The Interplay Between Implicit Bias and Adversarial Robustness in Linear Convolutional Neural Networks

Aurélien Boland A.M.M.BOLAND@TUE.NL
Data and AI Cluster & EAISI, Eindhoven University of Technology, The Netherlands

Hannah Pinson H.PINSON@TUE.NL
Data and AI cluster & EAISI, Eindhoven University of Technology, The Netherlands
Data Analytics Laboratory, Vrije Universiteit Brussel, Belgium

Abstract

The vulnerability of neural networks to adversarial examples is an important concern in machine learning. Despite active research on attack and defense algorithms, we lack a clear understanding of the origin of this vulnerability. This study provides a theoretical analysis of the relationship between the architecture of neural networks and their robustness to adversarial attacks, focusing on linear Convolutional Neural Networks (CNNs). Using the theory of implicit biases in linear neural networks, we provide a mathematical characterization of how kernel size and network depth affect adversarial robustness, deriving upper and lower bounds that outline these relationships. Our experiments on popular image datasets align closely with the theoretical trends, allowing us to conclude that the robustness of linear CNN to adversarial attacks decreases with the kernel size and depth. Moreover, our theory strengthens the bridge between implicit bias and robustness, laying the groundwork to further explore robustness from this perspective.

1. Introduction

Neural networks, despite their widespread use and success, have been shown to be vulnerable to adversarial attacks [22]. These attacks involve crafting small, often imperceptible perturbations to input data that cause the model to produce incorrect outputs. Since the discovery of this phenomenon, different attacks as well as defense mechanisms have been studied [13, 15, 18]. Although some work in the field focuses on developing state-of-the-art methods, there is also growing interest in understanding the origin of these perturbations [4, 17, 23, 26]. Recent work has investigated how the implicit bias of gradient-based training algorithms—that is, their tendency to prefer certain solutions among many that fit the data equally well—shapes robustness to adversarial attacks [4, 5, 14].

In this work we investigate the impact of convolution operations on the robustness of linear neural networks. More precisely, we study through the lens of implicit bias theory how the filter sizes and depth of linear CNN influence the robustness of the learned solution. Our contribution can be summarized as follows: (1) We provide lower and upper bounds on the implicit bias of linear convolutional neural networks with arbitrary depths and kernel sizes, extending the works of Dai et al. [3] and Jagadeesan et al. [10] that studied varying kernel sizes with a fixed depth of one; (2) We demonstrate how these bounds on the implicit bias can be translated to bounds on the robustness of the model, with our main result showing that the lower bound on the robustness of linear CNN measured with the ℓ_2-norm decreases with the kernels sizes and network depth; and (3) We show

© A. Boland & H. Pinson.

experimentally that this decreasing bound on the robustness of the model correlates with an actual decrease in the robustness.

To the best of our knowledge, the connection between implicit bias and adversarial robustness has first been studied by Faghri et al. [4]. They demonstrated that the architecture of a linear neural network as well as the optimization method influence the norm under which the model achieves optimal robustness. Frei et al. [5] showed that while some solutions for shallow MLPs with ReLU minimize loss and maximize ℓ_2-robustness under the assumption that data is generated by a mixture of gaussian distributions, gradient descent favors less robust ones. However, Min and Vidal [14] found that using polynomial ReLU activations instead leads gradient descent to maximally robust solutions. For a more in-depth review of the work on adversarial robustness and implicit bias see Appendix A.

2. Preliminary

Adversarial attacks aim to alter machine learning model outputs by introducing small perturbations to input data. For a binary classifier represented by a function $f : \mathbb{R}^n \to \mathbb{R}$ where the predicted class is given by the sign of $f(\boldsymbol{x})$, we define adversarial robustness in terms of the minimum perturbation needed to change the model's prediction.

Definition 1 (Adversarial robustness) *Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and a dataset $D = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=0}^{N-1}$, the adversarial robustness $\epsilon(f)$ is defined as:*

$$\epsilon(f|D) = \min_{(\boldsymbol{x},y)\in D} \|\boldsymbol{\eta}(f|\boldsymbol{x})\|_2 \tag{1}$$

where $\boldsymbol{\eta}(f|\boldsymbol{x})$ is the optimal adversarial perturbation defined as:

$$\boldsymbol{\eta}(f|\boldsymbol{x}) \in \arg\min_{\boldsymbol{\eta}\in\mathbb{R}^n} \|\boldsymbol{\eta}\|_2 \qquad s.t. \qquad f(\boldsymbol{x}+\boldsymbol{\eta}) = 0 \tag{2}$$

The definition of adversarial robustness presented above corresponds to the robustness metric that is maximized in the context of maximally robust classifier [1]. In this work, we study the interplay between robustness and implicit bias in the context of linear CNNs that we define as:

Definition 2 (1D Linear CNN) *We define a 1D linear CNN $f(x)$ with L convolutional layers of kernel sizes $(s_l)_{l=0}^{L-1}$ and input size n as:*

$$\begin{cases} f(x) & = \boldsymbol{w}^\intercal \boldsymbol{h}^{(L)}(\boldsymbol{x}), \\ \boldsymbol{h}^{(l)}(\boldsymbol{x}) & = \boldsymbol{k}^{(l-1)} \circledast \boldsymbol{h}^{(l-1)}(\boldsymbol{x}) \quad \forall l \in [1, L], \\ \boldsymbol{h}^{(0)}(\boldsymbol{x}) & = \boldsymbol{x}. \end{cases} \tag{3}$$

where $\circledast$ corresponds to the circular convolution operation[1]. Moreover, given that a linear CNN is a linear model, we can define a linear predictor $\boldsymbol{\beta}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right)$ such that:

$$f(\boldsymbol{x}) = \boldsymbol{\beta}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right)^\intercal \boldsymbol{x} \tag{4}$$

and $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{k}^{(L-1)}, ..., \boldsymbol{k}^{(0)}) \in \mathbb{R}^{n+\sum_{l=0}^{L-1} s_l}$ represents the learnable parameters.

1. see Appendix B for the definition of this operation

For simplicity, we define linear CNN in the case of one dimensional inputs in this section. The extension to the two-dimensional case (*2D linear CNN*) is provided in Appendix B.

We consider a linear CNN trained on a linearly separable dataset using gradient descent with exponential loss. In this setting, there exist multiple linear predictors $\boldsymbol{\beta}$ that perfectly separate the data, and in particular, multiple directions $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}/\|\boldsymbol{\beta}\|_2$ that achieve zero classification error. Despite this non-uniqueness, gradient descent does not converge to an arbitrary solution. Instead, while the norm $\|\boldsymbol{\beta}(\boldsymbol{\theta})\|_2$ diverges as training progresses, the direction $\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta})$ converges to a specific limit [11]. This directional convergence reflects the implicit bias of the optimization process. In the case of exponential loss, the limiting direction corresponds to the maximum ℓ_2 margin classifier in parameter space ($\boldsymbol{\theta}$).

Theorem 3 (Implicit bias of linear CNNs, adapted from [8, 21]) *Consider a linear CNN as defined in Definition 2 with kernel sizes $(s_l)_{l=0}^{L-1}$ trained on a linearly separable dataset $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=0}^{N-1}$ using gradient descent with the exponential loss function $\ell(u, y) = \exp(-uy)$. Assuming the loss converges to zero and the gradients converge in direction (detailed assumptions in Appendix C), the linear predictor $\boldsymbol{\beta}\left(\boldsymbol{\theta}^{(t)}\middle|(s_l)_{l=0}^{L-1}\right)$ converges in direction to $\overline{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}$, where:*

$$\overline{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}} \in \arg\min_{\boldsymbol{\beta}} \mathcal{R}\left(\boldsymbol{\beta}\middle|(s_l)_{l=0}^{L-1}\right) \quad s.t. \quad \forall i \in [0, N-1], y^{(i)}\boldsymbol{\beta}^{\intercal}\boldsymbol{x}^{(i)} \geq 1 \tag{5}$$

with $\mathcal{R}\left(\boldsymbol{\beta}\middle|(s_l)_{l=0}^{L-1}\right)$ representing the minimum parameter norm required to realize $\boldsymbol{\beta}$:

$$\mathcal{R}\left(\boldsymbol{\beta}\middle|(s_l)_{l=0}^{L-1}\right) := \min_{\boldsymbol{\theta}\in\mathbb{R}^p} \|\boldsymbol{\theta}\|_2^2 \quad s.t. \quad \boldsymbol{\beta} = \boldsymbol{\beta}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right) \tag{6}$$

For simplicity, we restricted Theorem 3 to the case of linear CNNs but the results from [8, 21] apply to any homogeneous neural network (see Appendix C for more details).

