ORIGINAL PAPER

Laplacian networks: bounding indicator function smoothness for neural networks robustness

CARLOS LASSANCE,[1] VINCENT GRIPON[1] AND ANTONIO ORTEGA[2]

For the past few years, deep learning (DL) robustness (i.e. the ability to maintain the same decision when inputs are subject to perturbations) has become a question of paramount importance, in particular in settings where misclassification can have dramatic consequences. To address this question, authors have proposed different approaches, such as adding regularizers or training using noisy examples. In this paper we introduce a regularizer based on the Laplacian of similarity graphs obtained from the representation of training data at each layer of the DL architecture. This regularizer penalizes large changes (across consecutive layers in the architecture) in the distance between examples of different classes, and as such enforces smooth variations of the class boundaries. We provide theoretical justification for this regularizer and demonstrate its effectiveness to improve robustness on classical supervised learning vision datasets for various types of perturbations. We also show it can be combined with existing methods to increase overall robustness.

Keywords: Machine learning, Robustness, Graph signal processing, Laplacian, Adversarial attacks

Received 16 January 2020; Revised 04 January 2021

I. INTRODUCTION

Deep learning (DL) networks provide state-of-the-art performance for many machine-learning tasks [1, 2]. Their ability to achieve good generalization is often explained by the fact they use very few priors about data [3]. On the other hand, their strong dependency on data may lead to selection of biased features of the training dataset, resulting in a lack of robustness in classification performance. Here robustness refers to the ability of a classifier to infer correctly even when the inputs (or the parameters of the classifier) are subject to perturbations. These perturbations can be due to general factors – such as noise, quantization of inputs or parameters, and adversarial attacks – as well as application specific ones – such as the use of a different camera lens, brightness exposure, or weather, in an imaging task.

An often studied case is that of adversarial attacks, where very small (imperceptible) changes on the input of a trained DL network have been shown to result in its misclassification [4, 5], thus providing evidence that DL networks may not be as robust to perturbations as the test accuracy would lead one to believe. In practice, in many applications it is unlikely that there would be adversaries, and in particular adversaries having access to the network parameters that are required to build these perturbations. This is why recent study advocates focusing on more realistic robustness [6], i.e. considering perturbations that are not linked to adversarial (or deliberate) noise [7].

Proposed approaches to improve robustness can be roughly divided into three classes: (i) changes in the inference methods, (ii) introduction of regularizers, and (iii) ad-hoc data augmentation. *Inference methods* use a procedure for prediction that differs from the one the network has been trained on (i.e. unlike the classic arg max classifier), e.g. based on a model ensemble composed of k-nearest neighbors classifiers for each layer [8]. Methods *introducing regularizers* range from minimizing the squared norm Jacobian [9] to controlling the Lipschitz constant of the network function via regularization [10, 11]. Note that bounding the Lipschitz constant of the network as proposed in these studies can be inconsistent with the training objective. We discuss this in further detail in Section II.A. Finally *data-augmentation methods* revolve around using deformed inputs during the training phase so that the network function generalizes better to the corresponding perturbations [5, 12–15]. However, there is no guarantee that training to

[1]Département Électronique, IMT Atlantique, 655 Avenue du Technopôle, Brest 29280, France
[2]Department of Electrical and Computer Engineering, Signal and Image Processing Institute, University of Southern California, 3740 McClintock Ave., EEB 436, Los Angeles, CA 90089-2564, USA

Corresponding author:
Carlos Lassance
Email: cadurosar@gmail.com

be robust to some perturbations can universally increase robustness to all types of perturbations, even if using adversarial examples during training.

Our proposed method fits in the second category: we introduce a regularizer that penalizes large deformations of the class boundaries throughout the network architecture, independently of the types of perturbations that we expect to face when the system is deployed. It also enforces a large margin r (i.e. mid-distance between examples of distinct classes) at each layer of the architecture.

To understand the intuition behind our proposed regularizer, first note that networks are typically trained with the objective of yielding zero error for the training set. If error on the training set is (approximately) zero then any two examples with different labels can be separated by the network, even if these examples are close to each other in the original domain. This means that the network function can create significant deformations of the space (small distances in the original domain map to larger distances in the final layers) and explains how an adversarial attack with small changes to the input can lead to class label changes. Our proposed regularizer penalizes big changes at the boundaries between classes. By forcing boundary deformations to evolve smoothly across the architecture, and at the same time by maintaining a large margin, the proposed regularizer therefore favors smooth variations instead, leading to better robustness. We will empirically demonstrate this claim on classical vision datasets.

