Extended Abstract Track

Adversarially-robust representation learning through spectral regularization of features

Editors: List of editors' names

Abstract

The vulnerability of neural network classifiers to adversarial attacks is a major obstacle to their deployment in safety-critical applications. Regularization of network parameters during training can be used to improve adversarial robustness and generalization performance. Usually, the network is regularized end-to-end, with parameters at all layers affected by regularization. However, in settings where learning representations is key, such as self-supervised learning (SSL), layers after the feature representation will be discarded when performing inference. For these models, regularizing up to the feature space is more suitable. To this end, we propose a new spectral regularizer for representation learning that encourages black-box adversarial robustness in downstream classification tasks.

Keywords: Representation learning, adversarial attacks

1. Introduction

Neural networks are vulnerable to adversarial attacks (Biggio et al., 2013; Szegedy et al., 2013). Even without access to the model parameters and only access to inputs and outputs, attackers can still fool the network (Liu et al., 2018; Li et al., 2022; Kurakin et al., 2018). Identifying effective training algorithms that guard against these black-box attacks has therefore garnered widespread attention (Athalye et al., 2018; Madry et al., 2018; Athalye et al., 2018; Gao et al., 2019; Xie et al., 2017; Liu et al., 2018; Cohen et al., 2019). However, recent years have seen growing popularity of representation learning paradigms to which many previous adversarial defenses are not immediately applicable. For instance, in transfer learning and self-supervised learning, the final linear readout layer of a network is retrained when performing downstream inference tasks. Only layers up to the feature representations are kept from the pretraining stage (Zbontar et al., 2021; Chen et al., 2020b; Radford et al., 2021; Hernandez et al., 2021). As standard adversarial training methods for classification networks typically require access to the last layer linear heads (Hein and Andriushchenko, 2017; Yoshida and Miyato, 2017), they cannot be directly applied in representation learning settings.

In this paper, we seek an adversarial training methodology that can be applied to these representation-focused training paradigms, in which the representation is pre-trained without knowledge of downstream tasks. In Section 2, we derive a theoretical guarantee on black-box adversarial robustness based on feature representations. This bound inspires our proposed regularizer Equation (3), which penalizes the top singular value of each hidden layer's weights (i.e. excluding the last layer's linear head). In Section 3, we empirically demonstrate the effectiveness of this regularizer. Our results provide evidence that having robust representations is crucial to adversarial robustness.

2. Spectral Regularization for Adversarial Robustness

2.1. Preliminaries

We first briefly introduce the notion of black-box adversarial robustness; a detailed overview of related works is given in Appendix A. Consider the problem of assorting n-dimensional data into K given classes. For a classification network $F(x;\Theta) : \mathbb{R}^n \times \mathbb{R}^{|\Theta|} \to \mathbb{R}^K$ with trainable parameters Θ, the class prediction is given by $\hat{y} = \arg\max_{k\in[K]} F_k(x;\Theta)$, which is correct if it agrees with the true class label y. In the following discussions we drop the Θ notation when there is no ambiguity. Here we assume the output logits are distinct.

We say $\delta_x \in \mathbb{R}^n$ is an **adversarial perturbation** to a correctly-classified sample x if it swaps the predicted class label, i.e., there is a class index $k \neq y$ such that $F_k(x+\delta_x) > F_y(x+\delta_x)$. Then, the **adversarial distance** Δ_x is defined as the minimal size of an adversarial perturbation for the datum x: $\Delta_x = \min_{\{\delta_x\in\mathbb{R}^n : \arg\max_{k\in[K]} F_k(x+\delta_x)\neq y\}} \|\delta_x\|_2$. In this paper we focus on l_2 norm but the analysis can be generalized to other norms as well. A network is **adversarially robust** to input perturbation with respect to sample x if Δ_x is large, and is globally adversarial robust if Δ_x is large for all correctly classified x.

2.2. A lower bound on adversarial distance

We now present our central lemma, which is an extension of the main result of Hein and Andriushchenko (2017). The objective of our analysis is to isolate the influence of the readout weights, as we focus on settings where the representation is learned in a pretraining phase and the readouts are trained separately to perform downstream tasks.

Lemma 1 *Suppose a neural network classifier $F : \mathbb{R}^n \to \mathbb{R}^K$ is continuously differentiable and can be decomposed into a feature map $\Phi(x) : \mathbb{R}^n \to \mathbb{R}^d$, where d is the feature dimension, and a linear decision head $f(z) : \mathbb{R}^d \to \mathbb{R}^K$: $F = \texttt{SoftMax} \circ f \circ \Phi$, with $f(z) = W^{(L)}z$ the last layer linear transformation without bias, and $W_c^{(L)}$ the c-th row of $W^{(L)}$, treated as a row vector. Suppose $x \in \mathbb{R}^n$ is a sample input belonging to class c and $\delta_x \in \mathbb{R}^n$ an adversarial perturbation to x that results in an incorrect prediction of class k. Fixing a perturbation radius 'budget' $R > 0$ such that $\|\delta_x\|_2 \leqslant R$, we have*

$$\Delta_x \geqslant \theta_x \|\Phi(x)\|_2 \cdot \frac{1}{\max_{y\in B_2(x,R)} \|\nabla\Phi(y)\|_2} \tag{1}$$

where $\theta_x = \frac{(W_c^{(L)} - W_k^{(L)})\Phi(x)}{\|W_c^{(L)} - W_k^{(L)}\|_2 \|\Phi(x)\|_2}$ is the cosine of the angle between the feature representation of x and the last layer "confidence weights" $W_c^{(L)} - W_k^{(L)}$ for class c relative to class k. Here, $\nabla\Phi(\cdot)$ is the Jacobian of the feature map with respect to the input.

Proof See Appendix B.1 for the proof of this lemma, extended to general l_p norms. ■

In this lower bound, the key term of interest from a feature representation perspective is the norm of the feature map's gradient $\|\nabla\Phi(\cdot)\|_2$ in the denominator. As it depends on the readout weights, the angle θ_x cannot be controlled during the pretraining phase when features are learned. Thus, the bound suggests that we should penalize $\max_{y\in B_2(x,R)} \|\nabla\Phi(y)\|_2$ during representation learning. This contrasts with previous works on supervised learning that

penalize the input-output Jacobian ∇F (Hein and Andriushchenko, 2017; Yoshida and Miyato, 2017; Hoffman et al., 2019). However, to make regularization computationally efficient, we must make several relaxations of this objective. First, we consider the norm of the gradient at x only, since searching over the ball is computationally expensive and not easily parallelizable. As the 2-norm of a matrix is its maximum singular value, we have

$$\|\nabla\Phi(x)\|_2^2 = \lambda_{\max}(\nabla\Phi(x)\nabla\Phi(x)^T) = \lambda_{\max}(\nabla\Phi(x)^T\nabla\Phi(x)) =: \lambda_{\max}(g), \tag{2}$$

where we have defined $g := \nabla\Phi(x)^T\nabla\Phi(x)$. Here, $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of a matrix. We recognize g as the pullback of the Euclidean metric on feature space back to input space, as studied by Zavatone-Veth et al. (2023). A robust classification network thus must have small $\lambda_{\max}(g)$.

