymbol{r}^\top \boldsymbol{x} = \frac{u^L}{(\sqrt{2})^{L-2}} \boldsymbol{r}^\top \boldsymbol{x}.$$

Equation (41) is thus proved.

We now prove that under Equation (40) on the weights at time $t=0$, we have that $\forall\, t \geq 0$, Equation (40) retains valid. We assume that $\sigma'(0)=0$. We substitute the low-rank weights defined in Equation (40) into the learning dynamics of deep bias-free ReLU networks and make simplifications. For the first layer,

$$\begin{aligned}
\tau \dot{\boldsymbol{W}}_1 &= \left\langle \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \odot \boldsymbol{W}_2^\top \cdots \boldsymbol{W}_L^\top (y - f(\boldsymbol{x})) \boldsymbol{x}^\top \right\rangle \\
&= \frac{u^{L-1}}{(\sqrt{2})^{L-2}} \left\langle \sigma'(\boldsymbol{W}_1 \boldsymbol{x}) \odot \boldsymbol{r}_1 \left(y - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{x} \right) \boldsymbol{x}^\top \right\rangle \\
&= \frac{u^{L-1}}{(\sqrt{2})^L} \begin{bmatrix} \boldsymbol{r}_1^+ \\ \boldsymbol{0} \end{bmatrix} \left\langle \left(y - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{x} \right) \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^+} + \frac{u^{L-1}}{(\sqrt{2})^L} \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{r}_1^- \end{bmatrix} \left\langle \left(y - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{x} \right) \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^-} \\
&= \frac{u^{L-1}}{(\sqrt{2})^L} \boldsymbol{r}_1 \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right). \qquad (43)
\end{aligned}$$

For intermediate layers $1 < l < L$,

$$\begin{aligned}
\tau \dot{\boldsymbol{W}}_l &= \left\langle \sigma'(\boldsymbol{h}_l) \odot \boldsymbol{W}_{l+1}^\top \cdots \boldsymbol{W}_L^\top (y - f(\boldsymbol{x})) \sigma(\boldsymbol{h}_{l-1})^\top \right\rangle \\
&= \left(\frac{u}{\sqrt{2}}\right)^{L-1} \begin{bmatrix} \boldsymbol{1} \\ \boldsymbol{0} \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{r}_l^+ \\ \boldsymbol{r}_l^- \end{bmatrix} \left\langle \left(y - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{x} \right) \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^+} \boldsymbol{r} \begin{bmatrix} {\boldsymbol{r}_{l-1}^+}^\top & \boldsymbol{0} \end{bmatrix} \\
&+ \left(\frac{u}{\sqrt{2}}\right)^{L-1} \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{1} \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{r}_l^+ \\ \boldsymbol{r}_l^- \end{bmatrix} \left\langle \left(y - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{x} \right) \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^-} \boldsymbol{r} \begin{bmatrix} \boldsymbol{0} & {\boldsymbol{r}_{l-1}^-}^\top \end{bmatrix} \\
&= \left(\frac{u}{\sqrt{2}}\right)^{L-1} \begin{bmatrix} \boldsymbol{r}_l^+ {\boldsymbol{r}_{l-1}^+}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right) \boldsymbol{r} \\
&+ \left(\frac{u}{\sqrt{2}}\right)^{L-1} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{r}_l^- {\boldsymbol{r}_{l-1}^-}^\top \end{bmatrix} \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right) \boldsymbol{r} \\
&= \frac{u^{L-1}}{(\sqrt{2})^L} \begin{bmatrix} \sqrt{2}\boldsymbol{r}_l^+ {\boldsymbol{r}_{l-1}^+}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \sqrt{2}\boldsymbol{r}_l^- {\boldsymbol{r}_{l-1}^-}^\top \end{bmatrix} \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right) \boldsymbol{r}. \qquad (44)
\end{aligned}$$

For the last layer,

$$
\begin{aligned}
\tau \dot{\boldsymbol{W}}_L &= \left\langle (y - f(\boldsymbol{x}))\sigma(\boldsymbol{h}_{L-1})^\top \right\rangle \\
&= \frac{u^{L-1}}{(\sqrt{2})^L} \left\langle \left(y - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{x} \right) \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^+} \boldsymbol{r} \begin{bmatrix} {\boldsymbol{r}_{L-1}^+}^\top & \boldsymbol{0} \end{bmatrix} \\
&+ \frac{u^{L-1}}{(\sqrt{2})^L} \left\langle \left(y - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{x} \right) \boldsymbol{x}^\top \right\rangle_{\mathbb{S}^-} \boldsymbol{r} \begin{bmatrix} \boldsymbol{0} & {\boldsymbol{r}_{L-1}^-}^\top \end{bmatrix} \\
&= \frac{u^{L-1}}{(\sqrt{2})^L} \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right) \boldsymbol{r} \boldsymbol{r}_{L-1}^\top.
\end{aligned}
\tag{45}
$$

Equations (43) to (45) can be re-written as Equation (42), which is easy to verify by substituting in the low-rank weights defined in Equation (40). The dynamics of a deep ReLU network given in Equation (42) and the dynamics of a deep linear network given in Equation (12) are the same except for constant coefficients.

