SN Computer Science (2021) 2:113
https://doi.org/10.1007/s42979-021-00514-x

ORIGINAL RESEARCH

The Coupling Effect of Lipschitz Regularization in Neural Networks

Nicolas Couellan[1,2]

Received: 10 November 2020 / Accepted: 8 February 2021 / Published online: 25 February 2021

Abstract
We investigate the robustness of feed-forward neural networks when input data are subject to random uncertainties. More specifically, we consider regularization of the network by its Lipschitz constant and emphasize its role. We highlight the fact that this regularization is not only a way to control the magnitude of the weights but has also a coupling effect on the network weights across the layers. We claim and show evidence on regression and classification datasets that this coupling effect brings a trade-off between robustness and expressiveness of the network. This suggests that Lipschitz regularization should be carefully implemented so as to maintain coupling across layers.

Keywords Artificial neural networks · Deep learning · Robustness · Regularization · Lipschitz constant

Introduction

With the increasing interest in deep neural networks, a lot of research work has been focusing lately on their sensitivity to input perturbations [3, 10, 14, 15]. Most of these investigations have highlighted their weakness to handle adversarial attacks and the addressed means to increase their robustness. Adversarial attacks are real threats that could slow down or eventually stop the development and applications of these tools wherever robustness guarantees are needed. If input uncertainty is not adversarial but simply generated by the context and environment of the specific application, deep networks do not show better behavior in terms of robustness. If no immunization mechanism is used, their generalization performance can greatly suffer. Actually, most of the techniques that are used to handle adversarial attacks can be also applied in this context. These techniques can mostly be classified into two categories: robust optimization techniques that consider an adversarial loss [10, 15] and regularization techniques that penalize noise expansion throughout the network [4, 7, 12, 18, 21, 22].

Nicolas Couellan
nicolas.couellan@recherche.enac.fr

1 ENAC, Université de Toulouse, 7 Avenue Edouard Belin, 31400 Toulouse, France

2 Institut de Mathématiques de Toulouse, Université de Toulouse, UPS IMT, 31062 Toulouse Cedex 9, France

In this article, we address the second type of techniques where robustness can be achieved by ensuring that the Lipschitz constant of the network remains small. The network can be seen as a mapping between inputs and outputs. Its robustness to input uncertainties can be controlled by how much the mapping output expands the inputs. In the case network with Lipschitz continuous activations, this is equivalent to controlling the Lipschitz constant of the whole network mapping.

Expressiveness is another important property of the neural network. It defines the ability of the network to represent highly complex functions. It is achieved by depth [2, 13]. Of course, such ability is also to be balanced with its generalization power to avoid over-fitting during training. On one hand, if the weights of the networks are free to grow too high, the generalization power will be low. On the other hand, if the weights are overly restricted, the expressiveness of the network will be low. Usually, this trade-off is controlled by constructing a loss that accounts for both training error and generalization error, the so-called regularized empirical risk functional. A parallel has been established between robustness and regularization [20]. Actually in the case of support vector machines, it has been shown that both are equivalent. The work on deep regularized networks we have mentioned above suggests that this is also true for neural networks. The idea we are developing here is a contribution along this line.

We argue that Lipschitz regularization does not only restrict the weights magnitude but it also implements a coupling mechanism across layers, allowing some weights to grow high while others are allowed to vanish. This happens across layers, meaning that the Lipschitz constant can remain small with some large weights values in one layer while some other weight in other layers counterbalance this growth by remaining small. On the contrary, if all weights are restricted uniformly across the network, even though network depth is large, the expressiveness of the network may be inhibited. Through numerical experiments on regression and classification datasets, we show evidence of this phenomenon and draw some conclusions and recommendations about software implementation of Lipschitz regularization.

In "Neural Network Robustness as a LipschitzConstant Regularization Problem", we introduce the relationship between network robustness and Lipschitz regularization and discuss the coupling mechanism taking place. "Experiments" is an illustration of the phenomenon through numerical experiments and "Conclusions" concludes the article.

Neural Network Robustness as a Lipschitz Constant Regularization Problem

Propagating Additive Noise Through the Network

Fully Connected Networks

Consider feed-forward fully connected neural networks that we represent as a successive composition of linear weighted combination of functions such that $x^l = f^l(W^l x^{l-1} + b^l)$ for $l = 1, \dots, L$, where $x^{l-1} \in \mathbb{R}^{n_{l-1}}$ is the input of the l-th layer, the function f^l is the L_f^l-Lipschitz continuous activation function at layer l, and $W^l \in \mathbb{R}^{n_l \times n_{l-1}}$ and $b^l \in \mathbb{R}^{n_l}$ are the weight matrix and bias vector between layer $l-1$ and l that define our model parameter $\theta = \{W^l, b^l\}_{l=1}^L$ that we want to estimate during training. The network can be seen as the mapping $g_\theta : x^0 \to g_\theta(x^0) = x^L$. The training phase of the network can be written as the minimization of the empirical loss $\mathcal{L}(x, y, \theta) = \frac{1}{n}\sum_{i=1}^n l_\theta(g_\theta(x_i), y_i)$ where l_θ is a measure of discrepancy between the network output and the desired output.