In principle, $\lambda_{\max}(g)$ is differentiable so long as the spectral gap is nonzero, and so can be naïvely added as a regularizer. However, computing the resulting updates using automatic differentiation suffers from high runtime and memory consumption (Zavatone-Veth et al., 2023). To alleviate such costs, we perform a series of relaxations. Assuming that the activation function has derivatives (almost everywhere) bounded by 1, we can bound $\lambda_{\max}(g)$ by the product of the largest singular values squared of the hidden layer weights $W^{(l)}$, $\sigma^2_{\max}(W^{(l)})$, where $l \in [L-1]$ for L the the total number of layers. This trick was used before by Yoshida and Miyato (2017). The resulting regularizer, which we henceforth refer to as `rep-spectral` because it is a `spectral` regularizer on `rep`resentations, is

$$L^{(\texttt{rep-spectral})}(\Theta) = \tfrac{\gamma}{2}\textstyle\sum_{l=1}^{L-1}\sigma^2_{\max}(W^{(l)}),. \tag{3}$$

where $\gamma \geqslant 0$ is the regularization strength. We present the full derivation of this regularizer in Appendix B.2. We discuss further approximations that can be made to reduce computational cost in Appendix D, and describe how to extend this regularizer to convolutional networks in Appendix C. One could instead take logarithms to directly regularize the product of singular values, but this would incur additional computational costs and result in a data-dependent scaling factors in gradient computation that could be heuristically absorbed in γ. The difference with Yoshida and Miyato (2017) is that we drop the penalization of the last layer. We refer to their proposed regularizer as `ll-spectral` to emphasize that it penalizes the top singular value of the readout weights, i.e., it adds the $l = L$ term to (3).

3. Experiments

We now evaluate our proposed `rep-spectral` regularizer based on the accuracy of classification and the adversarial distance of selected samples at the end of training. We focus on transfer learning in the main text, and provide a few supervised examples in Appendix F. To evaluate black-box adversarial robustness, we use the Tangent Attack (TA) method (Ma et al., 2021). We provide a more detailed discussion of the TA method in Appendix E, and document training and evaluation details in Appendix F.

As an example transfer learning task, we pretrain ResNet50 on ImageNet-1K (Deng et al., 2009) and finetune on CIFAR-10 (see Appendix F.3 for details). In the finetuning stage, the readout layer is trained from scratch, while only miniscule adjustments are made to the hidden layers. For comparison in this setting, we consider finetuning using the

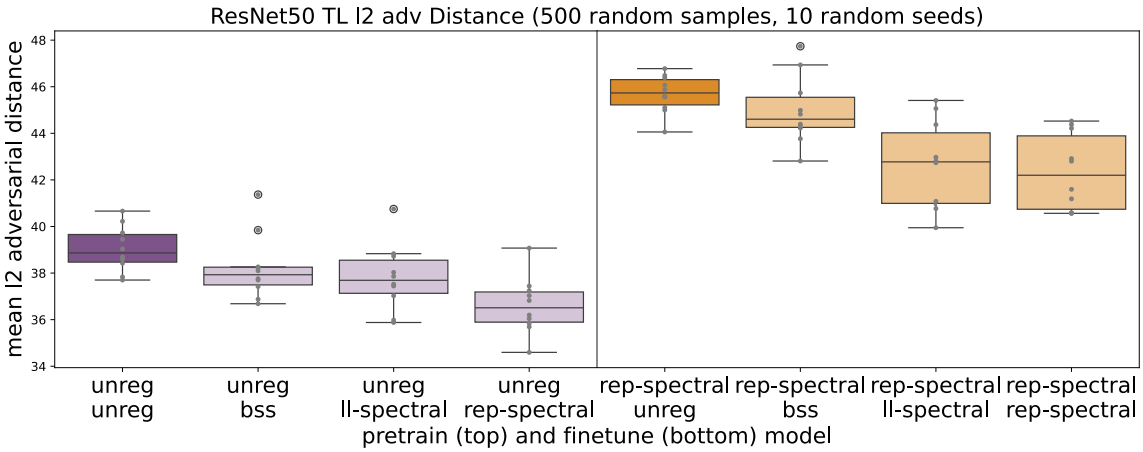


Figure 1: Mean Δ_x in transfer learning across different combinations of training schemes. The left half are finetuning from unregularized model, and the right half are finetuning from `rep-spectral` regularized model. In finetuning stage, adding additional regularization typically hurts adversarial robustness performance.

`ll-spectral` regularizer of Yoshida and Miyato (2017) or using batch spectral shrinkage (`BSS`) (Chen et al., 2019). We compute the mean adversarial distances for 500 randomly test samples and report the mean over 10 random seeds in Figure 1 and corresponding test accuracy in Figure 7. Although all model reaches 96% test accuracy consistently, they have dramatically different robustness level. We found adding `rep-spectral` regularization at the pretraining stage produces substantial gains in adversarial robustness, while adding adversarial regularization in the finetuning stage typically hurts adversarial robustness. The best robustness is obtained by adding our proposed regularizer during pretraining and then fine-tuning without regularization (compare the dark purple bar with the dark orange bar in Figure 1). A similar pattern holds in finetuning on other datasets; see Appendix F.3 and Figure 8 for more details. Therefore, our proposed method yields a substantial gain in robustness in a transfer learning setting.

4. Discussion

In this paper, we have shown that a simple regularization method encourages adversarial robustness during representation learning. This study opens the door for analyzing adversarial robustness on a per-layer basis. In deep networks, it is widely believed that early and later layers assume different roles in the learning tasks. Early layers may be responsible for low-level feature extraction, while later layers adapt to high-level and task-specific features (Cammarata et al., 2020; Feather et al., 2023; Zavatone-Veth et al., 2023). As analyzed in work by Dyballa et al. (2024) on the generalization performance of neural networks from a per-layer perspective, an interesting extension of our work would be to conduct ablation studies by turning on and off spectral regularizations for certain layers, not necessarily contiguous ones. This could allow one to identify the crucial layers contributing to model adversarial robustness.

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

Dimitris Bertsimas and Jack Dunn. *Machine learning under a modern optimization lens*. Dynamic Ideas LLC Charlestown, MA, 2019.

Dimitris Bertsimas and Dick den Hertog. Robust and adaptive optimization. *(No Title)*, 2022.

Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Lower bounds on adversarial robustness from optimal transport. *Advances in Neural Information Processing Systems*, 32, 2019.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, pages 387–402. Springer, 2013.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. Thread: Circuits. *Distill*, 2020. doi: 10.23915/distill.00024. https://distill.pub/2020/circuits.

Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daum III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020b. URL https://proceedings.mlr.press/v119/chen20j.html.

Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Elvis Dohmatob. Classifier-independent lower-bounds for adversarial robustness. *arXiv preprint arXiv:2006.09989*, 2020.

Luciano Dyballa, Evan Gerritz, and Steven W Zucker. A separability-based approach to quantifying generalization: which layer is best? *arXiv preprint arXiv:2405.01524*, 2024.

Jenelle Feather, Guillaume Leclerc, Aleksander Madry, and Josh H. McDermott. Model metamers reveal divergent invariances between biological and artificial neural networks. *Nature Neuroscience*, 26(11):2017–2034, Nov 2023. ISSN 1546-1726. doi: 10.1038/s41593-023-01442-0. URL https://doi.org/10.1038/s41593-023-01442-0.