3. Robustness Bounds from Implicit Bias in Linear CNNs

In this section, we investigate the relationship between the implicit bias of a linear CNN and its robustness. One advantage of studying robustness using linear models is that the optimal adversarial attack (see Definition 1) is well defined and can be computed analytically. Following standard vector calculus, the robustness of a trained linear CNN can be expressed as:

Lemma 4 (Robustness of a trained linear CNN) *Consider a linear CNN with kernel sizes $(s_l)_{l=0}^{L-1}$, trained on a dataset $D = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=0}^{N-1}$ using exponential loss under the conditions of Theorem 3. Let f_t denote the linear CNN at training iteration t. The robustness of the trained linear CNN is defined as:*

$$\bar{\epsilon}\left((s_l)_{l=0}^{L-1}\right) := \lim_{t\to\infty} \epsilon(f_t|D) \tag{7}$$

and is equal to:

$$\bar{\epsilon}\left((s_l)_{l=0}^{L-1}\right) = \frac{1}{\left\|\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right\|_2} \tag{8}$$

By studying the properties of the induced regularizer $\mathcal{R}\left(\boldsymbol{\beta}\middle|(s_l)_{l=0}^{L-1}\right)$ (Equation 6), we obtain the following lemma:

Lemma 5 (Bounds on the induced regularizer of a 1D linear CNN) *Let $\mathcal{R}\left(\boldsymbol{\beta}\middle|(s_l)_{l=0}^{L-1}\right)$ be the induced regularizer of a 1D linear CNN (Equation 6), we have $\forall (s_l)_{l=0}^{L-1} \in [1,n]^L, \boldsymbol{\beta} \in \mathbb{R}^p$:*

$$\sqrt{\frac{1}{\prod_{l=0}^{L-1} s_l}}\|\boldsymbol{\beta}\|_2 \leq n^{-\frac{L}{2}} \left(\frac{\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right)}{L+1}\right)^{\frac{L+1}{2}} \leq \|\boldsymbol{\beta}\|_2 \tag{9}$$

When $L = 1$, the result of Lemma 5 corresponds to the bounds derived by Dai et al. [3] and Jagadeesan et al. [10]. We prove the extension of this result to an arbitrary depth in Appendix E.4. Finally, by combining Lemma 4 and Lemma 5 we obtain the following theorem about the adversarial robustness of linear CNN:

Theorem 6 (Adversarial robustness bounds of a linear CNN) $\forall (s_l)_{l=0}^{L-1} \in [1,n]^L$:

$$\sqrt{\frac{1}{\prod_{l=0}^{L-1} s_l}}\,\bar{\epsilon}\big((1)_{l=0}^{L-1}\big) \;\leq\; \bar{\epsilon}\big((s_l)_{l=0}^{L-1}\big) \;\leq\; \bar{\epsilon}\big((1)_{l=0}^{L-1}\big). \tag{10}$$

In the 2D case, the same upper bound holds and the lower bound becomes $\bar{\epsilon}((s_l)) \geq \left(\prod_{l=0}^{L-1} s_l\right)^{-1} \bar{\epsilon}((1))$.

Theorem 6 establish theoretical bounds on adversarial robustness for linear CNNs, showing an upper bound that remains constant regardless of kernel size (and is tight when all kernels have size one) and a lower bound that decreases as the product of kernel sizes increases.

4. Experiments

In this section, we experimentally validate that the robustness of linear CNNs follows the same decreasing trend as our lower bound obtained in Theorem 6 when kernel sizes increase. In order to test our hypothesis, we train 2D linear CNNs with gradient descent and exponential loss.[2].To align with our theoretical framework, we selected image datasets that are linearly separable. We conducted experiments on the following datasets:

1. **MNIST**: A dataset comprising classes 0 and 1 from the MNIST training set [12]
2. **mini-ImageNet**: A dataset containing 500 images from the house finch class and 500 images from the robin class of ImageNet [20][3]

While the classification tasks are relatively simple (linearly separable), they correspond to tasks with realistic image statistics. Moreover we selected these datasets to represent two different cases of interest. The MNIST dataset provides an underparametrized linear classification task where the number of samples is significantly larger than the input dimension. In contrast, the mini-ImageNet

2. Please note that Theorem 6 applies to the infinite limit of the linear predictor while our experiments correspond to gradient descent applied for a finite number of steps (with a stopping criterion), for more details on that see Appendix F
3. The mini-ImageNet subset is available at https://huggingface.co/datasets/timm/mini-imagenet

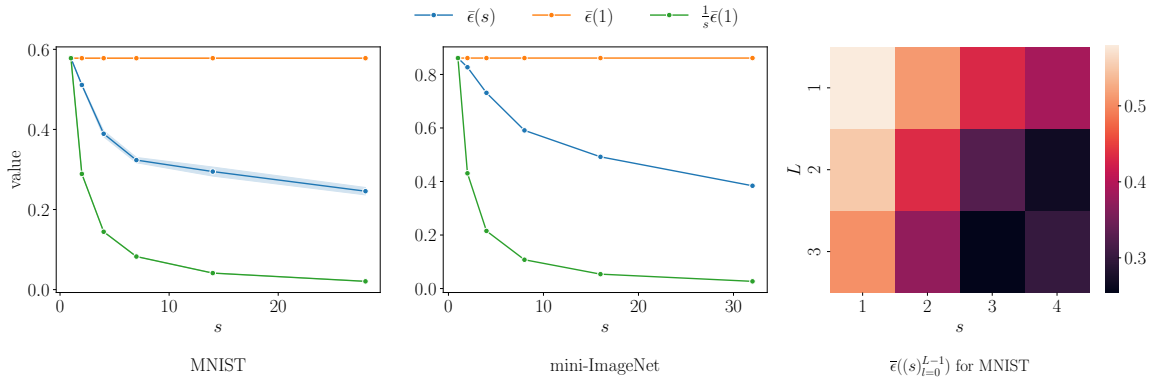


Figure 1: Adversarial robustness (as defined in Definition 1) and the corresponding bounds derived in Theorem 6. For each configuration, a 2D linear CNN is trained with stochastic gradient descent on exponential loss until the cosine similarity between the current linear predictor ($\boldsymbol{\beta}$) and the linear predictor 25 epochs before is smaller than 1.0×10^{-8} (directional convergence). The plots on the left and in the center corresponds to linear CNNs with 1 convolutional layer with kernel size s trained respectively on the 2 classes versions of MNIST and mini-ImageNet. The right plot corresponds to linear CNNs with L convolutional layers of kernel sizes s trained on MNIST. More details on the experimental setup in Appendix F

samples have significantly higher dimension which exceed the number of samples in the dataset (overparametrized). In Fig. 1, we plot the adversarial robustness of linear CNNs trained on these datasets with gradient descent and exponential loss as well as the corresponding bounds from Theorem 6 (see Appendix F for more details on the training setup). The results show that robustness decreases as either kernel size or depth increases, following the same trend as our lower bound $\frac{1}{s^L}\bar{\epsilon}((1)_{l=0}^{L-1})$. This confirms our hypothesis that increasing either parameter negatively impacts robustness.

5. Conclusion

In this work, we studied the adversarial robustness of linear CNNs through the lens of implicit bias. We derived theoretical upper and lower bounds that relate kernel size and network depth to robustness, showing that as these parameters increase, the lower bound on robustness decreases. Our experimental results support these theoretical findings, confirming that larger kernel sizes and deeper networks exhibit reduced robustness to adversarial attacks. These results provide new insights into the interplay between network architecture and adversarial robustness, strengthening the connection between implicit bias and robustness. Although our analysis is centered on linear CNNs, future work could extend these insights to nonlinear architectures and investigate their impact on practical adversarial defense strategies. This study lays the groundwork for a deeper theoretical understanding of robustness in neural networks, paving the way for the development of more resilient deep learning models.

References

[1] Aharon Ben-Tal, Arkadi Nemirovski, and Laurent El Ghaoui. *Robust optimization*. Princeton university press, 2009.

[2] Josue Ortega Caro, Yilong Ju, Ryan Pyle, Sourav Dey, Wieland Brendel, Fabio Anselmi, and Ankit Patel. Local convolutions cause an implicit bias towards high frequency adversarial examples. *ArXiv preprint*, abs/2006.11440, 2020.

[3] Zhen Dai, Mina Karzand, and Nathan Srebro. Representation costs of linear neural networks: Analysis and design. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 26884–26896, 2021.

