1-Path-Norm Regularization of Deep Neural Networks

Fabian Latorre [1] **Antoine Bonnet** [1] **Paul Rolland** [1] **Nadav Hallak** [2] **Volkan Cevher** [1]

Abstract

The so-called path-norm measure is considered one of the best indicators for good generalization of neural networks. This paper introduces a proximal gradient framework for the training of deep neural networks via 1-path-norm regularization, which is applicable to general deep architectures. We address the resulting nonconvex nonsmooth optimization model by transforming the intractable induced proximal operation to an equivalent differentiable proximal operation. We compare automatic differentiation (backpropagation) algorithms with the proximal gradient framework in numerical experiments on FashionMNIST and CIFAR10. We show that 1-path-norm regularization is a better choice than weight-decay for fully connected architectures, and it improves the robustness to the presence of noisy labels. In this latter setting, the proximal gradient methods have an advantage over automatic differentiation.

1. Introduction

In deep learning, assessing the generalization ability of neural networks is essential. This is reflected by the interest of the community in finding complexity measures to predict generalization, e.g., Jiang et al. [1]. One measure gaining prominence is the *path-norm* measure [2], which quantifies the complexity or length of the paths taken by the network during training. Comprehensive numerical experiments [3] found that among norm-based and margin-based measures, the path-norm of a neural network is the most positively-correlated with generalization (see Table 2 therein).

Among the family of path-norm complexity measures, the *1-path-norm* stands out as the only one providing *width-independent* generalization bounds for ReLU networks [2]. This is a striking property, as any excess-risk bound depending on the width of the network is otherwise incapable of explaining the good generalization of *overparametrized* neural networks, see for example Neyshabur et al. [4, 5], Novak et al. [6]. Notwithstanding, the path-norm expression is nonconvex and nonsmooth, making it hard to handle in an optimization framework. This might pose great difficulty to the automatic-differentiation (AD) paradigm: software such as PyTorch [7], or TensorFlow [8], may compute *incorrect* gradient information of compositions of non-differentiable functions [9, 10].

Our contributions. • We establish a theoretical connection that relates the 1-path-norm to the Lipschitz constant of networks with arbitrary depth with either differentiable or ReLU activations (Theorem 1). This significantly generalizes the result of Latorre et al. [11, Theorem 1] for shallow networks with differentiable activations.

• We introduce an approximate proximal gradient scheme for 1-path-norm regularization that requires only forward/backward passes through a modified network architecture (Algorithm 4). It is based on a differentiable reformulation of the proximal mapping problem, and uses first-order methods to avoid computing the exact proximal mapping. We verify that this leads to better solutions than AD.

• We study the effects of 1-path-norm regularization on image classification tasks. On FashionMNIST [12] and CIFAR10 [13] we show how 1-path-norm regularization improves the classification error of Fully connected architectures, and leads to more robust models in the presence of noisy data, compared to L2 (weight decay) or no regularization. We show that AD as optimizer performs well in terms of accuracy, compared to proximal methods, despite the lack of theory. However, the proximal gradient methods perform better than AD when we evaluate the robustness to random perturbations of the input data.

2. Preliminaries and Problem Statement

For an L-layer feedforward neural network $f_W(x) := W^L\sigma(W^{L-1}\sigma(\cdots\sigma(W^1x)\cdots))$ with a differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$ and weight matrices with dimensions determined by a sequence of *layer sizes*[1]

*Equal contribution [1]École polytechnique fédérale de Lausanne [2]Technion - Israel Institute of Technology. Correspondence to: Fabian Latorre <fabian.latorre@epfl.ch>.

LatinX in AI workshop at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023.

[1]Here, the 0-th and L-th layer correspond to the input and output layer, respectively.

$d_0, \dots, d_{L-1}, d_L$, its 1-path-norm [2] can be defined as:

$$P_1(W) \coloneqq \mathbb{1}^T \left|W^L\right| \left|W^{L-1}\right| \cdots \left|W^1\right| \mathbb{1} \tag{1}$$

where $\left|W^\ell\right|$ is the matrix obtained by entry-wise application of the absolute value function, and the symbol $\mathbb{1}$ denotes an all-ones column vector with dimension inferred by the context. One of the key properties of the 1-path-norm, is that it controls the *smoothness* of the network as it upper bounds its Lipschitz constant. This is known in the case of shallow networks with differentiable activation functions [11]. This result is more general and holds for networks of arbitrary depth and or networks with ReLU activations.

Theorem 1. *Let $f_W : \mathbb{R}^{d_0} \to \mathbb{R}$, $f_W(x) := W^L \sigma(W^{L-1} \sigma(\cdots \sigma(W^1 x) \cdots))$ be a network such that the gradient of the activation σ is globally bounded between zero and one, i.e., $0 \leq \sigma'(x) \leq 1$ or $\sigma(x) = ReLU(x)$. Choose the ℓ_∞-norm for the input space and $|\cdot|$ for the output space. The Lipschitz constant of the network, denoted by L_W is bounded as follows:*

$$L_W \leq P_1(W) \leq \prod_{\ell=1}^{L} \|W^\ell\|_\infty. \tag{2}$$

The right-hand-side of Equation (2) is usually referred to as the *trivial bound* based on the product of the norms of each weight matrix. Precisely, Theorem 1 states that the 1-path-norm is a better estimator of the ℓ_∞-Lipschitz constant of the network, than the trivial product bound, in the case of single output. The proof is provided in Appendix A. This also motivates the use of the 1-path-norm as a regularizer, given that the the Lipschitz constant is related to the generalization and robustness of the network [14–17].