Ruiqi Gao, Tianle Cai, Haochuan Li, Cho-Jui Hsieh, Liwei Wang, and Jason D Lee. Convergence of adversarial training in overparametrized neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Charles R. Harris, K. Jarrod Millman, Stfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernndez del Ro, Mark Wiebe, Pearu Peterson, Pierre Grard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357362, 2020. doi: 10.1038/s41586-020-2649-2.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in neural information processing systems*, 30, 2017.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv*, 2021.

Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. *arXiv*, 2019.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/cifar.html.

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]*, 2, 2010. URL http://yann.lecun.com/exdb/mnist.

Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. Qeba: Query-efficient boundary-based blackbox attack. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1221–1230, 2020.

Yao Li, Minhao Cheng, Cho-Jui Hsieh, and Thomas CM Lee. A review of adversarial attack and defense for classification methods. *The American Statistician*, 76(4):329–345, 2022.

Shengchao Liu, Dimitris Papailiopoulos, and Dimitris Achlioptas. Bad global minima exist and sgd can reach them. *Advances in Neural Information Processing Systems*, 33: 8543–8552, 2020.

Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. In *International Conference on Learning Representations*, 2018.

Yujia Liu, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. A geometry-inspired decision-based attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4890–4898, 2019.

Chen Ma, Xiangyu Guo, Li Chen, Jun-Hai Yong, and Yisen Wang. Finding optimal tangent points for reducing distortions of hard-label attacks. *Advances in Neural Information Processing Systems*, 34:19288–19300, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. 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/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE conference on computer vision and pattern recognition*, pages 413–420. IEEE, 2009.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. 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 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Md Farhamdur Reza, Ali Rahmati, Tianfu Wu, and Huaiyu Dai. Cgba: Curvature-aware geometric black-box attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 124–133, 2023.

Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*, 2018.

Alexandra Senderovich, Ekaterina Bulatova, Anton Obukhov, and Maxim Rakhuba. Towards practical control of singular values of convolutional layers. *Advances in Neural Information Processing Systems*, 35:10918–10930, 2022.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

Ziang Yan, Yiwen Guo, Jian Liang, and Changshui Zhang. Policy-driven attack: Learning to query for hard-label black-box adversarial examples. In *International Conference on Learning Representations*, 2020.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Jacob A Zavatone-Veth, Sheng Yang, Julian A Rubinfien, and Cengiz Pehlevan. Neural networks learn to magnify areas near decision boundaries. *arXiv preprint arXiv:2301.11375*, 2023.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.

Appendix A. Related Works

This section provides a brief overview of the adversarial robustness literature, with a focus on black-box defenses. Two major approaches to improve black-box adversarial robustness have been proposed: training with adversarial examples and regularization (Li et al., 2022; Bai et al., 2021). To train with adversarial samples, in each update to the network parameters, the training set is augmented with adversarial examples. Since we have access to input gradients during model training, one can use white-box attacks to find these examples (Goodfellow et al., 2014; Kurakin et al., 2018; Madry et al., 2018; Athalye et al., 2018; Gao et al., 2019; Xie et al., 2017; Liu et al., 2018; Cohen et al., 2019). By forcing the model to become robust to these perturbations during training, it becomes less susceptible to future adversarial attacks.

However, training with adversarial examples is computationally expensive, and it does not guarantee that the final classifier will be adversarially robust because of its strong dependence on the training dataset (Hein and Andriushchenko, 2017). A less data-dependent and more computation friendly method is to design regularizers that encourage robustness. By adding specially designed regularization terms, the model can escape bad, non-robust local minima during optimization (Liu et al., 2020). For linear regression, logistic regression, and decision trees with known uncertainty set structure, an exact equivalence between robustness and regularization has been established (Bertsimas and Dunn, 2019; Bertsimas and Hertog, 2022). In more advanced applications, one can derive regularizers that promote raising lower bounds on the adversarial distance (Hein and Andriushchenko, 2017; Bhagoji et al., 2019; Dohmatob, 2020). Our analysis attempts to generalize this adversarial robustness notion further for newer classification architectures.

In either case, searching for an adversarial sample with minimal adversarial distance in a black-box fashion is highly nontrivial. This makes black-box robustness evaluation rather difficult in practice, meaning that it is hard to evaluate defenses conclusively. Many query-based heuristic methods have been proposed (Brendel et al., 2017; Chen et al., 2020a; Li et al., 2020; Liu et al., 2019; Yan et al., 2020; Ma et al., 2021; Reza et al., 2023), which rely on iterative search for the closest point on the decision boundary of a trained model to a given sample. In our analysis we employ one such method, the tangent attack (TA) (Ma et al., 2021), to evaluate model adversarial robustness.

Appendix B. Proofs and Discussions

B.1. Proof of Theorem 1

We first restate the lemma using dual norm formulation.

Restatment of Theorem 1 *Suppose a neural network classifier $F : \mathbb{R}^n \to \mathbb{R}^K$ is continuously differentiable and can be decomposed as*

$$F = \texttt{SoftMax} \circ f \circ \Phi. \tag{4}$$

Suppose $x \in \mathbb{R}^n$ is a sample input belonging to class c and $\delta \in \mathbb{R}^n$ an adversarial perturbation to x with error class predicted as k. Further, given $\frac{1}{p} + \frac{1}{q} = 1$, assume $\|\delta\|_p \leqslant R$ for some

radius $R > 0$*, we have*

$$\|\delta_x\|_p \geqslant \frac{\left(W_c^{(L)} - W_k^{(L)}\right)\Phi(x) + b_c^{(L)} - b_k^{(L)}}{\|W_c^{(L)} - W_k^{(L)}\|_q} \cdot \frac{1}{\max\limits_{y \in B_p(x,R)} \|\nabla\Phi(y)\|_q} \tag{5}$$

where $f(z) = W^{(L)}z + b^{(L)}$ *is the last layer linear transformation,* $W_c^{(L)}$ *is the c-th row of* $W^{(L)}$ *treated as a row vector, and* $\nabla\Phi(.)$ *is the Jacobian of the feature map w.r.t. input.*

Proof This lemma directly extends the main theorem of Hein and Andriushchenko (2017), where a general classifier was considered instead.

Define $h = f \circ \Phi$. First, $F(x)$ and $h(x)$ have the same ordering of its coordinates, since `SoftMax` is a strictly monotonic transformation. More explicitly,

$$F_c(x+\delta) \leqslant F_k(x+\delta) \iff h_c(x+\delta) \leqslant h_k(x+\delta) \tag{6}$$

second, by Taylor expansion to the first order in integral form, we have

$$h_k(x+\delta) = h_k(x) + \int_0^1 \langle \nabla h_k(x+t\delta), \delta\rangle \, dt, \quad \forall k \in [K] \tag{7}$$

where $\nabla h_k(.)$ is the gradient taken w.r.t. to the input, not the parameter of the neural network. Applying Equation (7) to Equation (6) on both sides, we have