Assume now that the input sample x_i is corrupted by some bounded additive noise δ_i such that for all i in $\{1, \dots, n\}$, $\|\delta_i\| \le \Gamma_i$ for some positive constant Γ_i. We define $\tilde{x}_i^l = x_i^l + \delta_i^l$ as the noisy observation of x_i^l that we obtain after propagating a noisy input through layer l. We can write $\|\delta_i^l\| = \|\tilde{x}_i^l - x_i^l\| = \|f^l(W^l\tilde{x}_i^{l-1}) - f^l(W^l x_i^{l-1})\|$ that can be upper bounded by $L_f^l\|W^l(\tilde{x}_i^{l-1} - x_i^{l-1})\| = L_f^l\|W^l\delta_i^{l-1}\|$ since f^l is L_f^l-Lipschitz continuous. Therefore, the network mapping g_θ is L_{g_θ}-Lipschitz continuous and the propagation of the input noise throughout the whole network, leads to an output noise that satisfies the following:

$$\|\delta_i^L\| \le L_f\|W^1\| \times \|W^2\| \times \cdots \times \|W^L\|\Gamma_i$$

where $L_f = \prod_{l=1}^L L_f^l$, $\|W^l\|$ denotes the operator norm:

$$\|W^l\| = \sup_{x \in \mathbb{R}^{n_l*}} \frac{\|W^l x\|}{\|x\|}$$

and the quantity $\hat{L}_{g_\theta} = L_f\|W^1\| \times \|W^2\| \times \cdots \times \|W^L\|$ is actually an upper bound of L_{g_θ}.

Many common activation functions are actually 1-Lipschitz (ex: ReLu, hyperbolic tangent an other sigmoid functions) and therefore $\hat{L}_{g_\theta}$ can often be simplified as $\hat{L}_{g_\theta} = \prod_{l=1}^L \|W^l\|$.

Convolutional Networks

The case of convolutional neural networks (CNN) is not actually different from the case of fully connected networks. Indeed, a convolutional layer performs discrete convolutions with several filters. This can be seen as a weight matrix multiplication operation where the weight matrix is very sparse due to parameter sharing and sparse connectivity induced by the small size of filters usually used in practical applications. Therefore, the output of a l-th convolutional layer can also be written as $x^l = W^l x^{l-1} + b^l$ where x^{l-1} is a flattening transformation of a tensor X^{l-1} of feature maps (transformation of tensor into vector) and b^l is also a flattened version of the bias of the l-th layer (see [6, 7] for more details about the linearity of convolutional layers).

In convolutional architecture, usually convolution is combined with other types of layers. Activation layers (ex: ReLu activation) provide non linearity to the network and, as mentioned above, we often have $L_f^l = 1$. Pooling layers are also used to reduce the dimension of feature maps by summarizing extracted information and can also be viewed as an operator on the output of the convolutional layer that has a Lipschitz constant smaller than 1 [7].

Therefore, from the point of view of noise contraction or expansion developed above, combining the various types of layers usually found in CNN architecture, is exactly the same framework as the fully connected network case. Hence, the techniques developed in the following fully apply to deep network architectures and the conclusions that are drawn in the sequel of the article are also valid.

Network Noise Contraction by Controlling its Lipschitz Constant

In this section, for simplicity, but without loss of generality, we will consider 1-Lipschitz activation functions, meaning that $L_f = 1$ (this is for example the case of ReLu functions).

We have just seen that the quantity $\hat{L}_{g_\theta}(\theta)$, which equals to $\prod_{l=1}^L \|W^l\|$ here, says how much the input noise will be expanded after propagation through the network. Therefore, if during the training process, we ensure that this quantity remains small, we also ensure that input uncertainties will not be expanded by the successive neurons layers. There are two ways to control this quantity during training:

Constrained optimization: The idea is to solve the following empirical risk minimization problem:

$$\min_{\theta=(W^l,b^l)_{l=1}^L} \mathcal{L}(x,y,\theta) \text{ st } \prod_{l=1}^{L} \|W^l\| \leq L_{max}$$

where $L_{\max}$ is a positive parameter. The difficulty with this approach is the non linearity of the constraint. One would like, for example, to use a projected stochastic gradient method to solve the training problem. However, projecting onto this constraint is a difficult problem. To do so, in [7, 21], the authors have proposed to compute and restrict $\|W^l\|$ layer by layer instead of restricting the whole product. Restricting the norm of the weights layer by layer is actually very different from restricting the product of their norms. The layer by layer process isolates the tuning of weights while if the whole product is considered, some layers may be privileged against other. We will see in the next section how this can affect, for some datasets the expressiveness of the network.

Lipschitz regularization: The alternative is to introduce a regularization term in the loss as follows:

$$\min_{\theta=(W^l,b^l)_{l=1}^L} \frac{1}{\lambda}\mathcal{L}(x,y,\theta) + \prod_{l=1}^{L} \|W^l\|$$

where λ is a positive parameter. There are no projection involved, the regularization acts through the addition of a correction term in the gradient of the loss so as to ensure a low value of the Lipschitz constant. The gradient of the regularized loss, denoted $\mathcal{L}_r$ can be written as:

$$\nabla_\theta \mathcal{L}_r(x,y,\theta) = \frac{1}{\lambda}\nabla_\theta \mathcal{L}(x,y,\theta) + \nabla_\theta \hat{L}_{g_\theta}(\theta)$$

and, as mentioned above, depends on the complete cross-layer structure via $\nabla_\theta \hat{L}_{g_\theta}(\theta)$. To further emphasize this coupling effect, under the assumption that $L_f = 1$, we can rewrite $\hat{L}_{g_\theta}(\theta)$ as