The proposed regularizer is based on a series of graphs, one for each layer of the DL architecture, where each graph captures the similarity between training examples given their intermediate representation at that layer. Our regularizer favors small changes, from one layer to the next, in the distances between pairs of examples in different classes.[1] It achieves so by penalizing large changes in the smoothness (computed using the Laplacian quadratic form) of the class indicator vectors (viewed as "graph signals"). As a result, the margin is kept almost constant across layers, and the deformations of space are controlled at the boundary regions, as illustrated in Fig. 1. This draws heavily from previous studies of some of the authors [16, 17], and uses the robustness definition that was introduced in [18], which was inspired from preliminary versions of this study [19]. More generally, combining graphs and DL analysis has sparked a lot of interest recently. For example, Anirudh *et al.* [20] introduced different quantities related to graph signal processing (GSP) that can be used to extract interpretable results from DL networks, but, unlike our study, does not tackle robustness. Another example is [21] where the authors exploit graph convolutional layers [22], smoothing the latent representations of the inferred images using images from the training set, in order to increase the robustness of the network. Note that this could be described as a denoising of the inference (test) image, using the training ones. This differs from our study, which focuses on generating a smooth network function.

In Section III we demonstrate, using readily-available image classification datasets, the robustness of the proposed regularizer to common perturbations studied in the literature: (i) noise [7, 10], for which we show reductions in relative error increase, (ii) adversarial attacks [4, 5], for which the median defense radius [14] is increased by 50% in comparison with the baseline and by 12% in comparison with another method in the literature [11], and (iii) implementation defects, which result in only approximately correct computations [23], for which we increase the median accuracy by 48% relative to the baseline and 26% relative to another method in the literature [11].

The outline of the paper is as follows. In Section II we introduce the proposed regularizer and our definition of robustness. In Section III we evaluate the performance of our proposed method in various conditions and on vision benchmarks. Section IV summarizes our conclusions.

II. METHODOLOGY

A) Robustness definition

A deep neural network architecture is entirely described by its associated "network function." This network function f receives an input $\mathbf{x}$ (e.g. a tensor representing the pixel values of an image, normalized so that standard deviation is 1) and outputs a class-wise classification score $f(\mathbf{x})$. Typically this output is a vector with as many coordinates as the number of classes in the problem, where the highest valued coordinate is the decision of the network (i.e. arg max classifier). This function is constructed via the composition of multiple intermediate functions f^ℓ:

$$f = f^L \circ f^{L-1} \circ \cdots \circ f^1, \tag{1}$$

where each function f^ℓ is highly constrained, typically as the concatenation of a parameter-free nonlinear function with a parameterized linear function.

The function f is typically obtained based on a very large number of parameters, which are tuned during the learning phase. During this phase, a loss function is minimized over a set of training examples using a variant of the stochastic gradient descent algorithm. At the end of the training process, each training example is associated through f with a vector whose largest value is the actual class of that example, leading to an accuracy close to 100% on the training set. Importantly, the loss function usually targets a specific margin in the output domain. For example, when using the classical cross-entropy loss, the loss function is minimized when the output of the training examples are the one-hot-bit vectors of their corresponding class [3, 24], which corresponds to a margin in the output domain of about $\sqrt{2}/2$ for the $\mathcal{L}_2$ norm.

We define network robustness following the ideas in our previous study [18] as:

[1]Note that the distance between any two examples at a certain layer depends on their positions in the original domain and the network function applied up to that layer. Thus, constraints on the distances lead to constraints on the parameters of the network function.