[4] Fartash Faghri, Sven Gowal, Cristina Vasconcelos, David J Fleet, Fabian Pedregosa, and Nicolas Le Roux. Bridging the gap between adversarial robustness and optimization bias, 2021. Presented at the ICLR 2021 Workshop on Security and Safety in Machine Learning Systems.

[5] Spencer Frei, Gal Vardi, Peter L. Bartlett, and Nati Srebro. The double-edged sword of implicit bias: Generalization vs. robustness in relu networks. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[7] Suriya Gunasekar, Jason D. Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1827–1836. PMLR, 2018.

[8] Suriya Gunasekar, Jason D. Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9482–9491, 2018.

[9] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 125–136, 2019.

[10] Meena Jagadeesan, Ilya Razenshteyn, and Suriya Gunasekar. Inductive bias of multi-channel linear convolutional networks with bounded weight norm. In *Conference on Learning Theory*, pages 2276–2325. PMLR, 2022.

[11] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[14] Hancheng Min and René Vidal. Can implicit bias imply adversarial robustness? In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2574–2582. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.282.

[16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 86–94. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.17.

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Robustness of classifiers to universal perturbations: A geometric perspective. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[18] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4970–4979. PMLR, 2019.

[19] Hannah Pinson, Joeri Lenaerts, and Vincent Ginis. Linear cnns discover the statistical structure of the dataset using only the most dominant frequencies. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 27876–27906. PMLR, 2023.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[21] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, and Nathan Srebro. The implicit bias of gradient descent on separable data. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[23] Thomas Tanay and Lewis Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *ArXiv preprint*, abs/1608.07690, 2016.

[24] Yusuke Tsuzuku and Issei Sato. On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 51–60. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00014.

[25] Gal Vardi. On the implicit bias in deep-learning algorithms. *Communications of the ACM*, 66 (6):86–93, 2023.

[26] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019.

Appendix A. Additional related work

Theory on adversarial attacks In early work on adversarial attacks, Goodfellow et al. [6] showed that the presence of adversarial attacks does not require complex highly nonlinear models. Indeed, they demonstrated that these attacks are also present in linear models and hypothesized that they originate from the high dimensionality of the input. However, this hypothesis has been challenged by Tanay and Griffin [23] who showed that increasing the resolution of input images does not lead to a significantly smaller perturbation. They showed that the sensitivity of a network can be studied through the angle between the decision boundary and the submanifold of input data. Moosavi-Dezfooli et al. [16] observed the presence of a single small perturbation called Universal Adversarial Perturbations (UAP) that can fool a model for most of the input. The presence of these perturbations has been studied through the geometry of the decision boundary [17]. The authors showed that UAP can be explained by the curvature of the decision boundary. Ilyas et al. [9] study the vulnerability of machine learning models towards adversarial example through the scope of non robust features that generalize well. They show that some features learned by a model may generalize well to unseen data while being sensible to small adversarial signal.

Implicit bias and adversarial robustness The implicit bias has been studied for a large corpus of neural network architectures, training algorithms and losses (see Vardi [25] for a survey). In the context of full-width linear CNNs, i.e. linear CNNs where the convolution kernel sizes correspond to the input size, Gunasekar et al. [8] demonstrated that the model trained with exponential loss is biased towards a solution that minimize the $\ell_{\frac{2}{1+L}}$ norm in the Fourier space where L is the network depth[4]. The work has been extended to the case of multichannel CNNs by Jagadeesan et al. [10]. Faghri et al. [4] leveraged the characterization of the implicit bias in linear neural networks to analyze their sensitivity to adversarial attacks. Specifically, they demonstrated that the architecture of a linear neural network influences the norm under which the model achieves optimal robustness. For example, a dense linear neural network is shown to be optimal with respect to the ℓ_2-norm, whereas a single hidden layer full-width linear CNNs achieves optimality in terms of the ℓ_∞-norm in the Fourier space. Caro et al. [2] showed empirically that CNNs with small kernels are biased towards adversarial examples with higher frequency components than their full kernel size counterparts. They consider CNNs defined in a similar way as the linear CNN that we define later in this paper and also they considered the case where ReLU activations are added. They describe their findings as a result from the work on the implicit bias of linear CNNs done by Gunasekar et al. [8] and the Fourier uncertainty principle. Relying on the fact that CNNs are biased towards solution that are sparse in the Fourier space [8, 19], Tsuzuku and Sato [24] presented a simple but effective method to create UAP composed of a single frequency. Recent works have examined the robustness of non-linear models through implicit bias. Frei et al. [5] showed that while some solutions for shallow MLPs with ReLU minimize loss and maximize ℓ_2-robustness, gradient descent favors less robust ones. However, Min and Vidal [14] found that using polynomial ReLU activations instead leads gradient descent to maximally robust solutions.

4. measured by the number of hidden layers

Appendix B. Definitions of circular convolution and 2D linear CNN

Definition 7 (circular convolution) *The circular convolution operation is represented by the symbol $\circledast$ and defined as:*

$$\forall i \in [0, n-1] : (\boldsymbol{k} \circledast \boldsymbol{h})_i = \frac{1}{\sqrt{n}} \sum_{j=0}^{s-1} \boldsymbol{k}_j \boldsymbol{h}_{(i+j) \bmod n} \tag{11}$$

where $\boldsymbol{k} \in \mathbb{R}^s$ represents the kernel and $\boldsymbol{h} \in \mathbb{R}^n$ represents the output of the previous layer. The factor $\frac{1}{\sqrt{n}}$ is used to make the derivation of the implicit bias cleaner.

To define a 2D linear CNN, we still consider vectors as input. The input corresponds to the flattened version of the image. Formally, given an image $\boldsymbol{X} \in \mathbb{R}^{n \times n}$, we define its flattened representation $\boldsymbol{x} \in \mathbb{R}^{n^2}$ as:

$$\forall i, j \in [0, n-1]^2 : \boldsymbol{x}_{ni+j} = \boldsymbol{X}_{i,j} \tag{12}$$

By defining a 2D linear CNN in a form that closely resembles a 1D linear CNN, we can seamlessly translate the theory developed for the 1D case to the 2D case. A key aspect of this approach is the ability to express the model as $f(\boldsymbol{x}) = \boldsymbol{\beta}\left(\boldsymbol{\theta} \middle| (s_l)_{l=0}^{L-1}\right)^\intercal \boldsymbol{x}$. In the case of matrices, the circular convolution can be extended as:

$$\begin{aligned}(\boldsymbol{K} \circledast \boldsymbol{H})_{i_1,i_2} = \frac{1}{n} \sum_{j_1=0}^{s-1} \sum_{j_2=0}^{s-1} \Big(& \boldsymbol{K}_{j_1,j_2} \\ & \boldsymbol{H}_{(i_1+j_1) \bmod n, (i_2+j_2) \bmod n} \Big)\end{aligned} \tag{13}$$

where $\boldsymbol{K} \in \mathbb{R}^{s \times s}$ represents the kernel, and $\boldsymbol{H} \in \mathbb{R}^{n \times n}$. Let $\boldsymbol{k} \in \mathbb{R}^{s^2}$ and $\boldsymbol{h} \in \mathbb{R}^{n^2}$ represent the flattened versions of $\boldsymbol{K}$ and $\boldsymbol{H}$, respectively. We then introduce a new operator $\circledast_{2D}$ that enables us to restrict the computation to vector operations:

$$\forall i, j \in [0, n-1] : (\boldsymbol{k} \circledast_{2D} \boldsymbol{h})_{ni+j} = (\boldsymbol{K} \circledast \boldsymbol{H})_{i,j} \tag{14}$$

Thanks to this new operator, we can define the 2D linear CNN in a similar way as the 1D counterpart:

Definition 8 (2D Linear CNN) *We define a 2D linear CNN $f(x)$ with L convolutional layers of kernel sizes $(s_l)_{l=0}^{L-1}$ and input size n^2 as:*

$$\begin{cases} f(x) & = \boldsymbol{w}^\intercal \boldsymbol{h}^L(\boldsymbol{x}), \\ \boldsymbol{h}^{(l)}(\boldsymbol{x}) & = \boldsymbol{k}^{(l-1)} \circledast_{2D} \boldsymbol{h}^{(l-1)}(\boldsymbol{x}) \quad \forall l \in [1, L], \\ \boldsymbol{h}^0(\boldsymbol{x}) & = \boldsymbol{x}. \end{cases} \tag{15}$$

Here, $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{k}^{L-1}, ..., \boldsymbol{k}^0) \in \mathbb{R}^{n^2 + \sum_{l=0}^{L-1} s_l^2}$ represents the learned parameters. The corresponding linear predictor $\boldsymbol{\beta}\left(\boldsymbol{\theta} \middle| (s_l)_{l=0}^{L-1}\right) \in \mathcal{R}^{n^2}$ is defined such that:

$$f(\boldsymbol{x}) = \boldsymbol{\beta}\left(\boldsymbol{\theta} \middle| (s_l)_{l=0}^{L-1}\right)^\intercal \boldsymbol{x} \tag{16}$$

Appendix C. Implicit bias of linear CNNs

In the context of this work, we examine the implicit bias of a neural network trained with exponential loss. An important result in this scenario concerns the case of homogeneous neural networks.