$$\hat{L}_{g_\theta}(\theta) = \sqrt{\lambda_{\max}({W^1}^\top W^1)} \times \cdots \times \sqrt{\lambda_{\max}({W^L}^\top W^L)}$$

where $\lambda_{\max}(A)$ denotes the largest eigenvalue of matrix A. We see in this last expression that, if we could rotate the matrix ${W^l}^\top W^l$ at each layer l such that the principal axes are aligned with its eigenvectors, the upper bound $\hat{L}_{g_\theta}(\theta)$ would only depend on the product layer by layer of the largest weight length along these axes. The upper bound could then be seen as a layer by layer product of weight "size". This also means that if at one layer, weights are small, there is room for increase at another layer as long at the whole product remains small. In this sense, we say that the weights have more degrees of freedom than when they are restricted at each layer independently.

This is also illustrated on a very simple example in Fig. 1(right). We take the very simple case of a one hidden layer neural network with one input, one hidden and one output neuron, meaning that the parameter $\theta = (W^1, W^2)$ belongs to $\mathbb{R}^2$. In this case, the bound $\hat{L}_{g_\theta}(\theta)$ is equal to $|W^1| \times |W^2|$. The right figure shows the boundary $|W^1| \times |W^2| = 1$ and the center square defines the set $\{(W^1, W^2) \in \mathbb{R}^2 : |W^1| \leq 1, |W^2| = 1\}$ which is the restriction of each layer weight matrix to 1 independently. It clearly shows that this layer by layer restriction is more conservative than the Lipschitz upper bound that allows some W^1 to

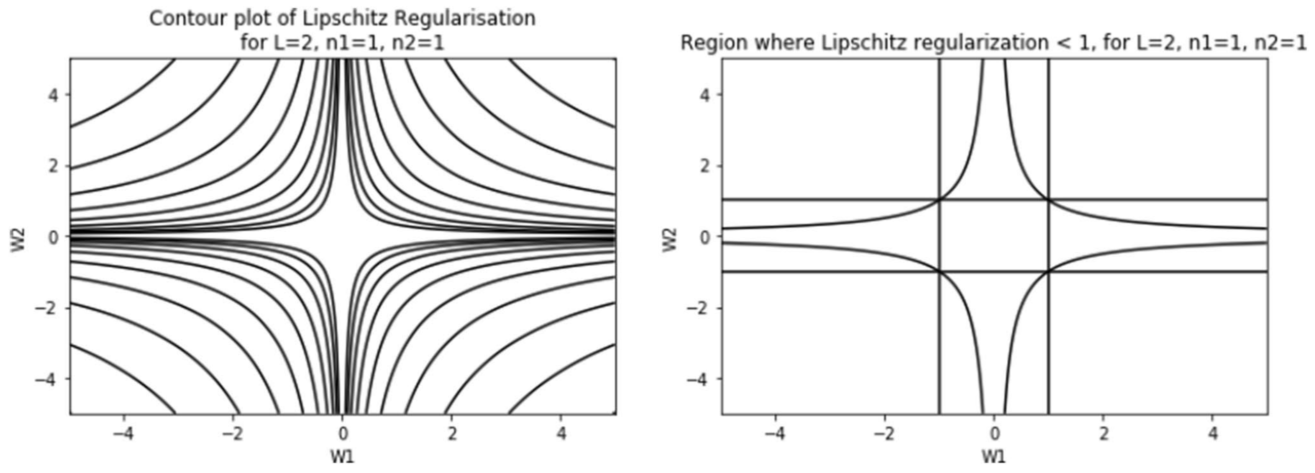

Fig. 1 Contours of $\hat{L}_{g_\theta}(\theta)$ (left) when $\theta \in \mathbb{R}^2$ and regions where $\hat{L}_{g_\theta}(\theta) < 1$ and $|W^1| < 1, |W^2| < 1$ with $\theta = (W^1, W^2)$ (right)

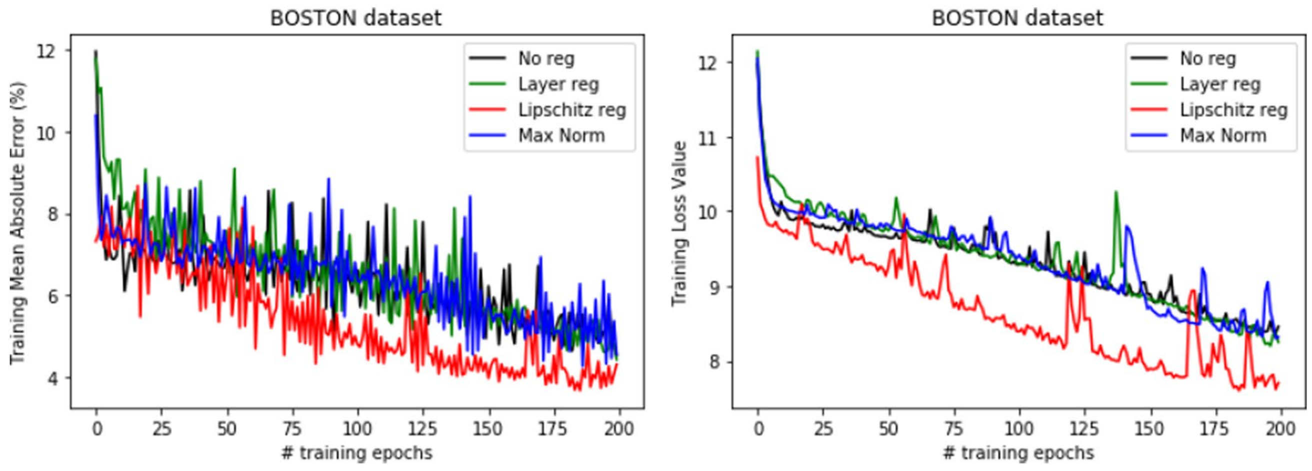

Fig. 2 Training mean average error and training loss profiles (Boston dataset)