$$\begin{aligned}
h_c(x) - h_k(x) &\leqslant \int_0^1 \langle \nabla h_k(x+t\delta) - \nabla h_c(x+t\delta), \delta\rangle \, dt && (8)\\
&\leqslant \|\delta\|_p \int_0^1 \|\nabla h_k(x+t\delta) - \nabla h_c(x+t\delta)\|_q \, dt \quad (\text{Hlder with } \frac{1}{p} + \frac{1}{q} = 1) && (9)\\
&\leqslant \|\delta\|_p \cdot \max_{y \in B_p(x,R)} \|\nabla h_k(y) - \nabla h_c(y)\|_q \qquad (R \text{ the norm of } \delta) && (10)\\
&\leqslant \|\delta\|_p \cdot \max_{y \in B_p(x,R)} \|\nabla\Phi(y)\|_q \|\nabla f_k(\Phi(y)) - \nabla f_c(\Phi(y))\|_q \quad (\text{Chain Rule}) && (11)
\end{aligned}$$

here we use $\|.\|_q$ to denote the vector-induced matrix norm when the input is a matrix. Using Equation (11), rearranging terms, we get

$$\|\delta\|_p \geqslant \frac{h_c(x) - h_k(x)}{\max\limits_{y \in B_p(x,R)} \|\nabla\Phi(y)\|_q \|\nabla f_k(...) - \nabla f_c(...)\|_q} \tag{12}$$

note that we explicitly drop the dependence of ∇f on its input since f is assumed linear. Parametrizing $f(.)$ by $W^{(L)}, b^{(L)}$ and use the fact that $h = f \circ \Phi$, we get Equation (5). Lastly, taking $p = q = 2$ and assuming centered data, we get Equation (1). ■

B.2. Derivation of Feature Regularization

In this section we derive the regularization presented in Equation (3). Consider an explicit parameterization of $F(.;\Theta)$ as a neural network with only linear layers parametrized by $\Theta = \{W^{(1)}, b^{(1)}, ..., W^{(L)}, b^{(L)}\}$ and non-linear activation function $\phi(.)$ and denote z^l as

the l-th layer preactivation value and $D^{(l)}$ a diagonal matrix with diagonals given by $\texttt{diag}(D^{(l)}) = \phi'(z^{(l)})$, then

$$\nabla\Phi(x) = D^{(L-1)}W^{(L-1)}D^{(L-2)}W^{(L-2)}\dots D^{(1)}W^{(1)} \tag{13}$$

therefore

$$\lambda_{\max}(g) = \lambda_{\max}(D^{(L-1)}W^{(L-1)}\dots D^{(1)}W^{(1)}(W^{(1)})^T(D^{(1)})^T\dots(W^{(L-1)})^T(D^{(L-1)})^T) \tag{14}$$

$$\leqslant \prod_{l=1}^{L-1} \lambda_{\max}(W^{(l)}(W^{(l)})^T) \cdot \lambda_{\max}(D^{(l)}(D^{(l)})^T) \tag{15}$$

where the last line uses the cyclic property in computing the eigenvalues recursively. Note that for common activation function the derivatives are upper-bounded by 1, and so the maximum eigenvalue $\lambda_{\max}(D^{(l)}(D^{(l)})^T)$ is upper bounded by 1. We have

$$\lambda_{\max}(g) \leqslant \prod_{l=1}^{L-1} \lambda_{\max}(W^{(l)}(W^{(l)})^T) = \prod_{l=1}^{L-1} \sigma^2_{\max}(W^{(l)}) \tag{16}$$

where $\sigma_{\max}$ is the largest singular value. This motivates Equation (3) by replacing the product by a sum for easier back propagation.

Appendix C. Linearization of a Multi-channel Convolution Layer

This section provides a brief description of the linearization of multi-channel convolution layer and subsequently getting the maximum eigenvalue of this linear operation. More details and proofs can be found in (Sedghi et al., 2018; Senderovich et al., 2022).

C.1. Construction of the linear map $\tilde{\mathcal{K}}$

A periodic 2D convolution operation can be considered a linear transformation on the vectorized input, and the weights are constructed from the filters. Consider $X \in \mathbb{R}^{\mathrm{c_{in}} \times n \times n}$ an input image to a convolution layer with $\mathrm{c_{in}}$ numbers of input channels and height/width given by n. A multichannel filter $\mathcal{K} \in \mathbb{R}^{\mathrm{c_{out}} \times \mathrm{c_{in}} \times k \times k}$ with stride s with $\mathrm{c_{out}}$ number of output channels and kernel size k can be rewritten into a matrix $\tilde{\mathcal{K}} \in \mathbb{R}^{\mathrm{c_{out}} n^2_{\mathrm{out}} \times \mathrm{c_{in}} n^2}$ such that

$$\mathrm{Vec}(\texttt{Conv2D}(X)) = \tilde{\mathcal{K}}\ \mathrm{Vec}(X) \tag{17}$$

where Vec(.) is a row-major reshaping of X (i.e. the default behavior of calling `.flatten()` in `NumPy` and `PyTorch`), and n_{out} the output height/width given by $n_{\mathrm{out}} = \lfloor \frac{n-1}{s} + 1 \rfloor$.

The transformation $\tilde{\mathcal{K}}$ consists of $n^2 \times n^2$ blocks of doubly circulant matrix, and each of the doubly circulant matrix contains data that come from appropriately slicing the zero-padded filter $\mathcal{K}$ that matches with the same shape of the 2D input image. It could be validated that the singular values of $\tilde{\mathcal{K}}$ is the union of all singular values of 2D FFT-transformed blocks of the appropriate slicing, so that to compute the top singular value of $\tilde{\mathcal{K}}$, one does not need to construct $\tilde{\mathcal{K}}$ itself but instead should record the top singular values of FFT-transformed slices and take the max of all these maximum singular values, which saves substantial computational resources.

In Code Block 1 we present `PyTorch` code that is modified from theorem 2 of Senderovich et al. (2022) for computing the square of top singular value of $\tilde{\mathcal{K}}$.

```python
import torch
from torch.nn.functional import pad

# function body
def get_multi_channel_top_eigval_with_stride(
    kernel: torch.Tensor, h: int, w: int, stride: int
) -> torch.Tensor:
    """
    compute top eigen value of a convolution layer
    * code tested only for even n and stride = 1 or 2.

    :param kernel: the conv2d kernel, with shape (c_out, c_in, k, k)
    :param h: the image height
    :param w: the image width
    :param stride: the stride of convolution
    :return the top singular value for conv layer
    """
    # pad zeros to the kernel to make the same shape as input
    c_out, c_in, k_h, k_w = kernel.shape
    pad_height = h - k_h
    pad_width = w - k_w
    kernel_pad = pad(kernel, (0, pad_height, 0, pad_width), mode="constant",
                                                value=0)
    str_shape_height, str_shape_width = h // stride, w // stride

    # downsample the kernel
    transforms = torch.zeros(
        (c_out, c_in, stride**2, str_shape_height, str_shape_width)
    ).to(kernel.device)
    for i in range(stride):
        for j in range(stride):
            transforms[:, :, i * stride + j, :, :] = kernel_pad[
                :, :, i::stride, j::stride
            ]

    # batch fft2
    transforms = torch.fft.fft2(transforms)
    transforms = transforms.reshape(c_out, -1, str_shape_height,
                                            str_shape_width)
    P = transforms.permute(2, 3, 0, 1)

    # compute singular value squared
    eigvals = torch.linalg.eigvalsh(
        torch.einsum("...ij,...kj->...ik", torch.conj(P), P)
        if P.shape[3] > P.shape[2]
        else torch.einsum("...ji,...jk->...ik", torch.conj(P), P)
    )

    # keep top eigenvalue only
    top_eig = eigvals.max()
    return top_eig
```