We can succintly describe the 1-path-norm of a network (1) as the sum of the absolute value of the product of the weights along each path from input to output layer. From now on we will refer to any such choice of one neuron per layer as one *path*. The following is the 1-path-norm regularized empirical risk minimization problem on n labeled training samples $(x_i, y_i) \in \mathbb{R}^{d_0} \times \mathbb{R}^{d_L}$, loss function $\mathcal{L}$ and regularization parameter $\lambda \in \mathbb{R}_{\geq 0}$:

$$\min_W \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f_W(x_i), y_i\right) + \lambda P_1(W) \tag{3}$$

When $L \geq 2$, common losses $\mathcal{L}$, such as the cross-entropy, lead to a composite optimization objective in (3) that is non-convex and non-smooth due to the presence of absolute values and products. Obviously, such a model cannot be solved globally.

Thus, instead of global optimality, we turn to the task of devising algorithms with non-asymptotic rates of convergence to first-order stationarity via the *proximal gradient* approach. Let us recall the definition of the proximal mapping, a well-known concept in optimization [18], in the context of the 1-path-norm: [19, Definition 12.23]:

$$\operatorname{prox}_{\lambda P_1(W)} \in \operatorname*{argmin}_Z \frac{1}{2}\|Z - W\|_F^2 + \lambda P_1(Z). \tag{4}$$

For the type of problem in consideration, eq. (4) constitutes a highly challenging task because it involves solving a non-convex and non-smooth problem. Indeed, in the case of the 1-path-norm, an efficient implementation of the proximal mapping is only known for the case of linear functions and shallow neural networks [11], which hinders the applicability of 1-path-norm regularization for contemporary deep network architectures used in practice.

Proximal-gradient type methods are the only first-order algorithms with guarantees of convergence for composite non-smooth non-convex problems [20]. Hence, it is worth exploring their use for 1-path-norm regularization, and decide if it is a better choice than the automatic differentiation approach. However, the intractability of the proximal operator of the 1-path-norm leaves as only alternative the option of replacing it with principled heuristics.

Algorithm 1 1-PN regularization using AD (Path-AD)

1: **for** $t = 1, \dots, T$ **do**
2: Sample $i_1, \dots, i_b \sim \text{Unif}[n]$
3: $W_{t+1} \leftarrow W_t - \gamma \nabla_W \left[\frac{1}{b}\sum_{j=1}^{b} \mathcal{L}\left(f_{W_t}(x_{i_j}), y_{i_j}\right) + \lambda P_1(W_t)\right]$
4: **return** W_T

Algorithm 2 (Stochastic) Proximal Gradient Descent

1: **for** $t = 1, \dots, T$ **do**
2: Sample $i_1, \dots, i_b \sim \text{Unif}[n]$
3: $W_{t+1/2} \leftarrow W_t - \gamma \nabla_W \frac{1}{b}\sum_{j=1}^{b} \mathcal{L}\left(f_{W_t}(x_{i_j}), y_{i_j}\right)$
4: $W_t \leftarrow \operatorname{prox}_{\gamma\lambda P_1}(W_{t+1/2})$
5: **return** W_T

3. Approximate Prox-Grad Scheme

We propose three strategies to optimize the objective in eq. (3). The first one is to use automatic differentiation directly on such objective (Algorithm 1), together with an off-the-shelf first-order optimization algorithm like SGD or Adam [21]. The second one is to follow the Proximal Gradient Template (Algorithm 2), and replace the proximal mapping by a few steps of automatic differentiation applied to the non-smooth objective defining it (4). This leads to Algorithm 3 (Prox-AD), where the approximation of the prox occurs between line 4 and line 10. The third strategy we propose relies on the following:

Lemma 1. *Let P be a function satisfying $P(W) = P(|W|)$. Its proximal mapping satisfies*

$$\begin{aligned} \mathrm{prox}_P(W) &= \textit{sign}(W) \odot \mathrm{prox}_P^+(|W|) \\ \mathrm{prox}_P^+(X) &\coloneqq \operatorname*{argmin}_{Z \in \mathbb{R}^d_+} \frac{1}{2}\|X - Z\|_F^2 + P(Z). \end{aligned} \tag{5}$$

We present a proof of Lemma 1 in Appendix B. Clearly, the 1-path-norm P_1 satisfies the conditions of Lemma 1, as it defined on the absolute values of the weight matrices $|W^L|, \ldots, |W^1|$. This result states that instead of solving $\mathrm{prox}_{\lambda P_1}$, we can alternatively solve $\mathrm{prox}^+_{\lambda P_1}$ which is a constrained optimization problem over the nonnegative orthant, with a differentiable objective. Indeed, over nonnegative weight matrices, the 1-path-norm is identical to the function $\mathbb{1}^T W_L \cdots W_1 \mathbb{1}$ which is infinitely differentiable. Our third strategy Prox-DIF (Algorithm 4) follows the proximal gradient template, but computes an approximation of the proximal mapping of the 1-path-norm using Projected Gradient Descent on the objective that defines $\mathrm{prox}^+_{\lambda P_1}$ (c.f., eq. (5), Lines 4 and 12 in Algorithm 4), and uses the sign of the weights to recover an approximate solution of $\mathrm{prox}_{\lambda P_1}$ (c.f., Line 10 in Algorithm 4), as Lemma 1 indicates.