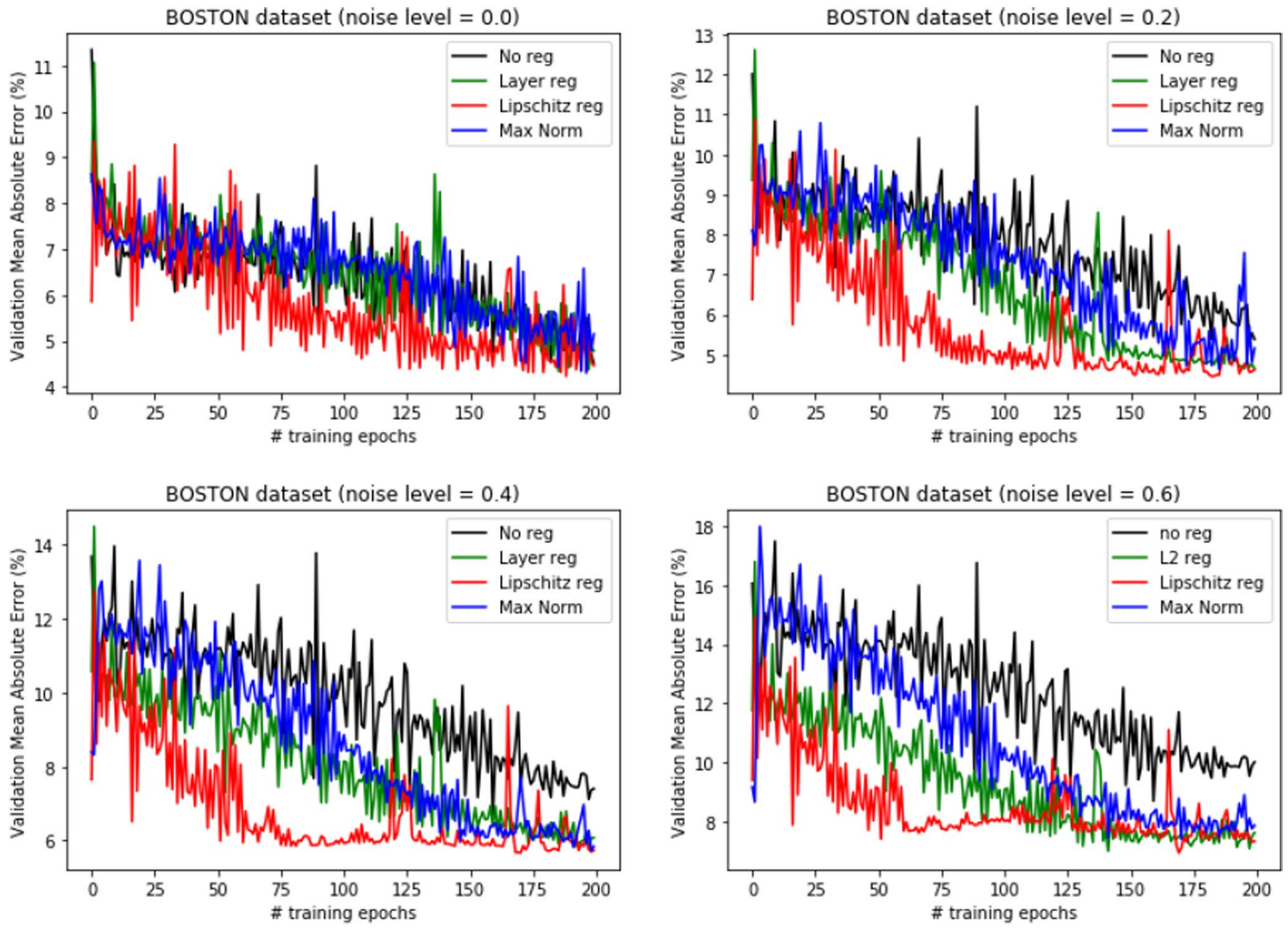

Fig. 3 Mean absolute validation error profiles (Boston dataset)

be large when W^2 is small or the other way around. This is what we refer to as the coupling mechanism across layers. Figure 1 only shows a very low dimensional case. If the dimension is large, it is easy to understand that, for a specific level of the regularization, the feasible set of the weights for the Lipschitz regularization will be much larger than the feasible set of restricted weights at each layer. We argue and show in the next section that for some specific datasets, this extra feasible volume for the weights enables better expressive power of the network. Note that, computationally, there are several difficulties with this coupling approach:

First, the Lipschitz regularization or more precisely, its upper bound is a non-convex function as shown for example on the 2D example of Fig. 1(left). Minimizing such a function may be difficult and available training algorithms such as stochastic gradient techniques or its variants may get trapped in a local minimum. This is not specific to this case. Non-convex regularization techniques have been proven to be effective in other contexts while facing the same difficulty [19]. However, in practice, the benefit is often confirmed.