Definition 9 (Homogeneous neural network) *Consider a neural network $\boldsymbol{f_\theta} : \mathbb{R}^n \to \mathbb{R}^m$ where $\boldsymbol{\theta} \in \mathbb{R}^p$ corresponds to the learned parameters. The neural network is homogeneous if there exists $L > 0$ such that for every $\alpha > 0$, $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{\theta} \in \mathbb{R}^p$: $\boldsymbol{f}_{\alpha\boldsymbol{\theta}}(\boldsymbol{x}) = \alpha^L \boldsymbol{f_\theta}(\boldsymbol{x})$*

We restrict our study to the case of neural networks trained with (vanilla) gradient descent. Work on implicit bias with other optimization methods can be found in the literature [7] but this goes beyond the scope of our work. If we consider the task of binary classification where the predicted class for $\boldsymbol{x}$ corresponds to the sign of $f_{\boldsymbol{\theta}}(\boldsymbol{x})$. Then we have the following results concerning the implicit bias of a homogeneous neural network:

Theorem 10 (Implicit bias of homogeneous neural networks, reformulation of Gunasekar et al. [8]) *Let $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ be a homogeneous neural network with the learnable parameters $\boldsymbol{\theta} \in \mathbb{R}^p$. For almost all datasets $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=0}^{N-1}$ separable by $\mathcal{B} := \{f_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathbb{R}^p\}$, almost all initializations $\boldsymbol{\theta}^{(0)}$, and any bounded sequence of step sizes $\{\eta_t\}_t$, consider the sequence of iterates $\boldsymbol{\theta}^{(t)}$ obtained using gradient descent with the exponential loss function $\ell(u, y) = \exp(-uy)$ associated to the empirical loss $L(\boldsymbol{\theta})$. If the following conditions hold:*

1. *The iterates $\boldsymbol{\theta}^{(t)}$ asymptotically minimize the objective, i.e., $L(\boldsymbol{\theta}^{(t)}) \to 0$,*
2. *$\boldsymbol{\theta}^{(t)}$, and consequently the predictors $f_{\boldsymbol{\theta}^{(t)}}$, converge in direction to yield a separator with positive margin,*
3. *The gradients with respect to the predictors, $\nabla_{f_{\boldsymbol{\theta}}} L(\boldsymbol{\theta}^{(t)})$, converge in direction,*

then the limit direction of the parameters

$$\overline{\boldsymbol{\theta}}^{\infty} = \lim_{t\to\infty} \frac{\boldsymbol{\theta}^{(t)}}{\|\boldsymbol{\theta}^{(t)}\|_2} \tag{17}$$

is a positive scaling of a first-order stationary point of the following optimization problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \|\boldsymbol{\theta}\|_2^2 \quad s.t. \quad \forall i \in [0, N-1],\ y^{(i)} f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \geq 1. \tag{18}$$

For completeness regarding Theorem 10, we give here a definition of gradient descent on empirical loss.

Definition 11 (Gradient descent on empirical loss) *Given a parametrized function $\boldsymbol{f_\theta} : \mathbb{R}^n \to \mathbb{R}^m$, where $\boldsymbol{\theta} \in \mathbb{R}^p$, a dataset $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=0}^{N-1}$ with $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$ and $y^{(i)} \in \mathcal{Y}$, and a loss function $l : \mathbb{R}^m \times \mathcal{Y} \to \mathbb{R}$, the empirical loss is:*

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} l\big(\boldsymbol{f_\theta}(\boldsymbol{x}^{(i)}), y^{(i)}\big). \tag{19}$$

Gradient descent updates the parameters iteratively according to:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t), \tag{20}$$

where $\eta_t > 0$ is the learning rate at iteration t, and $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) = \frac{1}{N}\sum_{i=0}^{N-1} \nabla_{\boldsymbol{\theta}} l\big(\boldsymbol{f}_{\boldsymbol{\theta}_t}(\boldsymbol{x}^{(i)}), y^{(i)}\big)$ is the gradient of the loss.

The Theorem 3 is reformulation of Theorem 10 in the case of binary linear classifier. If $f_{\boldsymbol{\theta}}(x) = \boldsymbol{\beta}(\boldsymbol{\theta})^\intercal \boldsymbol{x}$, then the optimization problem of Theorem 10 becomes :

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \|\boldsymbol{\theta}\|_2^2 \quad \text{s.t.} \quad \forall i \in [0, N-1],\, y^{(i)}\boldsymbol{\beta}(\boldsymbol{\theta})^\intercal \boldsymbol{x}^{(i)} \geq 1. \tag{21}$$

To prove the reformulation in Theorem 3 we have to show the equivalence of the optimization problem above with the optimization problem of the Theorem 3:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^n} \mathcal{R}(\boldsymbol{\beta}) \qquad \text{s.t.} \qquad \forall i \in [0, N-1], y^{(i)}\boldsymbol{\beta}^\intercal \boldsymbol{x}^{(i)} \geq 1 \tag{22}$$

More precisely, we have to show that $\tilde{\boldsymbol{\theta}}$ is a solution of equation 21 if and only if $\tilde{\boldsymbol{\beta}} = \boldsymbol{\beta}(\tilde{\boldsymbol{\theta}})$ is a solution of equation 22. To show equivalence between (21) and (22):

1. **Forward Direction ($\Rightarrow$):** Let $\widetilde{\boldsymbol{\theta}}$ solve (21). Define $\widetilde{\boldsymbol{\beta}} = \boldsymbol{\beta}(\widetilde{\boldsymbol{\theta}})$.
   - By constraints: $y^{(i)}\widetilde{\boldsymbol{\beta}}^\intercal \boldsymbol{x}^{(i)} \geq 1$ for all i.
   - By definition of $\mathcal{R}$: $\mathcal{R}(\widetilde{\boldsymbol{\beta}}) \leq \|\widetilde{\boldsymbol{\theta}}\|_2^2$.
   - Suppose $\exists \boldsymbol{\beta}'$ with $\mathcal{R}(\boldsymbol{\beta}') < \mathcal{R}(\widetilde{\boldsymbol{\beta}})$ and $y^{(i)}\widetilde{\boldsymbol{\beta}}^\intercal \boldsymbol{x}^{(i)} \geq 1$ for all i . Then $\exists \boldsymbol{\theta}'$ s.t. $\boldsymbol{\beta}' = \boldsymbol{\beta}(\boldsymbol{\theta}')$ and $\|\boldsymbol{\theta}'\|_2^2 < \|\widetilde{\boldsymbol{\theta}}\|_2^2$, contradicting the optimality of $\widetilde{\boldsymbol{\theta}}$. Thus, $\widetilde{\boldsymbol{\beta}}$ solves (22).
2. **Reverse Direction ($\Leftarrow$):**
   - Let $\widetilde{\boldsymbol{\beta}}$ solve (22). By definition of $\mathcal{R}$, $\exists \widetilde{\boldsymbol{\theta}}$ s.t. $\boldsymbol{\beta}(\widetilde{\boldsymbol{\theta}}) = \widetilde{\boldsymbol{\beta}}$ and $\|\widetilde{\boldsymbol{\theta}}\|_2^2 = \mathcal{R}(\widetilde{\boldsymbol{\beta}})$.
   - $\widetilde{\boldsymbol{\theta}}$ satisfies $y^{(i)}\boldsymbol{\beta}(\widetilde{\boldsymbol{\theta}})^\intercal \boldsymbol{x}^{(i)} \geq 1$ for all i.
   - Suppose $\exists \boldsymbol{\theta}'$ with $\|\boldsymbol{\theta}'\|_2^2 < \|\widetilde{\boldsymbol{\theta}}\|_2^2$ and $y^{(i)}\boldsymbol{\beta}(\boldsymbol{\theta}')^\intercal \boldsymbol{x}^{(i)} \geq 1$ for all i. Then $\boldsymbol{\beta}(\boldsymbol{\theta}')$ would satisfy $\mathcal{R}(\boldsymbol{\beta}(\boldsymbol{\theta}')) \leq \|\boldsymbol{\theta}'\|_2^2 < \mathcal{R}(\widetilde{\boldsymbol{\beta}})$, contradicting the optimality of $\widetilde{\boldsymbol{\beta}}$. Thus, $\widetilde{\boldsymbol{\theta}}$ solves (21).