Algorithm 3 1-path-norm regularization using AD for the Proximal Mapping. (Prox-AD)

1: **for** $t = 0, \ldots, T-1$ **do**
2: Sample $i_1, \ldots, i_b \sim \mathrm{Unif}[n]$
3: $W_{t+1/2} \leftarrow W_t - \gamma \nabla_W \frac{1}{b} \sum_{j=1}^b \mathcal{L}\left(f_{W_t}(x_{i_j}), y_{i_j}\right)$
4: **if** $t = 0 \pmod B$ **then**
5: $Z_0 = W_{t+1/2}$
6: **for** $t' = 0, \ldots, T'-1$ **do**
7: $Z_{t'+1} = Z_t - \gamma' \nabla_Z \left[\frac{1}{2}\|W_{t+1/2} - Z_{t'}\|_2^2 + \lambda\gamma P_1(Z_{t'})\right]$
8: $W_{t+1} = Z_{T'}$
9: **else**
10: $W_{t+1} = W_{t+1/2}$
11: **return** W_T

Efficiently computing the 1-path-norm for arbitrary architectures. We can efficiently compute the 1-path-norm as it is equivalent to the following process: (1) replace the activations in the network with the identity $\sigma(x) = x$; (2) replace the weights by their absolute values; (3) remove biases; (4) compute the forward pass of the resulting (linear) network on the vector of all-ones $\mathbb{1}$; and (5) sum the outputs. For example, in the case of CNNs, this avoids transforming the kernel matrices into their equivalent representation as a huge doubly circulant matrix (c.f., Sedghi et al. [22]), which would be inefficient memory-wise.

Algorithm 4 Differentiable Proximal training of 1-path-norm regularized NNs (Prox-DIF)

1: **for** $t = 0, \ldots, T-1$ **do**
2: Sample $i_1, \ldots, i_b \sim \mathrm{Unif}[n]$
3: $W_{t+1/2} \leftarrow W_t - \gamma \nabla_W \frac{1}{b} \sum_{j=1}^b \mathcal{L}\left(f_{W_t}(x_{i_j}), y_{i_j}\right)$
4: **if** $t = 0 \pmod B$ **then**
5: $Z_0 = |W_{t+1/2}|$
6: **for** $t' = 0, \ldots, T'-1$ **do**
7: $Z_{t'+1/2} = Z_t - \gamma' \nabla_Z \left[\frac{1}{2}\||W_{t+1/2}| - Z_{t'}\|_2^2 + \lambda\gamma P_1(Z_{t'})\right]$
8: $Z_{t'+1} = \max(0, Z_{t'+1/2})$
9:
10: $W_{t+1} = \mathrm{sign}(W_{t+1/2}) \odot Z_{T'}$
11: **else**
12: $W_{t+1} = W_{t+1/2}$
13: **return** W_T

4. Experiments

Proximal Mapping approximation. We compare the performance of the two proposed algorithms that approximate the proximal mapping of the 1-path-norm. This corresponds to line 4 in algorithm 3, and lines 4 and 12 in algorithm 4. In Figure 1, we observe that for larger values of the regularization parameter λ, Prox-DIF (Algorithm 4) has an advantage over the baseline Prox-AD (Algorithm 3), which uses automatic differentiation directly on the non-smooth objective eq. (4). Prox-DIF reaches lower values of the objective much faster. The differentiability of the Prox-DIF objective allows the use of momentum more effectively. In the case of shallow networks, where the optimum is known (cf. Latorre et al. [11, Algorithm 2]), Prox-DIF almost reaches such value. Nevertheless, as observed in the bottom row of fig. 1, for really small values of λ, the difference between the two methods is pretty small. This suggests that for problems where only a small amount of regularization is needed, there might be no advantage for Prox-DIF .

Impact on the Generalization Error. We train fully-connected networks and CNNs on the FashionMNIST and CIFAR10 benchmark datasets, using different types of regularization. We plot the validation accuracy as a function of the training epoch in Figure 2. We observe that for fully connected networks, the 1-path-norm is a significantly better choice than no-regularization or weight-decay (L2 regularization). We don't observe major differences among the 1-path-norm regularization methods, other than Prox-DIF achieving slightly less accuracy, or Prox-AD achieving a slightly more accurate MLP in CIFAR10. This is probably due to $\lambda = 10^{-6}$ and $\lambda = 10^{-3}$ being small values which lead to higher accuracy. For CNNs, 1-path-norm regularization struggles to improve over weight decay. This can be attributed to the fact that CNNs are

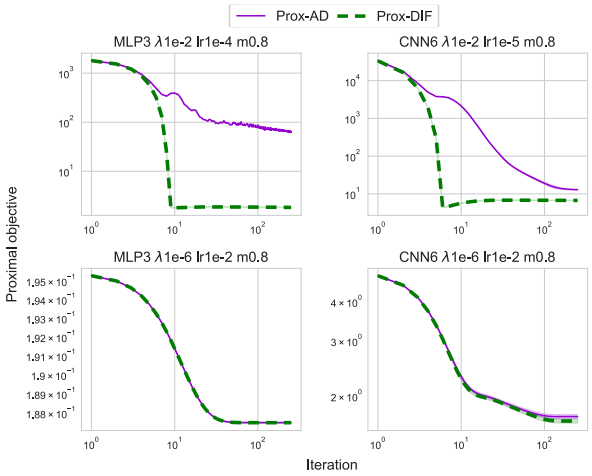

Figure 1. Objective value of the proximal mapping $0.5\|W-Z\|^2+\lambda P_1(Z)$ vs. iteration, using the inner-most loop in the algorithms Prox-AD (Algorithm 3) and Prox-DIF (Algorithm 4), for randomly initialized W. Averaged over 5 random initializations.