The computation of $\nabla_\theta \hat{L}_{g_\theta}(\theta)$ may also be difficult. In practice, it requires the use of numerical differentiation since there is no simple explicit expression of the gradient. In [21], alternatively, the authors have used a power iteration method to approximate the operator norm. Observe, however, that since the network parameter values must already be stored at any time during the training process, the computation of $\nabla_\theta \hat{L}_{g_\theta}(\theta)$ does not increase storage requirements.

With respect to these numerical difficulties, please also note that we only emphasize the role of the coupling effect of the Lipschitz regularization and we do not claim to provide efficient techniques to handle it especially on large problems. However, we want to point out that the future development of efficient robust neural network algorithms should preserve the cross-layer structure of the regularization. The design of methods that isolate layers are of course computationally interesting but will loose some of the property of Lipschitz regularization and may turn out to be over-conservative in terms of robustness, at least on some datasets, as we will see in the next section.

Table 1 Spectral norm of layer weights and network Lipschitz constant upper bound

Noise Level	No reg	Layer reg	Lipschitz reg	Max Norm
$\|\|W_1\|\|$	2.828	2.508	3.464	2.383
$\|\|W_2\|\|$	1.954	1.625	1.791	1.883
$\|\|W_3\|\|$	1.825	3.315	1.983	1.707
$\hat{L}_{g_\theta}(\theta)$	10.01	13.52	12.31	7.66

Experiments

In the following experiments, four various training regularized loss formulations are compared:

– (No reg) no regularization
– (Layer reg) spectral norm regularization at each layer (no coupling)
– (Lipschitz reg) Lipschitz regularization across layers (with coupling)
– (MaxNorm) MaxNorm constraint on weights (max value = 10 on weights at each layer) as described in [16].

The first experiment is a regression task where the mean average error, the mean average validation error, the loss values and the spectral norm of the weights matrix of each layer is analyzed during training for the four strategies. The second experiment is focusing on a pattern recognition task (classification) on a larger dataset (60000 training samples, 10000 validation samples), the MNIST dataset [11] and results are expressed in terms of training and validation accuracy.

Neural Network Regression with the Boston Dataset

To illustrate the coupling effect discussed above, we consider the neural network regression task with the Boston dataset [8]. For this purpose, we use a 5 layers feed-forward fully connected network using ReLu activation functions. The hidden layers each have 20 neurons. For training and testing, we use the keras library [1] under the python [17] environment.

The ADAM optimization algorithm [9] is used for training. To implement the "Layer reg" and "Lipschitz reg", a custom regularization and a custom loss were created respectively in keras. The training procedure was set to 200 epochs with a batch size of 50. The regularization parameter was selected by grid search. The training is carried out on a fraction of 4/5 of the entire dataset without perturbing the input data. However, we validate the various formulations with several levels of test uncertainties to evaluate the robustness of each formulation on the remaining fraction of the data. The noisy test inputs are generated as follows:

$$\tilde{x}_i = x_i + \delta_i \quad \forall i \in T$$

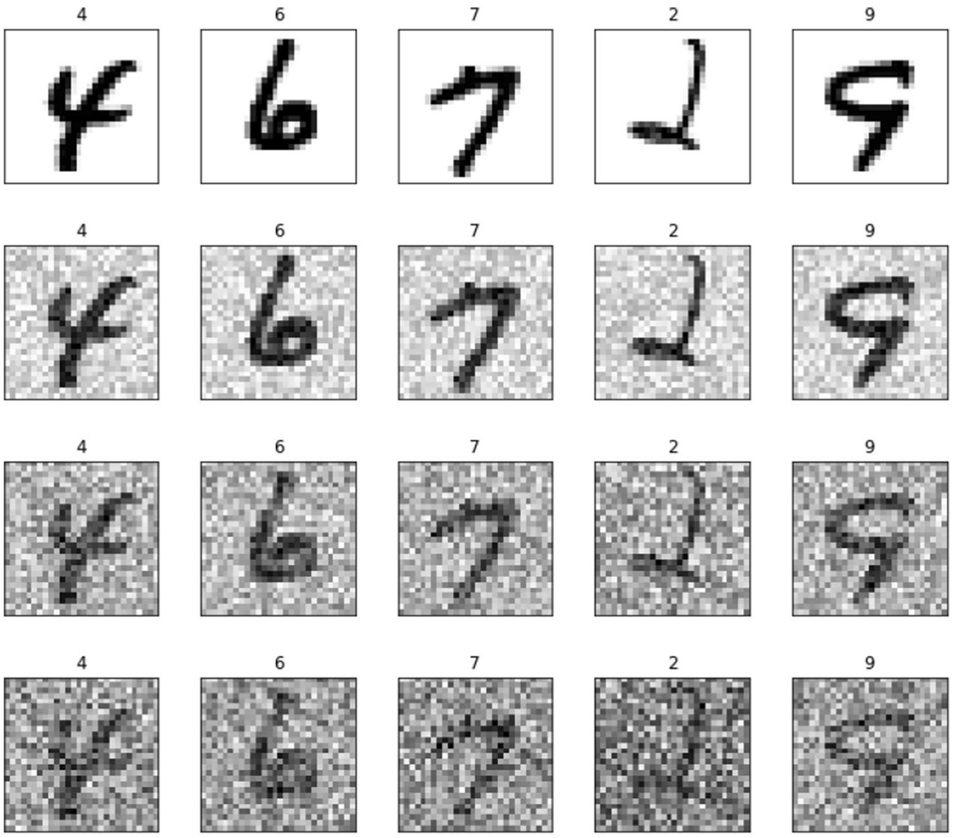

Fig. 4 MNIST dataset samples with various levels η of random noise ($\eta \in \{0.0, 0.2, 0.4, 0.6\}$ from top to bottom)