Code Block 1: Python code for computing the top singular value of the operator form of a convolutional layer

Algorithm 1 Power Iteration for Top Singular Value Squared

Require: $M \in \mathbb{C}^{m \times n}, N \in \mathbb{N}$ ▷ N number of iterations
Ensure: $\lambda = \sigma_1(M)^2$
  $v = v_0$ ▷ Randomly Generated or from previous training iterates
  $v \leftarrow \frac{v}{\|v\|_2}$ ▷ Normalize
  $i \leftarrow 0$
  while $i < N$ **do**
    $u \leftarrow Mv$
    $u \leftarrow \frac{u}{\|u\|_2}$
    $v \leftarrow M^* u$ ▷ Conjugate Transpose
    $v \leftarrow \frac{v}{\|v\|_2}$
    $i \leftarrow i + 1$
  end while
  $p = Mv$ ▷ $Mv_1 = \sigma_1 u_1$
  $\lambda = \|p\|_2^2$

C.2. Speeding Up Top Eigenvalue Computation: Power Iteration Across Parameter Updates

Since here we are only interested in the maximal eigenvalue, we could use batched power iteration (algorithm described in Appendix D) to jointly compute the top eigenvalues only for 2D FFT-transformed blocks. Both 2D FFT transformation and batch eigenvalue update are GPU friendly. Furthermore, since between consecutive parameter updates, the change in filter map is likely small, iterations in the power method can be amortized across parameter updates. We empirically find that the outcome of conducting a batch power iteration update to the top eigenvalues every 20 parameter updates have little difference in performance compared to computing the exact eigenvalue before each parameter update.

The amortization across parameter update or across epochs trick has been noticed previously in Yoshida and Miyato (2017), but was only applied to fully-connected layers.

C.3. Implication of the Dependence on Input Dimension

Although a 2D convolution operator applies to images with arbitrary size and channels, the top singular value of the linearized map depends on the size and channels. To impose spectral regularization to convolution layers, it is thus recommended to use the size and channels of the test images for regularization. The impact of regularizing on one set of images and testing on another with different shapes and channels remains to be explored.

Appendix D. Power Iteration to Compute the Top Singular Value of a Linear Map

Power iteration is an iterative algorithm to compute the eigenspectrum of a diagonalizable matrix. We can extend the algorithm to compute the largest singular value of a complex-valued matrix. The algorithm is described in Algorithm 1. In practice, $N = 1$ is enough, since the parameter moves slowly as we train the cross entropy classification loss.

Appendix E. Finding Adversarial Distances by Tangent Attack Ma et al. (2021)

The Tangent Attack algorithm proposed by Ma et al. Ma et al. (2021) provides a good heuristic for finding adversarial distances in the black-box setting. Here, we briefly sketch how this algoritm operates; we refer the reader to Ma et al. (2021) for details.

Given a correctly classified sample x, we initialize the algorithm by adding a fixed number of Gaussian perturbations with predetermined standard deviations that adapts to the input dimension. Among all perturbed samples, we keep ones that were classified differently by the neural network classifier and select the one with the minimum l_2 distance to the original sample x. We then run a binary search along the line segment from x to that sample to locate a point that is on the decision boundary. We call this point x_0 as our initial guess to the adversarial sample to x.

Next we iteratively shrink the distance between $x_t, t \in \{0, 1, ..., T\}$ and x, where T is a predefined maximum number of updates to the adversarial guesses so that at the end of the algorithm x_T is considered the adversarial sample to x and the adversarial distance $\delta_x = \|x_T - x\|_2$. The update is done by performing the following three key steps in sequence: at each $t \in \{0, 1, ..., T-1\}$

1. **estimate a normal direction** to the decision boundary, pointing to the adversarial region: we take local perturbation to x_t. Based on the prediction on these perturbations, averaging the vectors that give adversarial prediction gives us an estimate to the normal direction;

2. **find the tangent point** in the 2D plane generated by x, x_t, and the normal direction. construct a hemisphere in the direction of the normal vector with a predefined small radius, find the tangent plane to the hemisphere that passes through x and locate the tangent point k. This step is done by analytic geometry and a closed form update can be analytically derived.

3. **conduct a binary search** along the line segment from x to k, get a sample on the decision boundary and assign that to x_{t+1}. In this way, x_{t+1} is a valid adversarial sample with a different prediction from x but is closer to x than x_t.

In our experiments, we use $T = 40$ throughout. Other hyperparameters such as the radius of hemisphere and the number of local perturbations for normal direction estimation follows identically from the hemisphere implementation in Ma et al. (2021).

Appendix F. Experiment Details

Experiments reported in the main text required less than 240 GPU-hours.

Our code base is adapted from various publicly available ones, including `TangentAttack` [1] with an Apache V2 license for evaluating the adversarial distances, `FixRes` (Touvron et al., 2019) [2] with a CC BY-NC 4.0 license for multi-GPU training Resnet50, `SimCLR` [3]

1. https://github.com/machanic/TangentAttack
2. https://github.com/facebookresearch/FixRes
3. https://github.com/sthalles/SimCLR

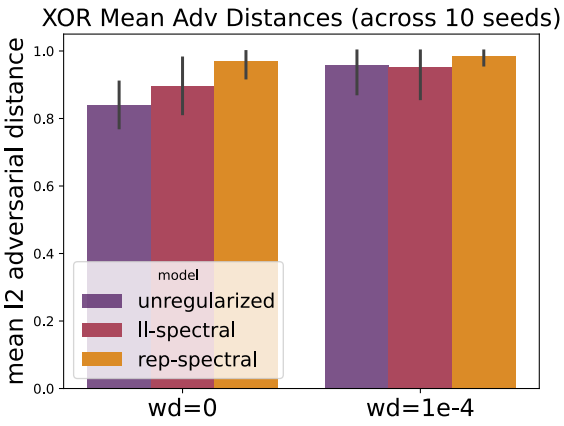


Figure 2: Average Δ_x found by TA across 10 different seeds for models trained on XOR with (*right*) and without (*left*) weight decay, and with the inclusion of our proposed `rep-spectral` regularizer, the `ll-spectral` regularizer that includes all layers, or no additional spectral regularizer. Error bars show ± 1 standard deviation over seeds.

with an MIT license for SimCLR model data loading and evaluation, `BarlowTwins` [4] with an MIT license for BarlowTwins data loading and evaluation, `practical_svd_conv` [5] with a BSD-3-Clause license for efficient computations of top singular value of 2D convolution layers, and lastly `nn_curvature` [6] with an MIT license for volume element computations.

Our `Python` code also uses some common publicly available packages, including `NumPy` (Harris et al., 2020) with a BSD license, `Matplotlib` (Hunter, 2007) with a BSD license, `Pandas` (McKinney et al., 2010) under a BSD license, `scikit-learn` (Pedregosa et al., 2011) with a BSD license, `PyTorch` (Paszke et al., 2019) under a modified BSD license, `tqdm` with an MIT license, and `toml` with an MIT license.

Data used in the project include MNIST LeCun et al. (2010), CIFAR-10 Krizhevsky (2009), ImageNet-1K Deng et al. (2009), Stanford Dog (Khosla et al., 2011), Oxford Flowers (Nilsback and Zisserman, 2008), and MIT indoor (Quattoni and Torralba, 2009).