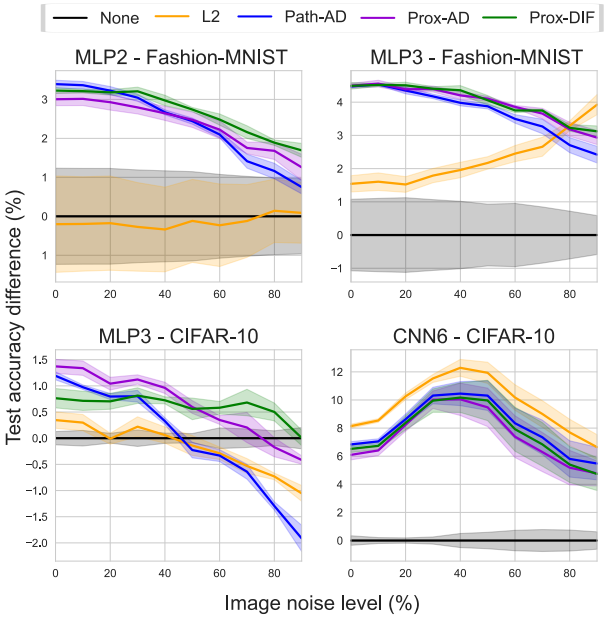

Figure 3. Absolute difference in test accuracy with regards to the unregularized model vs image noise level, for different training algorithms. Averaged over 5 independent runs.

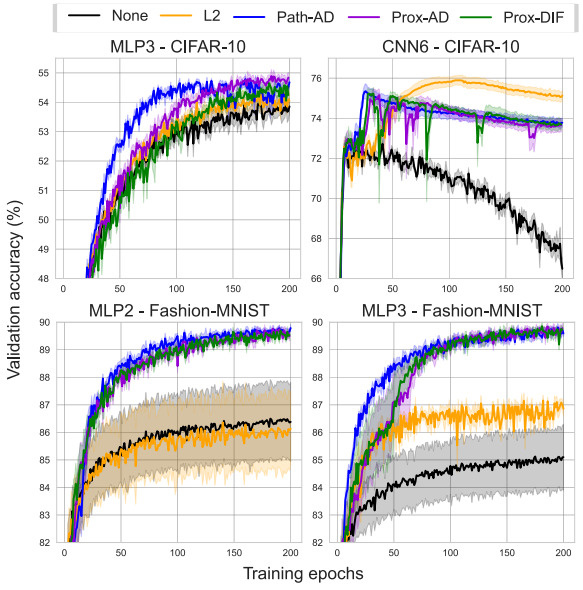

Figure 2. Validation accuracy vs. epoch, for different training algorithms. Averaged over 5 independent runs.

equivalent to highly-sparse fully connected networks [22]. Due to the reduced parameter count in CNNs, the number of different paths from input to output is small by design. As the 1-path-norm controls complexity by reducing the number of paths it can be expected to perform poorly for CNNs. As such, 1-path-norm regularization might be more suitable for architectures like MLP-mixer [23].

Robustness to Noise. We study the effect of regularization in the presence of noisy data. Uniform noise is sampled from an ℓ_∞-ball of varying radius (referred to as the noise level) and added to the test images. In this setting, the 1-path-norm regularization improves all fully-connected networks and achieves significantly higher accuracy than weight-decay or no regularization on the majority of noise levels, for both datasets. In MLPs we observe that the proximal methods Prox-AD and Prox-DIF are better than Path-AD consistently, in particular for CIFAR10 it is crucial to use them to achieve higher robustness, with Prox-DIF achieving higher robustness over Prox-AD for large levels of the noise, in particular visible for FashionMNIST-MLP2 and CIFAR10-MLP3. This is consistent with the sparsity-inducing property of Prox-DIF (line 8). For CNNs 1-path-norm fails to provide benefits, probably due to the already highly sparse structure of the CNNs. Our goal is not to provide state-of-the-art numbers for robustness, rather, regularization can be used together with Adversarial Training [24] to enhance robustness, as is done for example in [25].

5. Conclusions

Our results indicate that 1-path-norm regularization clearly outperforms weight-decay on robustness and generalization tasks using fully-connected networks (MLPs). On the other hand, for highly sparse architectures like CNNs the effect is more muted or negative. Indeed, 1-path-norm works better when the number of effective paths in the network is large i.e., MLPs or MLP-Mixer [23]. Nevertheless, the difference between different optimizers for 1-path-norm regularization seems to be small, but appear more noticeable when the sparsity of the network is crucial, e.g., robustness against noise in the data. Hence, the use of AD for 1-path-norm regularization is not discouraged despite its lack of theory.