where T is the test set, x_i is a nominal input set aside before training from the Boston dataset and δ_i is an additive uncertainty such that $\delta_i = \eta(x_i^{\max} - x_i^{\max})u_i$ where $u_i \sim U([0,1])$ (the uniform distribution on the interval [0, 1]), $\eta \in \{0, 0.2, 0.4, 0.6\}$ is the noise level and $x_i^{\min}$ and $x_i^{\max}$ are the vectors of minimum values and maximum values for each input features. Figure 2 provides the training mean average error and loss profiles during training while Fig. 3 gives the mean average validation error for the various noise levels η.

During training, Fig. 2 shows that the Lipschitz regularization achieves better mean absolute error and loss values than the other techniques that achieve all together similar results. One could suspect over-fitting of the data during training with Lipschitz regularization but the validation phase on unseen data as shown in Fig. 3 actually does not confirm this. The mean absolute validation error achieved by the Lipschitz regularization is better than the other methods. This is confirmed for all levels of uncertainties, meaning that Lipschitz regularization is, for the Boston dataset, the technique that provides the highest level of robustness. More specifically, it is worth noticing that the layer-by-layer Lipschitz regularization as opposed to the Lipschitz regularization across layer is not performing well here. At the highest noise level $\eta = 0.6$, the Lipschitz regularization achieves a low mean absolute validation error twice as fast as the layer-by-layer approach. The MaxNorm approach achieves better results than the non regularized model but is behind the layer-by-layer approach. These results are consistent with the fact that Lipschitz regularization provides a good level of accuracy. When looking at Table 1, we can observe some significant differences in the spectral norms of layer weights for the various network instances. All formulations tend to emphasize the first layer except the layer-by-layer approach that allocates more weight mass at the last layer. The Lipschitz regularization across layers tends also to achieve higher weight values than others, which is natural when considering the shape of the regularization as shown in Fig. 1. The value of the network Lipschitz constant are similar except for the MaxNorm case