F.1. Shallow MLPs trained on a toy dataset

To gain intuition for how our regularizer shapes representations, we first apply it to a single-hidden-layer MLP trained on a toy 2D XOR task. Though this task is of course unrealistic, it is potentially useful because we can directly visualize the input space. Given 4 data points at $[\pm 1, \pm 1] \in \mathbb{R}^2$, we use a network with 8 hidden units and GELU nonlinearity. To train shallow network for clean and noisy XOR with 8 hidden units, we train for 15000 epochs using full batch GD with 1.0 learning rate, 0.9 momentum, zero or 1e-4 weight decay, the same regularization strength ($\gamma = 0.0001$), and a regularization burnin period of 10500 epochs (70% unregularized training + 30% regularized training), all models could classify all 4 points correctly but demonstrate vastly different decision landscapes.

4. https://github.com/facebookresearch/barlowtwins/tree/main
5. https://github.com/WhiteTeaDragon/practical_svd_conv
6. https://github.com/Pehlevan-Group/nn_curvature

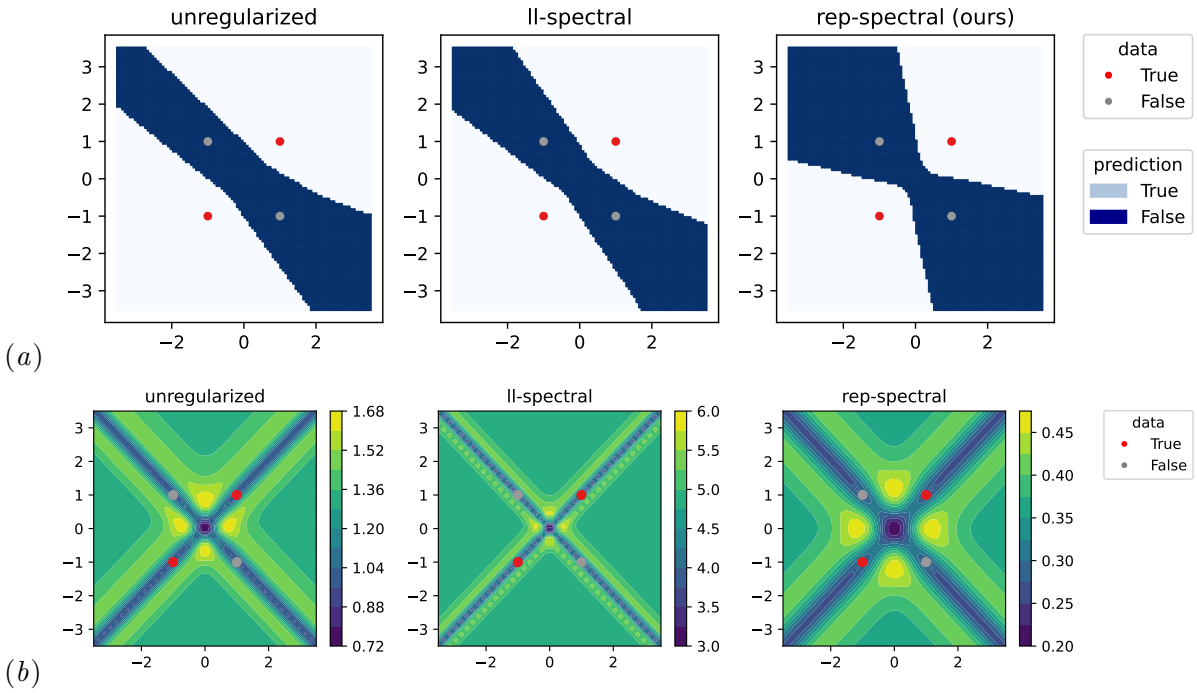


Figure 3: Effect of spectral regularization on the representations of MLPs trained on the toy XOR task. (a). Direct visualization of the decision boundaries of models trained using no regularization (*left*), `ll-spectral` regularization (*middle*), and our proposed `rep-spectral` regularization (*right*). The four training points are shown, colored according to their class. (b). Visualization of the volume element, which measures the sensitivity of the representation to small variations in the input, for models trained with each of these three methods.

Though our regularizer is motivated by the independent-pretraining, we find that even in this fully-supervised setting we obtain improved adversarial robustness, even compared to training using the `ll-spectral` regularizer that penalizes the last layer (Figure 2).

How does this robustness arise? As the input space for the XOR task is two-dimensional, we can directly visualize it. Examining the decision boundaries in Figure 3, we see that our `rep-spectral` regularizer results in increased classification margin, while the `ll-spectral` regularizer does not substantially affect the decision boundary. This difference reflects the fact that `ll-spectral` mostly just penalizes the readout layer norm and fails to control the feature layer norm (Figure 4). To gain a more detailed understanding of how the representations differ, we visualize the volume element corresponding to the metric induced by the feature map, i.e., $\sqrt{\det g}$ (Zavatone-Veth et al., 2023). In Figure 3, we see that the `rep-spectral` regularizer noticeably increases the areas of small volume element (and thus low representational sensitivity) near the class centers relative to the unregularized and `ll-spectral` models.

For shallow network, there are only two connection layers. The difference between Yoshida and Miyato (2017)'s `ll-spectral` regularization and our `rep-spectral` regularization is

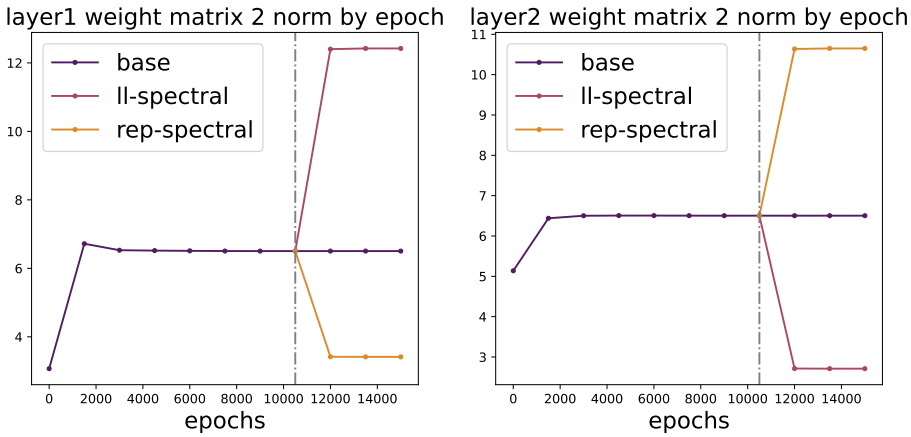


Figure 4: `ll-spectral` regularization and `rep-spectral` regularization weight norm change over training on the clean XOR data shown in Figure 3. At epoch 10500 we turn on the adversarial regularization, before which there is only crossentropy loss.

heuristically most pronounced in this case, since here `rep-spectral` is regularizing only half of the layers that `ll-spectral` is regularizing.