1-path-norm regularization is not without drawbacks. First, an initial round of tuning is required for each model architecture to obtain satisfactory parameters for the proximal approximation method. Secondly, the computational cost of the proximal step might slow down training, so a balance struck by skipping the proximal map in some iterations [26] or reducing the number of iterations for the proximal approximation step. Thirdly, 1-path-norm increases exponentially with regards to network depth, which might cause numerical problems for 1-path-norm regularization of really deep neural networks. This issue requires further attention and is left for future work. Finally, it does not consistently improve the robustness in the noisy-labels task.

Acknowledgements

This work is funded (in part) through a PhD fellowship of the Swiss Data Science Center, a joint venture between EPFL and ETH Zurich. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n. 725594 - time-data). This work was supported by the Swiss National Science Foundation (SNSF) under grant number 200021_178865 / 1.

References

[1] Yiding Jiang, Pierre Foret, Scott Yak, Daniel M. Roy, Hossein Mobahi, Gintare Karolina Dziugaite, Samy Bengio, Suriya Gunasekar, Isabelle Guyon, and Behnam Neyshabur. NeurIPS 2020 Competition: Predicting Generalization in Deep Learning. *arXiv e-prints*, art. arXiv:2012.07976, December 2020.

[2] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 1376–1401, Paris, France, 03–06 Jul 2015. PMLR.

[3] Yiding Jiang*, Behnam Neyshabur*, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020.

[4] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *ICLR (Workshop)*, 2015. URL `http://arxiv.org/abs/1412.6614`.

[5] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BygfghAcYX`.

[6] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HJC2SzZCW`.

[7] 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 *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[8] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[9] Sham M Kakade and Jason D Lee. Provably correct automatic sub-differentiation for qualified programs. In *Advances in Neural Information Processing Systems 31*, pages 7125–7135. Curran Associates, Inc., 2018.

[10] Jérôme Bolte and Edouard Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming*, pages 1–33, 2020.

[11] Fabian Latorre, Paul Rolland, Nadav Hallak, and Volkan Cevher. Efficient proximal mapping of the 1-path-norm of shallow networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5651–5661. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/latorre20a.html.

[12] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[14] Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. Robust large margin deep neural networks. *Trans. Sig. Proc.*, 65(16):4265–4280, aug 2017. ISSN 1053-587X. doi: 10.1109/TSP.2017.2708039. URL https://doi.org/10.1109/TSP.2017.2708039.

[15] Cem Anil, James Lucas, and Roger Baker Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, 2018.

[16] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Mach. Learn.*, 110(2): 393–416, feb 2021. ISSN 0885-6125. doi: 10.1007/s10994-020-05929-w. URL https://doi.org/10.1007/s10994-020-05929-w.

[17] Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgöwer. Training robust neural networks using lipschitz bounds. *IEEE Control Systems Letters*, 6:121–126, 2022. doi: 10.1109/LCSYS.2021.3050444.

[18] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, jan 2014. ISSN 2167-3888. doi: 10.1561/2400000003. URL https://doi.org/10.1561/2400000003.

[19] H.H. Bauschke and Patrick Louis Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011. doi: 10.1007/978-1-4419-9467-7. URL https://hal.inria.fr/hal-00643354.

[20] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, jul 2013. doi: 10.1007/s10107-013-0701-9.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 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. URL http://arxiv.org/abs/1412.6980.

[22] Hanie Sedghi, Vineet Gupta, and Philip M. Long. The singular values of convolutional layers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJevYoA9Fm.

[23] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Peter Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-mixer: An all-MLP architecture for vision. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=EI2KOXKdnP.

[24] 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.

[25] Zhenyu Zhu, Fabian Latorre, Grigorios Chrysos, and Volkan Cevher. Controlling the complexity and lipschitz constant improves polynomial nets. In *International Conference on Learning Representations*, 2022.

[26] Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich, and Peter Richtarik. ProxSkip: Yes! Local gradient steps provably lead to communication acceleration! Finally! In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15750–15769. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/mishchenko22b.html.

[27] Tong Chen, Jean B Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic optimization for lipschitz constants of relu networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F.

Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19189–19200. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/dea9ddb25cbf2352cf4dec30222a02a5-Paper.pdf.

A. Proof of theorem 1

We prove theorem 1 by first showing that the 1-path-norm is upper bounded by the product-of-norms lemma 2 and then showing that the 1-path-norm upper bounds the Lipschtiz constant of the network lemma 3.

Lemma 2. *Let $W = [W^1, \dots, W^L]$ be the weight matrices of a fully-connected network with a single output. Then*

$$P_1(W) \leq \Pi_{\ell=1}^{L} \|W^\ell\|_\infty \tag{6}$$

Proof. We proceed by induction. The base case corresponds to shallow networks and is already known as Latorre et al. [11, Theorem 1.]. Assume that the result is true for L layers. Let $W = [W^1, \dots, W^{L+1}]$ where $W^{L+1} \in \mathbb{R}^{1,d_L}$ is a matrix with a single row. In this case we have