Hence, $\widetilde{\boldsymbol{\theta}}$ solves (21) $\iff$ $\widetilde{\boldsymbol{\beta}} = \boldsymbol{\beta}(\widetilde{\boldsymbol{\theta}})$ solves (22).

Appendix D. Additional Lemmas

Lemma 12 (Robustness of a linear model) *let $f(\boldsymbol{x}) = \boldsymbol{\beta}^\intercal \boldsymbol{x}$, then we have*

$$\epsilon(f|D) = \min_{x\in D} \frac{|\boldsymbol{\beta}^\intercal \boldsymbol{x}|}{\|\boldsymbol{\beta}\|_2 \|\boldsymbol{x}\|_2} \|\boldsymbol{x}\|_2 = \min_{x\in D} |\cos(\boldsymbol{\beta}, \boldsymbol{x})|\, \|\boldsymbol{x}\|_2 \tag{23}$$

where $\cos(\cdot,\cdot)$ corresponds to cosine similarity.

Lemma 13 (General form of the induced regularizer) *In the case of a 1D linear we have:*

$$\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = (L+1)\left(\inf_{\substack{\forall l\in[0,L-1]:\\ v^{(l)}\in\mathbb{R}^{s_l},\\ \|v^{(l)}\|=1}} \sum_{i=0}^{n-1} \frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\widehat{\boldsymbol{v}^{(l)}}_i\right|^2}\right)^{\frac{1}{L+1}} \tag{24}$$

In the case of a 2D linear we have:

$$\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = (L+1)\left(\inf_{\substack{\forall l\in[0,L-1]:\\ v^{(l)}\in\mathbb{R}^{s_l^2},\\ \|v^{(l)}\|=1}} \sum_{i=0}^{n^2-1} \frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\widehat{\boldsymbol{v}^{(l)}}_i\right|^2}\right)^{\frac{1}{L+1}} \tag{25}$$

where $\widehat{\boldsymbol{\beta}}$ is the Fourier transform of $\boldsymbol{\beta}$ (see Definitions 14 and 15).

Appendix E. Proofs Lemmas and Theorem

E.1. Proof Lemma 12

First we need to show that:

$$\boldsymbol{\eta}(f|\boldsymbol{x}) = -\left(\frac{\boldsymbol{\beta}^\intercal\boldsymbol{x}}{\|\boldsymbol{\beta}\|_2^2}\right)\boldsymbol{\beta} \quad \text{with} \quad f(\boldsymbol{x}) = \boldsymbol{\beta}^\intercal\boldsymbol{x} \tag{26}$$

Using the definition 1, we have $\boldsymbol{\eta}(f|\boldsymbol{x})$ is a solution of:

$$\arg\min \|\boldsymbol{\eta}\|_2 \quad \text{s.t.} \quad \boldsymbol{\beta}^\intercal(\boldsymbol{x}+\boldsymbol{\eta}) = 0 \tag{27}$$

Given a ℓ_2-norm constraint on $\boldsymbol{\eta}$, we have that then absolute value product $\boldsymbol{\beta}^\intercal\boldsymbol{\eta}$ is maximal if $\boldsymbol{\beta}$ and $\boldsymbol{\eta}$ are parallel. So, we have:

$$\exists\alpha\in\mathbb{R} : \boldsymbol{\eta} = \alpha\boldsymbol{\beta} \tag{28}$$

The last step is to find the value of α. This can be done by using the constraint in equation 27:

$$\boldsymbol{\beta}^\intercal(\boldsymbol{x}+\boldsymbol{\eta}) = \boldsymbol{\beta}^\intercal\boldsymbol{x} + \alpha\|\boldsymbol{\beta}\|_2^2 = 0 \tag{29}$$

$$\alpha = -\frac{\boldsymbol{\beta}^\intercal\boldsymbol{x}}{\|\boldsymbol{\beta}\|_2^2} \tag{30}$$

$$\boldsymbol{\eta}(f|\boldsymbol{x}) = \alpha\boldsymbol{\beta} = -\left(\frac{\boldsymbol{\beta}^\intercal\boldsymbol{x}}{\|\boldsymbol{\beta}\|_2^2}\right)\boldsymbol{\beta} \tag{31}$$

Now that we have the optimal adversarial attack for a given input ($\boldsymbol{\eta}(f|\boldsymbol{x})$), we can derive the corresponding adversarial robustness for a dataset $D = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=0}^{N-1}$:

$$\epsilon(f|D) = \min_{x\in D} \|\boldsymbol{\eta}(f|\boldsymbol{x})\|_2 \tag{32}$$

$$= \min_{x\in D} \left\|-\left(\frac{\boldsymbol{\beta}^\intercal\boldsymbol{x}}{\|\boldsymbol{\beta}\|_2^2}\right)\boldsymbol{\beta}\right\|_2 \tag{33}$$

$$= \min_{x\in D} \frac{|\boldsymbol{\beta}^\intercal\boldsymbol{x}|}{\|\boldsymbol{\beta}\|_2^2}\|\boldsymbol{\beta}\|_2 \tag{34}$$

$$= \frac{1}{\|\boldsymbol{\beta}\|_2}\min_{\boldsymbol{x}\in D}|\boldsymbol{\beta}^\intercal\boldsymbol{x}| \tag{35}$$

$$= \min_{x\in D} \frac{|\boldsymbol{\beta}^\intercal\boldsymbol{x}|}{\|\boldsymbol{\beta}\|_2\|\boldsymbol{x}\|_2}\|\boldsymbol{x}\|_2 \tag{36}$$

$$= \min_{x\in D} |\cos(\boldsymbol{\beta},\boldsymbol{x})|\,\|\boldsymbol{x}\|_2 \tag{37}$$

□

E.2. Proof Lemma 4

We observe in Lemma 12 that the robustness of a linear model does not depends on the amplitude of $\boldsymbol{\beta}$ but on the angle of $\boldsymbol{\beta}$ with some input $\boldsymbol{x}$ (cosine similarity). Because a linear CNN with kernel sizes $(s_l)_{l=0}^{L-1}$ converge in direction to $\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}$ (Theorem 3), we have:

$$\bar{\epsilon}\left((s_l)_{l=0}^{L-1}\right) = \min_{x\in D}\left|\cos\left(\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}, \boldsymbol{x}\right)\right| \|\boldsymbol{x}\|_2 \tag{38}$$

$$= \frac{1}{\left\|\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right\|_2} \min_{\boldsymbol{x}\in D}\left|\left(\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right)^\intercal \boldsymbol{x}\right| \tag{39}$$

The last step in order to proof our lemma is to proof that $\min_{\boldsymbol{x}\in D}\left|\left(\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right)^\intercal \boldsymbol{x}\right| = 1$. By definition (see Theorem 3), $\min_{\boldsymbol{x}\in D}\left|\left(\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right)^\intercal \boldsymbol{x}\right|$ is a solution of:

$$\min_{\boldsymbol{\beta}} \mathcal{R}\left(\boldsymbol{\beta}\middle|(s_l)_{l=0}^{L-1}\right) \quad \text{s.t.} \quad \forall i \in [0, N-1], y^{(i)}\boldsymbol{\beta}^\intercal \boldsymbol{x}^{(i)} \geq 1 \tag{40}$$

We see that having at least one of the constraint in equation 5 that is tight, i.e. $\exists i \in [0, N-1]$, $y^{(i)}\left(\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right)^\intercal \boldsymbol{x}^{(i)} = 1$, implies that $\min_{\boldsymbol{x}\in D}\left|\left(\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right)^\intercal \boldsymbol{x}\right| = 1$. We will prove by contradiction that at least one constraint is tight.