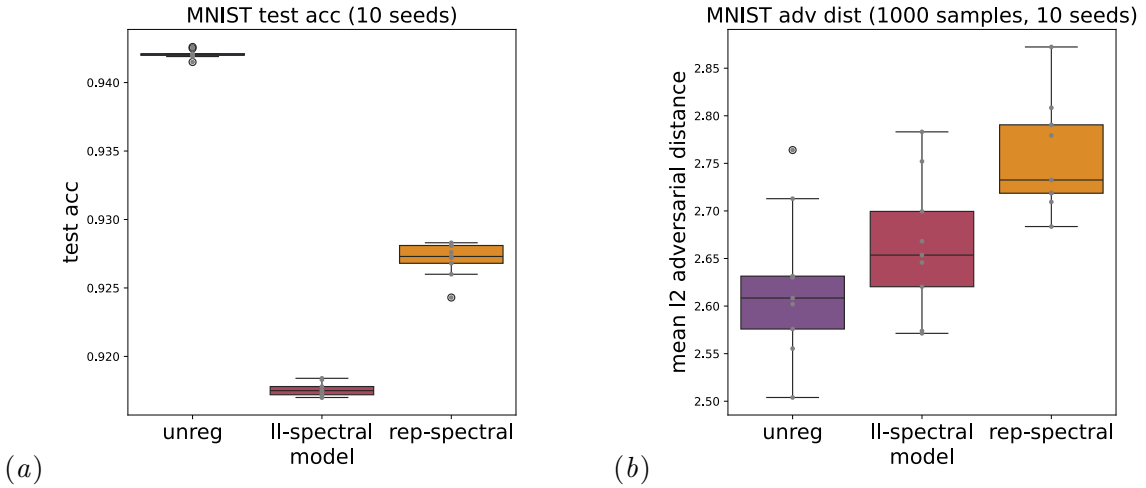
In clean XOR training, by plotting the matrix 2-norm of the connection weights in each layer through respective regularization, we found a striking contrast between the two. By imposing the same regularization strength for `rep-spectral` and `ll-spectral`, `rep-spectral` is able to control effectively the weight norm of the feature layer with an expansion of weight norm in the last layer; `ll-spectral` regularization is the exact opposite that it fails to control the weight norm in the feature layer. This is shown in the Figure 4.

F.2. Shallow MLPs trained on MNIST images

We next apply our regularizer to single-hidden-layer MLPs trained to classify MNIST images LeCun et al. (2010), with flattened input of 784 dimensions and 2000 hidden units. To train shallow network with 784 hidden units on MNIST dataset, we train for 200 epochs using SGD with batch size 1024, learning rate 0.1, momentum 0.9, weight decay 1e-4, $\gamma = 0.001$, and a regularization burn-in period of 160 epochs (80% unregularized training + 20% regularized training). The update of the eigendirections are done through one power iteration every parameter update. We sample 1000 testing images and apply the TA algorithm to detect the minimum l_2 perturbation.

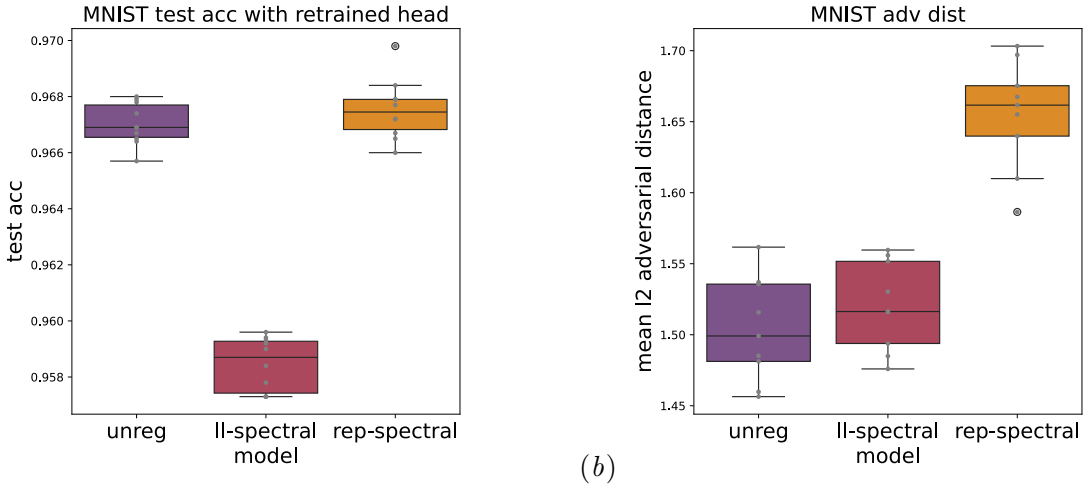
In the fully supervised setting, though regularization slightly decreases test accuracy, we observe that excluding the last layer produces a smaller loss of accuracy and a larger increase in adversarial distance compared to regularizing all layers (Figure 5).

To test whether our regularizer leads to more robust representations, we discard the linear head and perform classification using multilogistic regression on the fixed feature representations. We use multilogisitic regression with a l_2 regularization parameter 1 (i.e. the default setting when applying `sklearn.linear_model.LogisticRegression`) trained on the feature representations of the same train samples. Surprisingly, in this setting our regularizer does not hurt test accuracy compared to the unregularized model (Figure 6).



(a) (b)

Figure 5: Spectral regularization during supervised training of a single-hidden-layer MLP on MNIST images improves robustness. For each regularization method, we show text accuracy (*left*) and adversarial distance averaged across 1000 samples (*right*) across 10 random seeds. Gray dots indicate the results for individual seeds, while boxplots show the mean and quartiles.



(a) (b)

Figure 6: Re-training a readout from the hidden representation of an MLP pretrained on MNIST. For each regularization method, we show text accuracy (*left*) and adversarial distance averaged across 1000 samples (*right*) across 10 random seeds. Gray dots indicate the results for individual seeds, while boxplots show the mean and quartiles.

Moreover, the resulting model is more adversarially robust than when supervised pretraining of the representation is performed without regularization or with `ll-spectral` regularization (Figure 6). This suggests that, on a simple image classification task, our method achieves its stated goal: to enable representation learning that gives good adversarial robustness when an unregularized readout is trained to perform downstream tasks.

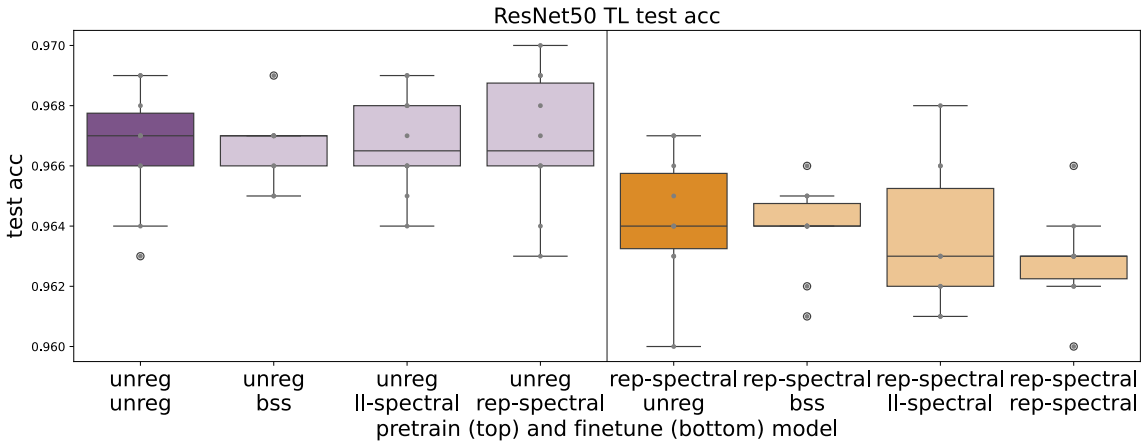


Figure 7: Test accuracy in transfer learning across multiple combinations of training schemes. The left half are finetuning from unregularized model, and the right half are finetuning from `rep-spectral` regularized model. All models reach an accuracy level of 96%, but have different adversarial robustness level shown in Figure 1.