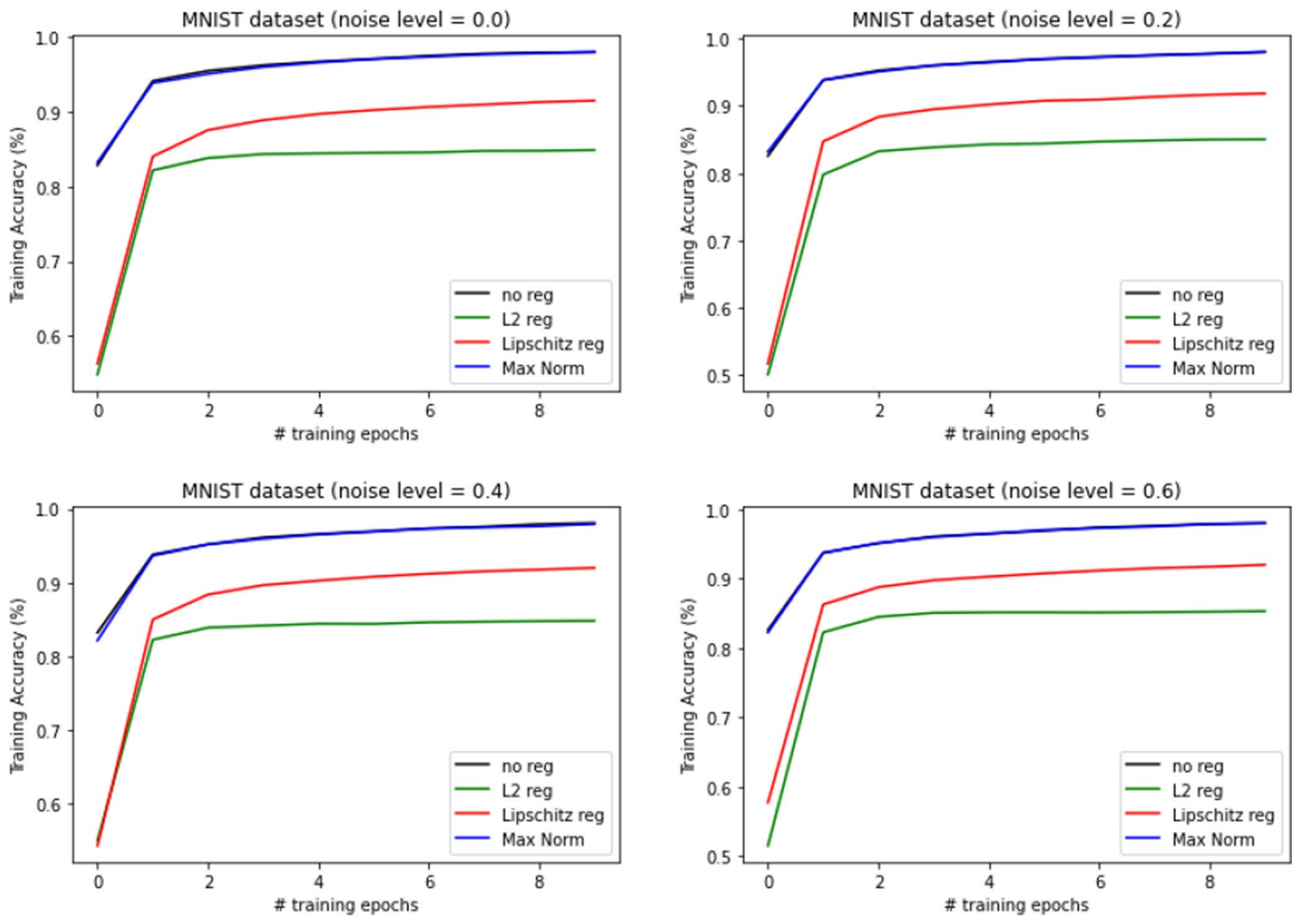

Fig. 5 Training accuracy profiles (MNIST dataset)

that tends to restrict more the weights at all layers. This example confirms that the Lipschitz regularization (across layers) provides more freedom to the weights than other techniques for the same value of the network Lipschitz constant. This confirms that, for some datasets, letting the regularization play a coupling mechanism across layers helps in finding the best compromise between robustness and expressiveness of the network.

Neural Network Classification with the MNIST Dataset

In this classification experiment, with use the MNIST dataset with 60000 training samples and 10000 validation samples. Each sample is a 28×28 image of handwritten characters. As before, a 5 layers feed-forward fully connected network with ReLu activations for input and hidden layers and a softmax activation at the output layer is used. For the last layer, since the Lipschitz constant of the softmax function can be bounded by 1 [5], the regularized loss in the *Lipschitz reg* formulation is taken as $\frac{1}{\lambda}\mathcal{L}(x, y, \theta) + \prod_{l=1}^{L-1} \|W^l\|$. During the experiment, the ADADELTA optimization algorithm [23] was outperforming the ADAM algorithm, therefore it was taken as the common training optimization method. Only a few epochs are required to achieve very good validation accuracy (above 0.9) for the MNIST dataset and early stopping is therefore applied after 10 epochs. The exact same perturbation process from the first experiment is used here again with $\eta \in \{0, 0.2, 0.4, 0.6\}$. Figure 4 shows 5 input samples where various noise level η was applied. Figure 5 reports the accuracies during training on the training samples and for the various η values while Fig. 6 reports the validation accuracies on the validation set.

On Figs. 5 and 6, when $\eta = 0$, the algorithms that less constrain weights are outperforming the others as they allow better expressiveness of the network. However as noise is introduced ($\eta > 0$), this is not the case, anymore, the Lipschitz and L2 regularizers are achieving the best validation accuracies while the others seem to overfit training data and the trend is confirmed as η grows (with low levels of validation accuracy for all algorithms when $\eta = 0.6$). This experiment shows how the Lipschitz and L2 regularizers