$$\begin{aligned}
P_1(W) &= \mathbb{1}^T |W^{L+1}||W^L||W^{L-1}| \dots |W^1|\mathbb{1} \\
&= \sum_{i=1}^{d_L} |W_i^{L+1}||W^L[i,:]||W^{L-1}| \dots |W^1|\mathbb{1} \\
&= \sum_{i=1}^{d_L} |W_i^{L+1}| P_1(W^L[i,:], \dots, W^1) \\
&\leq \sum_{i=1}^{d_L} |W^{L+1}[1,i]| \underbrace{\|W^L[i,:]\|_1}_{\leq \|W^L\|_\infty} \Pi_{\ell=1}^{L-1} \|W^\ell\|_\infty \\
&\leq \sum_{i=1}^{d_L} |W^{L+1}[1,i]| \|W^L\|_\infty \Pi_{\ell=1}^{L-1} \|W^\ell\|_\infty \\
&= \|W^{L+1}[1,:]\|_1 \|W^L\|_\infty \Pi_{\ell=1}^{L-1} \|W^\ell\|_\infty \\
&= \Pi_{\ell=1}^{L+1} \|W^\ell\|_\infty
\end{aligned} \tag{7}$$

where the second equality is due to the definition of the path-norm, the first inequality is due to the induction hypothesis and the second inequality is due to the definition of the ∞-operator norm $\|W\|_\infty = \max_{i=1}^{d} \|W[i,:]\|_1$ is the maximum ℓ_1-norm of the rows. Note that in the last two lines $\|W^{L+1}[1,:]\|_1$ denotes the ℓ_1-norm as a vector while $\|W^{L+1}\|_\infty$ denotes the ∞-operator norm of W^{L+1} as a matrix with a single row. □

Lemma 3. *Let $f_W : \mathbb{R}^{d_0} \to \mathbb{R}$, $f_W(x) = W^L \sigma(W^{L-1}\sigma(\dots \sigma(W^1 x) \dots)$ be a neural network where $W^L \in \mathbb{R}^{d_L \times 1}$, i.e., a single-output network. In the case of a differentiable activation function suppose that that the derivative of the activation is globally bounded between zero and one. For ReLU, we have that the subgradient satisfiest $0 \leq \sigma'(x) \leq 1$. Its Lipschitz constant, denoted as L_W, with respect to the ℓ_∞ norm (for the input space) and absolute-value (for the output space) satisfies the inequality $L_W \leq P_1(W)$.*

Proof. In the case of differentiable activation functions, we have that the ℓ_∞-Lipschitz constant of f_W is equal to the supremum of the ℓ_1-norm of its gradient, over its domain (c.f., Latorre et al. [11, Theorem 1]). In the case of ReLU activations, Chen et al. [27, Lemma 1] ensures that the ℓ_∞-Lipschitz constant is still equal to the supremum of the ℓ_1-norm of the "gradient" computed with the backpropagation algorithm, which applies the chain-rule even in the presence of the non-differentiable ReLU activation. Despite not being the true gradient, we will denote this element as $\nabla f_W(x)$. We use the notation $Df = \nabla^\top f$ for the Jacobian (transpose of gradient).

Denote by $f_W^\ell(x)$ the value of the ℓ-th hidden layer (pre-activation) in the forward pass over the network, that is:

$$f_W^\ell(x) = W^\ell \sigma(W^{\ell-1}\sigma(\dots \sigma(W_1 x) \dots)) \tag{8}$$

We now upper bound the Lipschitz constant as follows, using Latorre et al. [11, Theorem 1] and the chain rule:

$$L_W = \sup_x \|\nabla f_W(x)\|_1 \tag{9}$$

$$= \sup_x \sup_{\|t\|_\infty \le 1} Df_W(x)t \tag{10}$$

$$= \sup_x \sup_{\|t\|_\infty \le 1} W^L \operatorname{Diag}(\sigma'(f_W^{L-1}(x)))W^{L-1} \operatorname{Diag}(\sigma'(f_W^{L-2}(x))) \dots W^1 t \tag{11}$$

$$\le \sup_{0 \le s^\ell \le 1} \sup_{\|t\|_\infty \le 1} W^L \operatorname{Diag}(s^L)W^{L-1} \operatorname{Diag}(s^{L-1}) \dots W^2 \operatorname{Diag}(s^2)W^1 t \tag{12}$$

$$\le \sup_{0 \le s^\ell \le 1} \sup_{\|t\|_\infty \le 1} \widehat{W}^L \widehat{W}^{L-1} \widehat{W}^{L-2} \dots \widehat{W}^2 W^1 t \tag{13}$$

where $\widehat{W^\ell} = W^\ell \operatorname{Diag}(s^\ell)$. We see tthe right hand side of eq. (9) is equivalent to evaluating a fully connected linear neural network on the input t, and then taking the supremum over all possible values of t i.e., $-1 \le t \le 1$ and $s^\ell = \sigma'(f_W^{L-1}(x))$, so $0 \le s^\ell \le 1$ by our assumption on the gradient of the activation (or subgradient in the case of ReLU). The output of a linear neural network precisely corresponds to summing the product of weights over all input-output paths. We arrive at the following:

$$L_W \le \sup_{0 \le s^\ell \le 1} \sup_{\|t\|_\infty \le 1} \sum_{(i_0,\dots,i_L)} W^1[i_1, i_0] t_{i_0} \Pi_{\ell=2}^L \widehat{W}^\ell[i_\ell, i_{\ell-1}] \tag{14}$$

$$L_W \le \sup_{0 \le s^\ell \le 1} \sup_{\|t\|_\infty \le 1} \sum_{(i_0,\dots,i_L)} W^1[i_1, i_0] t_{i_0} \Pi_{\ell=2}^L W^\ell[i_\ell, i_{\ell-1}] s^\ell_{i_{\ell-1}} \tag{15}$$

$$L_W \le \sum_{(i_0,\dots,i_L)} \Pi_{\ell=1}^L |W^\ell[i_\ell, i_{\ell-1}]| = P_1(W) \tag{16}$$