We assume there exists $\bar{\boldsymbol{\beta}}$ which is a solution of equation 5 such that no constraints are tight:

$$\exists \delta > 0 : \forall i \in [0, N-1],\ y^{(i)}\bar{\boldsymbol{\beta}}^\intercal \boldsymbol{x}^{(i)} \geq 1 + \delta \tag{41}$$

Let's consider $\bar{\boldsymbol{\beta}}' = \frac{1}{1+\delta}\bar{\boldsymbol{\beta}}$, we see that $\bar{\boldsymbol{\beta}}'$ also satisfies the constraints in equation 5. In order to prove that $\bar{\boldsymbol{\beta}}$ cannot be a solution of equation 5, we need to show that $\mathcal{R}\left(\boldsymbol{\beta}\middle|(s_l)_{l=0}^{L-1}\right) > \mathcal{R}\left(\boldsymbol{\beta}'\middle|(s_l)_{l=0}^{L-1}\right)$. Let $\bar{\boldsymbol{\theta}} \in \mathbb{R}^p$ be a solution of equation 6 for the linear predictor $\bar{\boldsymbol{\beta}}$:

$$\bar{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \|\boldsymbol{\theta}\|_2^2 \quad \text{s.t.} \quad \bar{\boldsymbol{\beta}} = \boldsymbol{\beta}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right) \tag{42}$$

Based on the definition of linear CNN we have:

$$\boldsymbol{\beta}\left(\frac{1}{(1+\delta)^{1/L}}\bar{\boldsymbol{\theta}}\middle|(s_l)_{l=0}^{L-1}\right) = \frac{1}{(1+\delta)}\boldsymbol{\beta}\left(\bar{\boldsymbol{\theta}}\middle|(s_l)_{l=0}^{L-1}\right) \tag{43}$$

$$= \frac{1}{(1+\delta)}\bar{\boldsymbol{\beta}} \tag{44}$$

$$= \bar{\boldsymbol{\beta}}' \tag{45}$$

For the induced regularizer we have:

$$\mathcal{R}\left(\bar{\boldsymbol{\beta}}'\middle|(s_l)_{l=0}^{L-1}\right) \geq \left\|\frac{1}{(1+\delta)^{1/L}}\bar{\boldsymbol{\theta}}\right\|_2^2 > \|\bar{\boldsymbol{\theta}}\|_2^2 = \mathcal{R}\left(\bar{\boldsymbol{\beta}}\middle|(s_l)_{l=0}^{L-1}\right) \tag{46}$$

□

E.3. Proof Lemma 13

In order to prove the lemma, we first need to introduce a new representation of the linear CNN using Fourier tramsform

LINEAR CNN IN THE FOURIER SPACE

Definition 14 (1D DFT) *Given a spatial dimension $n \in \mathbb{Z}_+$, we define the 1D DFT of a vector $\boldsymbol{x} \in \mathbb{R}^m$ with $0 < m \leq n$ as $\widehat{\boldsymbol{x}} \in \mathbb{C}^n$ such that:*

$$\forall \mu \in [0, n-1] : \widehat{\boldsymbol{x}}_\mu := \frac{1}{\sqrt{n}} \sum_{s=0}^{m-1} \boldsymbol{x}_s (\omega_n)^{\mu s} \tag{47}$$

with $\omega_n := e^{-\frac{2\pi}{n} i}$ and $i^2 = -1$.

Definition 15 (2D DFT) *Given a spatial dimension $n \in \mathbb{Z}_+$, we define the 2D DFT of a vector $\boldsymbol{x} \in \mathbb{R}^{m^2}$ with $0 < m \leq n$, which represent a flattened image, as $\widehat{\boldsymbol{x}} \in \mathbb{C}^n$ such that:*

$$\forall \mu, \nu \in [0, n-1]^2 : \widehat{\boldsymbol{x}}_{n\mu+\nu} := \frac{1}{n} \sum_{s=0,t=0}^{m-1} \boldsymbol{x}_{ms+t} (\omega_n)^{\mu s} (\omega_n)^{\nu t} \tag{48}$$

with $\omega_n := e^{-\frac{2\pi}{n} i}$ and $i^2 = -1$.

For conciseness, we use the same notation for 1D and 2D fourier transform. If the input represent a 1D signal then 1D DFT is applied and if the input represent a flattened image then 2D DFT is applied. We defined the DFT such that it corresponds to a unitary transform:

$$\forall x \in \mathbb{R}^m : \|\widehat{\boldsymbol{x}}\|_2 = \|\boldsymbol{x}\|_2 \tag{49}$$

Thanks to the convolution theorem [5] we have:

$$\forall \boldsymbol{k} \in \mathbb{R}^m, \boldsymbol{h} \in \mathbb{R}^n : (\widehat{\boldsymbol{k} \circledast \boldsymbol{h}}) = \widehat{\boldsymbol{k}}^* \odot \widehat{\boldsymbol{h}} \qquad \text{(1D case)} \tag{50}$$

$$\forall \boldsymbol{k} \in \mathbb{R}^{m^2}, \boldsymbol{h} \in \mathbb{R}^{n^2} : (\widehat{\boldsymbol{k} \circledast_{2D} \boldsymbol{h}}) = \widehat{\boldsymbol{k}}^* \odot \widehat{\boldsymbol{h}} \qquad \text{(2D case)} \tag{51}$$

where $\widehat{\boldsymbol{k}}^*$ is the complex conjugate of $\widehat{\boldsymbol{k}}$. By applying the results above we obtain a Fourier form for the linear CNN:

Lemma 16 *The linear CNN as defined in definitions 2 and 8 can be rewritten as diagonal neural network in the Fourier space. More precisely, let $f : \mathbb{R}^n \to \mathbb{R}$ be the linear CNN, we have that $\forall \boldsymbol{x} \in \mathbb{R}^n$:*

$$\begin{cases} f(x) & = \langle \widehat{\boldsymbol{w}}, \widehat{\boldsymbol{h}}^{(L)}(\boldsymbol{x}) \rangle, \\ \widehat{\boldsymbol{h}}^{(l)}(\boldsymbol{x}) & = \left(\widehat{\boldsymbol{k}}^{(l-1)}\right)^* \odot \widehat{\boldsymbol{h}}^{(l-1)}(\boldsymbol{x}) \quad \forall l \in [1, L], \\ \widehat{\boldsymbol{h}}^{(0)}(\boldsymbol{x}) & = \widehat{\boldsymbol{x}}. \end{cases} \tag{52}$$

where $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = (\boldsymbol{x}^*)^\intercal \boldsymbol{y}$ is the complex inner product between two complex vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}^n$.

$$f(\boldsymbol{x}) = \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle = \langle \widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{x}} \rangle \tag{53}$$

and:

$$\widehat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}\big|(s_l)_{l=0}^{L-1}\right) = \widehat{\boldsymbol{w}} \odot \widehat{\boldsymbol{k}}^{(L-1)} \odot \ldots \odot \widehat{\boldsymbol{k}}^{(0)} \tag{54}$$

5. For a proof in the 1D case see appendix C of Gunasekar et al. [8]

PROOF

For a given 1D linear CNN with the kernel sizes $(s_l)_{l=0}^{L-1}$, we define $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{k}^{(L-1)}, \dots, \boldsymbol{k}^{(0)}) \in \mathbb{R}^{n+\sum_{l=0}^{L-1} s_l}$.

$$\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = \min_{\boldsymbol{\theta}\in\mathbb{R}^p} \|\boldsymbol{\theta}\|_2^2 \qquad \text{s.t.} \qquad \boldsymbol{\beta} = \boldsymbol{\beta}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right) \tag{55}$$

$$= \min_{\boldsymbol{\theta}\in\mathbb{R}^p} \left(\|\boldsymbol{w}\|_2^2 + \sum_{l=0}^{L-1} \|\boldsymbol{k}^{(l)}\|_2^2\right) \qquad \text{s.t.} \qquad \boldsymbol{\beta} = \boldsymbol{\beta}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right) \tag{56}$$

$$= \min_{\boldsymbol{\theta}\in\mathbb{R}^p} \left(\|\widehat{\boldsymbol{w}}\|_2^2 + \sum_{l=0}^{L-1} \left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2^2\right) \qquad \text{s.t.} \qquad \widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right) \tag{57}$$

By the AM-GM inequality, we have :

$$\|\widehat{\boldsymbol{w}}\|_2^2 + \sum_{l=0}^{L-1} \left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2^2 \geq (L+1)\left(\|\widehat{\boldsymbol{w}}\|_2 \prod_{l=0}^{L-1} \left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2\right)^{\frac{2}{L+1}} \tag{58}$$

Moreover, the bound is tight if $\forall l \in [0, L-1] : \|\widehat{\boldsymbol{w}}\|_2 = \left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2$. An interesting property of $\widehat{\boldsymbol{\beta}}$ is that we can rescale the parameters without changing $\widehat{\boldsymbol{\beta}}$, more precisely:

For a given $\alpha > 0$ and $l \in [0, L-1]$, we define $\widehat{\boldsymbol{w}}' := \alpha\widehat{\boldsymbol{w}}$, $\widehat{\boldsymbol{k}^{(l)}}' := \frac{1}{\alpha}\widehat{\boldsymbol{k}^{(l)}}$ and $\widehat{\boldsymbol{\theta}}' = \left(\widehat{\boldsymbol{w}}', \widehat{\boldsymbol{k}^{(L-1)}}, \dots, \widehat{\boldsymbol{k}^{(l)}}', \dots, \widehat{\boldsymbol{k}^{(0)}}\right)$. We can see that:

$$\widehat{\boldsymbol{\beta}}(\widehat{\boldsymbol{\theta}}') = \alpha\widehat{\boldsymbol{w}} \odot \widehat{\boldsymbol{k}}^{(L-1)} \odot \dots \odot \frac{1}{\alpha}\widehat{\boldsymbol{k}^{(l)}} \odot \dots \widehat{\boldsymbol{k}}^{(0)} \tag{59}$$

$$= \widehat{\boldsymbol{w}} \odot \widehat{\boldsymbol{k}}^{(L-1)} \odot \dots \odot \widehat{\boldsymbol{k}^{(l)}} \odot \dots \widehat{\boldsymbol{k}}^{(0)} \tag{60}$$

$$= \widehat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right) \tag{61}$$

So we can rescale the parameters such that $\forall l \in [0, L-1] : \|\widehat{\boldsymbol{w}}\|_2 = \left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2$ without changing $\widehat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right)$ which is our constraint in equation 57. Thanks to this property, we have:

$$\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = \min_{\boldsymbol{\theta}\in\mathbb{R}^p} (L+1)\left(\|\widehat{\boldsymbol{w}}\|_2 \prod_{l=0}^{L-1} \left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2\right)^{\frac{2}{L+1}} \qquad \text{s.t.} \qquad \widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}\middle|(s_l)_{l=0}^{L-1}\right) \tag{62}$$

Using the definition of $\widehat{\boldsymbol{\beta}}$ and considering $\forall l \in [0, L-1] : \|\widehat{\boldsymbol{w}}\|_2 = \left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2$ we have that :

$$\forall i \in [0, n-1] : \widehat{\boldsymbol{w}}_i = \begin{cases} \dfrac{\widehat{\boldsymbol{\beta}}_i}{\prod_{l=0}^{L-1} \widehat{k^{(l)}}_i} & \text{if } \widehat{\boldsymbol{\beta}}_i \neq 0 \\ 0 & \text{else.} \end{cases} \tag{63}$$

If we insert equation 63 in equation 62:

$$\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = (L+1)\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left(\prod_{l=0}^{L-1}\left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2\right)^{\frac{2}{L+1}}\left(\sum_{i\in supp(\widehat{\boldsymbol{\beta}})}\frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\widehat{k^{(l)}}_i\right|^2}\right)^{\frac{1}{L+1}} \quad \text{s.t.} \quad \widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}|(s_l)_{l=0}^{L-1}\right) \tag{64}$$

where $supp(\widehat{\boldsymbol{\beta}}) = \{i : \widehat{\boldsymbol{\beta}}_i \neq 0\}$. If we define:

$$\forall l \in [0, L-1] : \boldsymbol{v}^{(l)} := \frac{\boldsymbol{k}^{(l)}}{\|\boldsymbol{k}^{(l)}\|_2} \tag{65}$$

Then we can easily show that $\forall l \in [0, L-1]$:

$$\|\boldsymbol{v}^{(l)}\|_2 = \|\widehat{\boldsymbol{v}^{(l)}}\|_2 = 1 \tag{66}$$

$$\widehat{\boldsymbol{v}^{(l)}} = \frac{\widehat{\boldsymbol{k}^{(l)}}}{\|\widehat{\boldsymbol{k}^{(l)}}\|_2} \tag{67}$$

$$\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = (L+1)\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left(\prod_{l=0}^{L-1}\left\|\widehat{\boldsymbol{k}^{(l)}}\right\|_2\right)^{\frac{2}{L+1}}\left(\sum_{i\in supp(\widehat{\boldsymbol{\beta}})}\frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\|\widehat{\boldsymbol{k}^{(l)}}\|_2^2\left|\widehat{\boldsymbol{v}^{(l)}}_i\right|^2}\right)^{\frac{1}{L+1}} \tag{68}$$

$$\text{s.t.} \quad \widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}|(s_l)_{l=0}^{L-1}\right)$$

$$= (L+1)\left(\min_{\substack{\forall l\in[0,L-1]:\\ v^{(l)}\in\mathbb{R}^{s_l},\\ \|v^{(l)}\|=1}}\sum_{i\in supp(\widehat{\boldsymbol{\beta}})}\frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\widehat{\boldsymbol{v}^{(l)}}_i\right|^2}\right)^{\frac{1}{L+1}} \tag{69}$$

$$= (L+1)\left(\inf_{\substack{\forall l\in[0,L-1]:\\ v^{(l)}\in\mathbb{R}^{s_l},\\ \|v^{(l)}\|=1}}\sum_{i=0}^{n-1}\frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\widehat{\boldsymbol{v}^{(l)}}_i\right|^2}\right)^{\frac{1}{L+1}} \tag{70}$$

For the case of a 2D linear CNN, the proof is almost the same except that we have vector of dimension n^2 instead of n. □

E.4. Proof Lemma 5

Now that we have obtained a more explicit form for $\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right)$ (see Lemma 13), we will be able to derive our bounds. To simplify the notations we will compute bounds of $\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right)$ which is defined as:

$$\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = n^{-\frac{L}{2}}\left(\frac{\mathcal{R}\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right)}{L+1}\right)^{\frac{L+1}{2}} \tag{71}$$

The bounds to prove then become:

$$\sqrt{\frac{1}{\prod_{l=0}^{L-1} s_l}}\|\boldsymbol{\beta}\|_2 \leq \mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) \leq \|\boldsymbol{\beta}\|_2 \tag{72}$$

Thanks to Lemma 13, we have:

$$\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = n^{-\frac{L}{2}} \sqrt{\inf_{\substack{\forall l\in[0,L-1]:\\ v^{(l)}\in\mathbb{R}^{s_l},\\ \|v^{(l)}\|=1}} \sum_{i=0}^{n-1} \frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\widehat{\boldsymbol{v}^{(l)}}_i\right|^2}} \tag{73}$$

Because $\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right)$ is an infimum with more constraints than $\mathcal{R}'\left(\boldsymbol{\beta}|(1)_{l=0}^{L-1}\right)$, we have:

$$\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) \leq \mathcal{R}'\left(\boldsymbol{\beta}|(1)_{l=0}^{L-1}\right) \tag{74}$$

Moreover we have a closed form solution for $\mathcal{R}'\left(\boldsymbol{\beta}|(1)_{l=0}^{L-1}\right)$:

$$\mathcal{R}'\left(\boldsymbol{\beta}|(1)_{l=0}^{L-1}\right) = n^{-\frac{L}{2}} \sqrt{\inf_{\substack{\forall l\in[0,L-1]:\\ v^{(l)}\in\mathbb{R},\\ v^{(l)}=1}} \sum_{i=0}^{n-1} \frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\widehat{\boldsymbol{v}^{(l)}}_i\right|^2}} \tag{75}$$

$$= n^{-\frac{L}{2}} \sqrt{\sum_{i=0}^{n-1} \frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\frac{1}{\sqrt{n}}\right|^2}} \tag{76}$$

$$= \sqrt{\sum_{i=0}^{n-1}\left|\widehat{\boldsymbol{\beta}}_i\right|^2} = \|\widehat{\boldsymbol{\beta}}\|_2 = \|\boldsymbol{\beta}\|_2 \tag{77}$$

So we obtain the upper bound:

$$\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) \leq \|\boldsymbol{\beta}\|_2 \tag{78}$$

The last step is to prove our lower bound:

$$\sqrt{\frac{1}{\prod_{l=0}^{L-1} s_l}}\|\boldsymbol{\beta}\|_2 \leq \mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) \tag{79}$$

For that we will first prove the following upper bound on $|\widehat{\boldsymbol{v}^{(l)}}_j|$:

$$\forall j \in [0, n-1] : |\widehat{\boldsymbol{v}^{(l)}}_j| \leq \sqrt{\frac{s_l}{n}} \tag{80}$$

We do it as follow:

$$\left|\widehat{\boldsymbol{v}^{(l)}}_j\right| = \left|\frac{1}{\sqrt{n}}\sum_{k=0}^{s_l-1}\boldsymbol{v}_k^{(l)}(\omega_n)^{jk}\right| \tag{81}$$

$$\leq \frac{1}{\sqrt{n}}\sum_{k=0}^{s_l-1}\left|\boldsymbol{v}_k^{(l)}(\omega_n)^{jk}\right| = \frac{1}{\sqrt{n}}\sum_{k=0}^{s_l-1}\left|\boldsymbol{v}_k^{(l)}\right| = \frac{\|\boldsymbol{v}^{(l)}\|_1}{\sqrt{n}} \tag{82}$$

$$\leq \frac{\sqrt{s_l}\|\boldsymbol{v}^{(l)}\|_2}{\sqrt{n}} = \sqrt{\frac{s_l}{n}} \tag{83}$$