F.3. Transfer Learning

In unregularized pretraining, we train ResNet50 for 120 epochs using SGD with learning rate 0.02, momentum 0.9, weight decay 1e-4, linear scheduling 30 epochs with decay 0.1. We distribute batchsize of 512 images across 4 GPUs for training. For regularized pretraining, starting at epoch 80 (67% unregularized training + 33% regularized training), we turn on `rep-spectral` regularization with $\gamma = 0.001$, with power update of top eigenvectors every 160 parameter updates to alleviate computation costs.

We test these two models' performance at finetune time, in which we train on CIFAR10 scale to 224 by 224 images with the same dimensionality with ImagNet for 50 epochs using SGD with 0.01 learning rate on linear head and 0.002 learning rate for the backbone, each with a ConsineAnneling scheduling with max parameter 200 epochs for both backbone and linear head. We visualize the test accuracy for each model of each random seed in Figure 7. On average, finetuned model starting with regularized weights have 0.2% drop in test accuracy than the ones staring with unregularized weights.

To see if our regularization method is effective beyond the CIFAR10 dataset, we also tested performance on more commonly used transfer learning target dataset: Stanford Dog (Khosla et al., 2011) contains images of 120 different kinds of dogs; Oxford Flowers (Nilsback and Zisserman, 2008) contains images of 102 different types of flowers; MIT Indoor (Quattoni and Torralba, 2009) contains indoor scenes of 67 different categories. Three dataset have the same input dimensionality with ImageNet1k. Similar as in finetuning on CIFAR10, we the model pretrained on ImageNet either with or without our regularizations, and in the finetunning stage we perform normal CrossEntropy optimization. As different target dataset have different inherent complexity, we finetune on the three models using different set of hyperparameters. To finetune on Stanford dog, we train for 50 epochs using SGD with a learning rate 0.005 for last layer and 0.001 for the feature layers; to finetune on Oxford Flowers we train for 1000 epochs using SGD with a learning rate 0.005 for last layer and

0.001 for the feature layer; and to finetune on MIT Indoor, we train for 100 epochs using SGD with a learning rate 0.01 and 0.002 for the feature layers. All dataset are trained with a btachsize of 64. With either pretrained weights with or without regularization, we repeat each training for 5 times and report the end test accuracy and adversarial distances in Figure 8. Although finetuned models starting from regularized weights have 2% drop in test accuracy compared to the finetuned models starting with unregularized weights, we have roughly 10% increase in mean adversarial distances across different dataset, suggesting that our feature map trained is not uniquely robust to finetuning on one particular dataset.

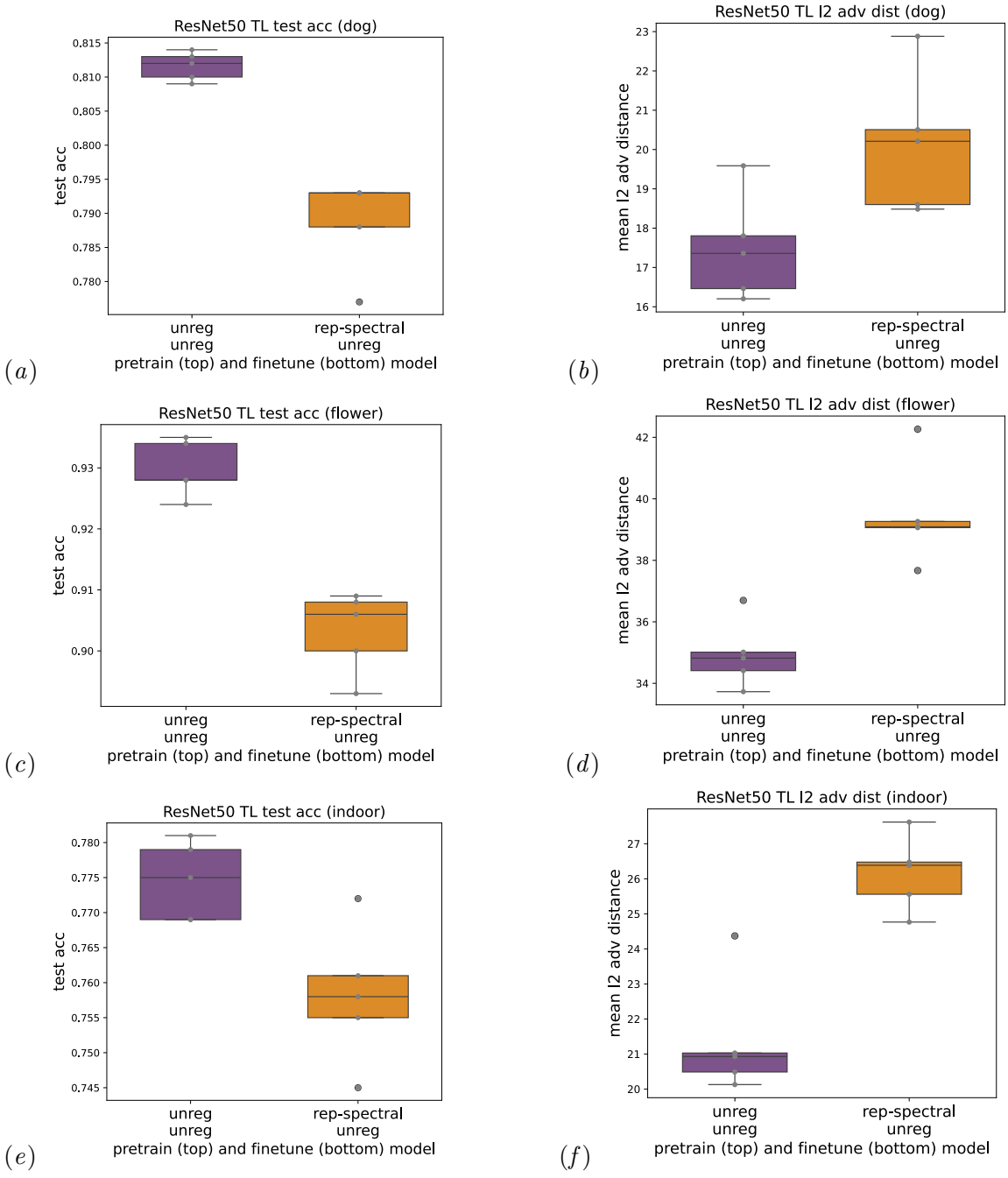


Figure 8: Test accuracy and mean adversarial distances from pretraining on ImageNet and finetuning on Stanford Dog (top row), Oxford Flowers (middle row), MIT Indoor (bottom row). The grey dots indicate the value from each of the 5 random seeds. Sacrificing at most 2% of test accuracy, we obtain at least 10% gain in the adversarial distances on average.

Extended Abstract Track