□

B. Proof of lemma 1

Let P be a function satisfying $P(W) = Q(|W|)$. Denote by $\odot$ the element-wise multiplication operation. First note that

$$\|\text{sign}(X \odot W) \odot X - W\|^2 \le \|X - W\|^2 \tag{17}$$

Too see this there are two cases. In the first case X_i has the same sign as W_i so $\text{sign}(X_i W_i) = 1$ and $(\text{sign}(X_i W_i)X_i - W_i)^2 = (X_i - W_i)^2$. In the second case, they have opposite signs, then $|\text{sign}(X_i W_i)X_i - W_i| = |-X_i - W_i| \le |X_i| + |W_i| = |X_i - W_i|$. Now, due to the assumption we also have $P(\text{sign}(X \odot W) \odot X) = P(X)$ i.e., P doesn't change after changing signs of variables. With these observations we have:

$$\frac{1}{2}\|\text{sign}(X \odot W) \odot X - W\|^2 + P(\text{sign}(X \odot W) \odot X) \le \frac{1}{2}\|X - W\|^2 + P(X) \tag{18}$$

this implies

$$\min_X \frac{1}{2}\|\text{sign}(X \odot W) \odot X - W\|^2 + P(\text{sign}(X \odot W) \odot X) \le \min_X \frac{1}{2}\|X - W\|^2 + P(X) \tag{19}$$

Letting $\text{sign}(X \odot W) \odot X = \widehat{X}$ we see that the opposite inequality also holds. Hence,

$$\min_X \frac{1}{2}\|\text{sign}(X \odot W) \odot X - W\|^2 + P(\text{sign}(X \odot W) \odot X) = \min_X \frac{1}{2}\|X - W\|^2 + P(X) \tag{20}$$

Now, we modify the objective function in the left hand side as follows:

$$\begin{aligned} &\frac{1}{2}\|\text{sign}(X \odot W) \odot X - W\|^2 + P(\text{sign}(X \odot W) \odot X) \\ &= \frac{1}{2}\|\text{sign}(W) \odot (\text{sign}(X) \odot X - |W|)\|^2 + P(\text{sign}(X \odot W) \odot X) \\ &= \frac{1}{2}\|\text{sign}(X) \odot X - |W|\|^2 + P(\text{sign}(X) \odot X) \end{aligned} \tag{21}$$

Table 1. Layer dimensions of fully-connected networks.

Model	Dataset	Dimensions		
		Input	Hidden layers	Output
MLP2	FMNIST	$[28, 28, 1]$	$[500, 500]$	10
MLP3	FMNIST	$[28, 28, 1]$	$[500, 500, 500]$	10
MLP3	CIFAR10	$[32, 32, 3]$	$[1024, 512, 256]$	10

Noting that $\text{sign}(X) \odot X$ is a matrix with nonnegative entries, we have the following:

$$\min_{X \geq 0} \frac{1}{2}\|X - |W|\|^2 + P(X) = \min_{X} \frac{1}{2}\|X - W\|^2 + P(X) \tag{22}$$

Hence we see that both objective functions defining $\text{prox}_P(W)$ and $\text{prox}_P^+(|W|)$ have the same value. By the previous arguments the way to transform an element of $\text{prox}_P^+(|W|)$ into an optimal solution of $\text{prox}_P(W)$ is by multiplying by the sign of W, this follows from the last equation in eq. (21). This concludes the result.

C. Experimental details and additional plots

Model Architectures. We now detail the specific model architectures referred to in section 4. The dimensions of the multilayer perceptrons trained on both Fashion-MNIST and CIFAR-10 are listed in the table below. A single convolutional neural network architecture (CNN6) was trained on CIFAR-1. The CNN model consists of an input layer followed by four convolutional layers and two fully connected layers. The convolutional layers each have a kernel of size 3×3, and respectively had $32, 32, 64, 64$ output channels. A max pooling layer with 2×2 kernel size was applied after each pair of convolutional layers. Then, two fully-connected layers were added, respectively of widths 1600 and 512. ReLu activation was applied after each convolutional layer except the last fully connected layer. No dropout was used for any of the models, as we aimed to isolate the effects of each regularization method. All models were trained with batches of 64 samples.

Hyperparameter selection. Training hyperparameters, such as step size γ, momentum β and regularization strength λ were selected through independent grid search for each regularization method, model architecture and dataset. An initial round of grid search was run on a wide grid of parameters for 50 training epochs on 3 independent runs. The best hyperparameters were then selected for each model, then the models were retrained for 200 training epochs on 5 independent runs. Model training included early stopping, by which each model's training was halted when maximal validation accuracy was reached. All grid searches included both stochastic gradient descent (SGD) and Adam optimizers.