We can use this upper bound on $|\widehat{\boldsymbol{v}^{(l)}}_j|$ to produce an lower bound on $\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right)$:

$$\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right) = n^{-\frac{L}{2}}\sqrt{\inf_{\substack{\forall l\in[0,L-1]:\\ v^{(l)}\in\mathbb{R}^{s_l},\\ \|v^{(l)}\|=1}}\sum_{i=0}^{n-1}\frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\widehat{\boldsymbol{v}^{(l)}}_i\right|^2}} \tag{84}$$

$$\geq n^{-\frac{L}{2}}\sqrt{\sum_{i=0}^{n-1}\frac{\left|\widehat{\boldsymbol{\beta}}_i\right|^2}{\prod_{l=0}^{L-1}\left|\sqrt{\frac{s_l}{n}}\right|^2}} = \sqrt{\frac{1}{\prod_{l=0}^{L-1}s_l}}\|\boldsymbol{\beta}\|_2 \tag{85}$$

□

E.5. Proof Theorem 6

Using the fact that $\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}$ is the minimizer of $\mathcal{R}'\left(\boldsymbol{\beta}|(s_l)_{l=0}^{L-1}\right)$ under the data constraints, we obtain:

$$\left\|\bar{\boldsymbol{\beta}}^{(1)_{l=0}^{L-1}}\right\|_2 = \mathcal{R}'\left(\bar{\boldsymbol{\beta}}^{(1)_{l=0}^{L-1}}|(1)_{l=0}^{L-1}\right) \leq \mathcal{R}'\left(\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}|(1)_{l=0}^{L-1}\right) = \left\|\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right\|_2 \tag{86}$$

$$\sqrt{\frac{1}{\prod_{l=0}^{L-1}s_l}}\left\|\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right\|_2 \leq \mathcal{R}'\left(\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}|(s_l)_{l=0}^{L-1}\right) \leq \mathcal{R}'\left(\bar{\boldsymbol{\beta}}^{(1)_{l=0}^{L-1}}|(s_l)_{l=0}^{L-1}\right) \leq \left\|\bar{\boldsymbol{\beta}}^{(1)_{l=0}^{L-1}}\right\|_2 \tag{87}$$

By moving the terms in the equations above, we obtain:

$$\left\|\bar{\boldsymbol{\beta}}^{(1)_{l=0}^{L-1}}\right\|_2 \leq \left\|\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right\|_2 \leq \left(\sqrt{\prod_{l=0}^{L-1}s_l}\right)\left\|\bar{\boldsymbol{\beta}}^{(1)_{l=0}^{L-1}}\right\|_2 \tag{88}$$

By combining the fact that $\bar{\epsilon}\left((s_l)_{l=0}^{L-1}\right) = \frac{1}{\left\|\bar{\boldsymbol{\beta}}^{(s_l)_{l=0}^{L-1}}\right\|_2}$ (see Lemma 4) and equation 88, we obtain:

$$\frac{1}{\prod_{l=0}^{L-1}s_l}\bar{\epsilon}\left((1)_{l=0}^{L-1}\right) \leq \bar{\epsilon}\left((s_l)_{l=0}^{L-1}\right) \leq \bar{\epsilon}\left((1)_{l=0}^{L-1}\right) \tag{89}$$

□

Appendix F. Experimental setup

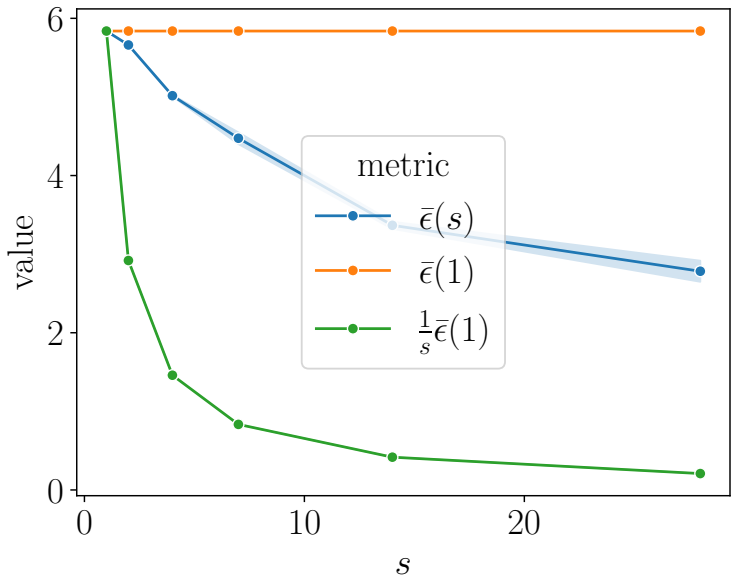
For all datasets and kernel configuration we trained the model with stochastic gradient descent with a batch size 64. Let $\boldsymbol{\beta}\left(\boldsymbol{\theta}_t \middle| (s_l)_{l=0}^{L-1}\right)$ be the linear predictor at epoch t, at each 25 epochs we compute the following directional convergence metric:

$$\gamma(t) = \cos\left(\boldsymbol{\beta}\left(\boldsymbol{\theta}_t \middle| (s_l)_{l=0}^{L-1}\right), \boldsymbol{\beta}\left(\boldsymbol{\theta}_{t-25} \middle| (s_l)_{l=0}^{L-1}\right)\right) \tag{90}$$

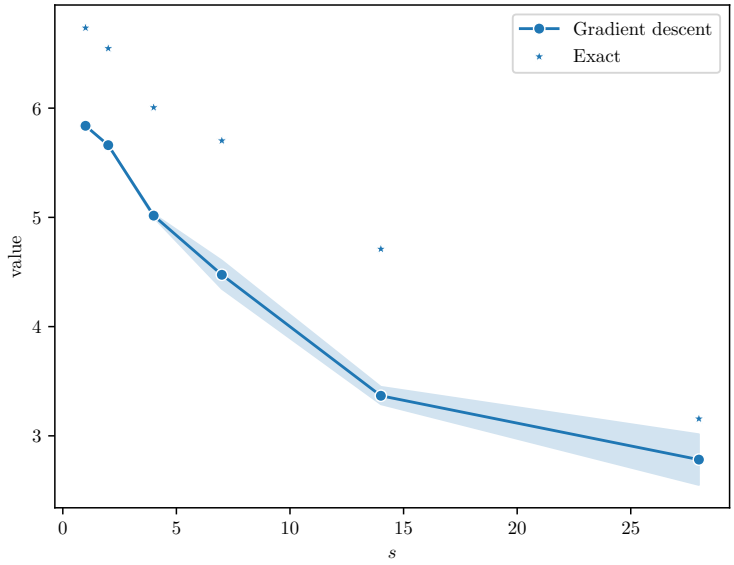
Where cos is the cosine similarity. The training stop when $\gamma(t) < 1.0 \times 10^{-8}$. For all experiments the models are initialized with Xavier normal initialization and a gain of 0.1. For the MNIST experiments the learning rate is 5.12×10^{-2}. In the case of the mini-ImageNet experiments, the learning rate is $\sqrt{\frac{64}{s}} \times 2 * 10^{-2}$ where s is the kernel size. Moreover, for the mini-ImageNet experiments, we also clipped the gradient after each epoch such that the ℓ_2 norm of the gradient stay smaller than 10.[6] The use of gradient clipping is also related to training stability. The mini-ImageNet images are convert from RGB to grayscale and resized to 256×256. To confirm the low variance in the robustness for a given configuration with our stopping criterion, we do 5 runs for each kernel sizes in the experiment with MNIST and linear CNN with 1 convolutional layer (left plot in Fig. 1), the standard deviation is represented by the blue area in the left plot in Fig. 1.

Computing numerically the exact solution for equation 5 is difficult for large datasets. In Fig. 2, we compare the exact solution obtained using constrained optimization and the solution obtained with gradient descent (with the same stopping criterion as described above) on subset of (binary) MNSIT with 5 samples per class. While the robustness is higher with exact solution, we observed the same decreasing trend in both scenario.

6. see https://pytorch.org/docs/stable/generated/torch.nn.utils.clip_grad_norm_.html



(a) gradient descent



(b) exact solution for $\bar{\epsilon}(s)$

Figure 2: Robustness obtained with gradient descent vs the exact solution for a subset of MNSIT with 5 samples per class