We now prove that the low-rank weights retain low-rank once formed. We substitute the low-rank weights defined in Equation (40) into the left-hand side of Equations (43) to (45) and get

$$
\begin{aligned}
\tau \frac{d}{dt} u \boldsymbol{r}_1 \boldsymbol{r}^\top &= \frac{u^{L-1}}{(\sqrt{2})^L} \boldsymbol{r}_1 \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right), \\
\tau \frac{d}{dt} u \begin{bmatrix} \sqrt{2}\boldsymbol{r}_l^+ {\boldsymbol{r}_{l-1}^+}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \sqrt{2}\boldsymbol{r}_l^- {\boldsymbol{r}_{l-1}^-}^\top \end{bmatrix} &= \frac{u^{L-1}}{(\sqrt{2})^L} \begin{bmatrix} \sqrt{2}\boldsymbol{r}_l^+ {\boldsymbol{r}_{l-1}^+}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \sqrt{2}\boldsymbol{r}_l^- {\boldsymbol{r}_{l-1}^-}^\top \end{bmatrix} \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right) \boldsymbol{r}, \\
\tau \frac{d}{dt} u \boldsymbol{r}_{L-1}^\top &= \frac{u^{L-1}}{(\sqrt{2})^L} \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right) \boldsymbol{r} \boldsymbol{r}_{L-1}^\top.
\end{aligned}
$$

By cancelling out the nonzero common terms on both sides, we reduce the dynamics to two differential equations. The first one is about the norm of a layer u. The second one is about the rank-one direction in the first layer $u\boldsymbol{r}$.

$$
\begin{aligned}
\tau \frac{d}{dt} u &= \frac{u^{L-1}}{(\sqrt{2})^L} \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right) \boldsymbol{r}, \\
\tau \frac{d}{dt} u \boldsymbol{r}^\top &= \frac{u^{L-1}}{(\sqrt{2})^L} \left(\boldsymbol{\beta}^\top - \left(\frac{u}{\sqrt{2}}\right)^L \boldsymbol{r}^\top \boldsymbol{\Sigma} \right).
\end{aligned}
$$

After the weights have formed the low-rank structure specified in Equation (40), the norm of each layer u and the rank-one direction of the first layer $\boldsymbol{r}$ evolve while $\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_{L-1}$ stay fixed. Hence, under Equation (40) on the weights at time $t = 0$, we have that $\forall\, t \geq 0$, Equation (40) retains valid.

Appendix G. Implementation Details

All networks are initialized with small random weights. Specifically, the initial weights in the l-th layer are sampled i.i.d. from a normal distribution $\mathcal{N}(0, \gamma^2/N_l)$ where N_l is the number of weights in the l-th layer. The initialization scale γ is specified below.

Figure 1. The networks have width 500. The initialization scale is $\gamma = 10^{-8}$. The learning rate is 0.004. The input is 20-dimensional, $\boldsymbol{x} \in \mathbb{R}^{20}$. We sample 1000 i.i.d. vectors $\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and include both $\boldsymbol{x}_n$ and $-\boldsymbol{x}_n$ in the dataset, resulting in 2000 data points. The output is generated as $y = \boldsymbol{w}^\top \boldsymbol{x} + \sin\left(4\boldsymbol{w}^\top \boldsymbol{x}\right)$ where elements of $\boldsymbol{w}$ are randomly sampled from a uniform distribution $\mathcal{U}[-0.5, 0.5]$. This dataset satisfy Condition 3 since the empirical input distribution is even and the output is generated by an odd function.

Figure 2. The dataset contains six data points $[1,1], [-1,-1], [1,-1], [-1,1], [-1,0], [1,0.15]$. The network width is 100. The initialization scale is $\gamma = 10^{-3}$. The learning rate is 0.025.

Figure 3. The networks have width 100. The initialization scale is $\gamma = 10^{-2}$. The learning rate is 0.1. The networks are trained 20000 epochs. The dataset is generated in the same way as Figure 1 except that the output is generated as $y = \boldsymbol{w}^\top \boldsymbol{x}$.