Additional parameters were required by the Prox-AD (algorithm 3) and Prox-DIF (algorithm 4) regularization methods for the optimizer of the proximal objective; namely the inner step size γ' and inner momentum β'. These parameters were pre-tuned by minimizing the proximal objective eq. (4) on randomly initialized network weights and a wide range of regularization strengths $\lambda' = \lambda \cdot \gamma$ on for each architecture (refer to algorithm 1 for notation and Figure 1 for examples). During grid search for the outer loop parameters, the inner loop optimizer and parameters were selected for both Prox-AD and Prox-DIF as those minimizing the average final proximal objective after 250 iterations of Prox-AD over 5 independent runs. Once the best combination of the above parameters were selected, a third and final round of grid search was performed to tune the number of inner iterations T' and skip-prox parameter B. The values for the skip-prox parameter were chosen as divisors of the total number of batches so that every epoch training would end with a proximal step. All selected parameters are shown in the table below.

Noise robustness. Training hyperparameters were selected at noiseless conditions then reused for the noise robustness experiments at all noise levels, since the same parameters were often found to yield the highest performance under all noise conditions. The robustness to noisy data of different regularization methods was compared by evaluating the accuracy of each trained model on the same test set at different noise levels, as shown in (fig. 3-bottom). Examples of images at varying amounts of image corruption with uniform random noise are shown below.

Table 2. Training parameters selected for each regularization method.

Model	Dataset	Regularization	Hyperparameters
MLP2	FMNIST	None	SGD; $\gamma = .1, \beta = .6$
		L2	SGD; $\gamma = .1, \beta = .6, \lambda = 10^{-4}$
		Path-AD	SGD; $\gamma = .1, \beta = .9, \lambda = 10^{-6}$
		Prox-AD	Outer: SGD; $\gamma = .1, \beta = .9, \lambda = 10^{-3}$ Inner: Adam; $\gamma' = 10^{-4}, T' = 250, B' = 125$
		Prox-DIF	Outer: SGD; $\gamma = .1, \beta = .9, \lambda = 10^{-3}$ Inner: Adam; $\gamma' = 10^{-4}, T' = 250, B' = 125$
MLP3	FMNIST	None	SGD; $\gamma = 10^{-2}, \beta = .9$
		L2	SGD; $\gamma = .1, \beta = .5, \lambda = 10^{-3}$
		Path-AD	SGD; $\gamma = .1, \beta = .9, \lambda = 10^{-6}$
		Prox-AD	Outer: SGD; $\gamma = .1, \beta = .8, \lambda = 10^{-4}$ Inner: Adam; $\gamma' = 10^{-3}, T' = 250, B' = 125$
		Prox-DIF	Outer: SGD; $\gamma = .1, \beta = .9, \lambda = 10^{-3}$ Inner: Adam; $\gamma' = 10^{-3}, T' = 250, B' = 125$
MLP3	FMNIST	None	SGD; $\gamma = .1, \beta = .1$
		L2	SGD; $\gamma = .1, \beta = .1, \lambda = 10^{-4}$
		Path-AD	SGD; $\gamma = .1, \beta = .8, \lambda = 10^{-7}$
		Prox-AD	Outer: SGD; $\gamma = .1, \beta = .7, \lambda = 10^{-3}$ Inner: Adam; $\gamma' = 10^{-3}, T' = 250, B' = 125$
		Prox-DIF	Outer: SGD; $\gamma = .1, \beta = .7, \lambda = 10^{-3}$ Inner: Adam; $\gamma' = 10^{-3}, T' = 250, B' = 125$
MLP3	FMNIST	None	SGD; $\gamma = 10^{-2}, \beta = .9$
		L2	SGD; $\gamma = 10^{-2}, \beta = .9, \lambda = 10^{-4}$
		Path-AD	SGD; $\gamma = 10^{-2}, \beta = .9, \lambda = 10^{-9}$
		Prox-AD	Outer: SGD; $\gamma = 10^{-2}, \beta = .9, \lambda = 10^{-5}$ Inner: Adam; $\gamma' = 10^{-4}, T' = 200, B' = 125$
		Prox-DIF	Outer: SGD; $\gamma = 10^{-2}, \beta = .9, \lambda = 10^{-5}$ Inner: Adam; $\gamma' = 10^{-4}, T' = 200, B' = 125$

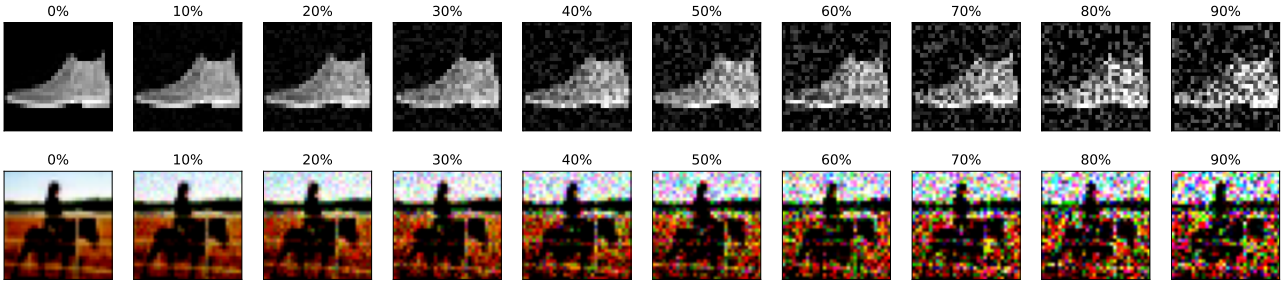

Figure 4. Examples of noisy images generated from the Fashion-MNIST dataset (top) and the CIFAR-10 dataset (bottom), at various noise levels ranging from 0% to 90%.