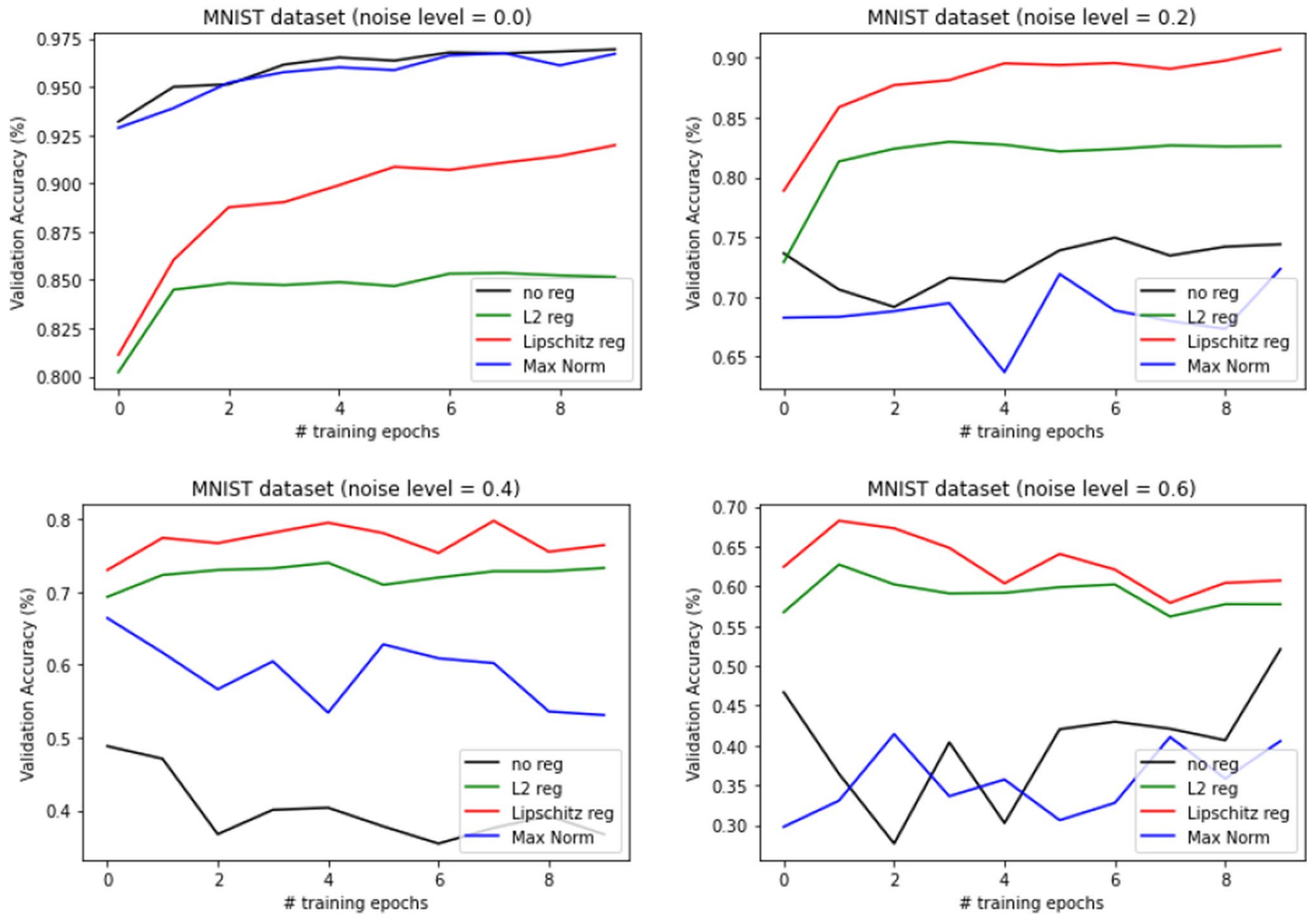

Fig. 6 Validation accuracy profiles (MNIST dataset)

are acting as robust regularizers by trying to reduce the Lipschitz constant of the network. Again, in this case, as η grows, the Lipschitz regularizer that balances weights values across layers achieves better validation accuracy than the L2-regularizer. This shows, as in the previous experiment, that the cross-layer robust strategy tends to allow better expressiveness than the layer-by-layer strategy in place in the L2 regularization.

Conclusions

In this article, we have discussed some properties of Lipschitz regularization used as a "robustification" method in neural networks. Specifically, we have shown that this regularization does not only control the magnitude of the weights but also their relative impact across the layers. It acts as a coupling mechanism across layers that allows some weights to grow when other counterbalance this growth to control the expansion of noise throughout the network. Most of Lipschitz regularization implementations we are aware of actually isolate the layers and do not benefit from the coupling mechanism. We believe that this knowledge should be useful in the future and help designing robust neural network training that achieves also good expressiveness properties.

Compliance with ethical standards

Conflict of interest The author states that there is no conflict of interest.

References

1. Chollet F, et al. Keras. GitHub. https://github.com/fchollet/keras. 2015.
2. Eldan R, Shamir O. The power of depth for feedforward neural networks. In: COLT; 2016.
3. Fawzi A, Fawzi O, Frossard P. Analysis of classifiers' robustness to adversarial perturbations. Mach Learn. 2018;107(3):481–508.
4. Finlay C, Oberman A, Abbasi B. Improved robustness to adversarial examples using lipschitz regularization of the loss. 2018. arXiv:1810.00953

5. Gao B, Pavel L. On the properties of the softmax function with application in game theory and reinforcement learning. 2018. arXiv:1704.00805
6. Goodfellow I, Bengio Y, Courville A. Deep learning. London: MIT Press; 2016.
7. Gouk H, Frank E, Pfahringer B, Cree M. Regularisation of neural networks by enforcing lipschitz continuity. Mach Learn. 2018;110:393–410.
8. Harrison D, Rubinfeld D. Hedonic prices and the demand for clean air. J Environ Econ Manag. 1978;5:81–102.
9. Kingma DP, Ba J. Adam: a method for stochastic optimization. 2017. arXiv:1412.6980
10. Kolter JZ, Wong E. Provable defenses against adversarial examples via the convex outer adversarial polytope. In: ICML; 2018.
11. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86:2278–324.
12. Finlay C, Calder J. Abbasi B, Oberman A. Lipschitz regularized deep neural networks generalize and are adversarially robust. 2019. arXiv:1808.09540
13. Raghu M, Poole B, Kleinberg J, Ganguli S, Sohl-Dickstein J. On the expressive power of deep neural networks. In: Proceedings of the 34th International Conference on Machine Learning; 2017, pp. 2847–54.
14. Raghunathan A, Steinhardt J, Liang P. Certified defenses against adversarial examples. 2020. arXiv:1801.09344
15. Shaham U, Yamada Y, Negahban S. Understanding adversarial training: increasing local stability of supervised models through robust optimization. Neurocomputing. 2018;307:195–204.
16. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15:1929–58.
17. Van R, Guido D, Fred L. Python 3 Reference Manual. CreateSpace, Scotts Valley, CA 2009.
18. Virmaux A, Scaman K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In: Proceedings of the 32nd international conference on neural information processing systems. Canada, pp 3839–3848 2018.
19. Wen F, Chu L, Liu P, Qiu RC. A survey on nonconvex regularization based sparse and low-rank recovery in signal processing, statistics, and machine learning. 2019. arXiv:1808.05403
20. Xu H, Caramanis C, Mannor S. Robustness and regularization of support vector machines. J Mach Learn Res. 2009;10:1485–510.
21. Yoshida Y, Miyato T. Spectral norm regularization for improving the generalizability of deep learning. 2017. arXiv:1705.10941
22. Tsuzuku Y, Sato I, Sugiyama M. Lipschitz-Margin training: scalable certification of perturbation invariance for deep neural networks. 2018. arXiv:1802.04034
23. Zeiler MD. ADADELTA: an adaptive learning rate method. 2012. arXiv:1212.5701