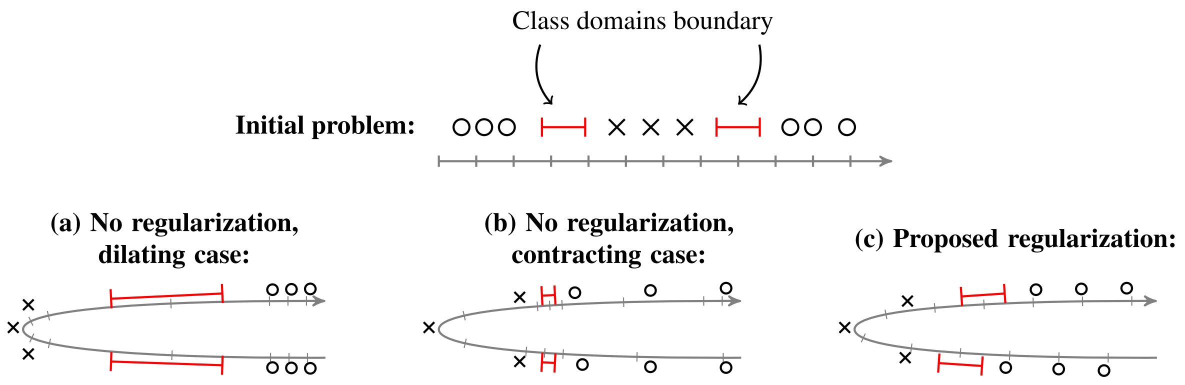
Class domains boundary
Initial problem:
(a) No regularization, dilating case:
(b) No regularization, contracting case:
(c) Proposed regularization:

Fig. 1. Illustration of the effect of our proposed regularizer. In this example, the goal is to classify circles and crosses (top). Without use of regularizers (bottom left), the resulting embedding may considerably stretch the boundary regions. Consequently, the risk is to obtain sharp transitions in the network function (that would correspond to a large value of α in equation (2)). Another possible issue would be to push inputs closer to the boundary (bottom center), thus reducing the margin (that would correspond to a small value of r in equation (2)). Forcing small variations of smoothness of label signals (bottom right), we ensure the topology is not dramatically changed in the boundary regions.

Definition 1. *A network function f is α-**robust** over a domain R and for $r > 0$, which we denote $f \in Robust_\alpha(R, r)$, if f is locally α-Lipschitz within a radius r of any point in domain R. Recall that a function f is α-Lipschitz if $\|f(x+\varepsilon) - f(x)\| \leq \alpha\|\varepsilon\|, \forall x, \forall \varepsilon$. Formally, $f \in Robust_\alpha(R, r)$ if we have:*

$$\|f(\mathbf{x}+\varepsilon) - f(\mathbf{x})\| \leq \alpha\|\varepsilon\|, \quad \forall \mathbf{x} \in R, \forall \varepsilon \quad s.t.\ \|\varepsilon\| < r\,. \tag{2}$$

Obviously, we would like to obtain a function f that is α-robust for any valid input. But since we only have access to training samples, we only enforce the property over the training set.

In our definition, robustness captures a compromise between margin (represented by r) and slope (represented by α) of the network function. This is in contrast to other studies [11] where robustness is directly linked to the Lipschitz constant of the network function. The proposed definition is less strict in the sense that f α-Lipschitz implies that $f \in Robust_\alpha(r), \forall r$, but conversely $f \in Robust_\alpha(r)$ for some r does not guarantee the function will be α-Lipschitz. The main motivation for introducing this weaker definition of robustness is that we do not want network functions to be contractive *everywhere*. Indeed, if all mappings are contractive everywhere we cannot hope to separate some samples in different classes.

Since DL functions are compositional, it is possible to achieve the robustness of Definition 1 by enforcing that property for each of its elementary entities. Denote f^ℓ an elementary function and $\mathbf{x}_i^c$ the input of f^ℓ corresponding to the i-th example of class c. Denote $\mu_r = \{(\mathbf{x}_i^c, \mathbf{x}_j^{c'}), \|\mathbf{x}_i^c - \mathbf{x}_j^{c'}\| \leq r \wedge c \neq c'\}$ the set of pairs of examples of distinct classes that are at distance at most r. Then the following necessary condition is easily derived:

$$f^\ell \in Robust_\alpha(r) \Rightarrow \begin{cases} \mu_r = \emptyset \text{ or} \\ \alpha \geq \max_{(\mathbf{x},\mathbf{x}')\in\mu_r} \dfrac{\|f^\ell(\mathbf{x}) - f^\ell(\mathbf{x}')\|}{\|\mathbf{x}-\mathbf{x}'\|} \end{cases} \tag{3}$$

Selecting a different α at each layer could prove challenging in practice. A simpler option is to target the same α at each layer of the architecture. Up to normalization at the input or at the output, we can therefore target $\alpha = 1$ at each layer, which translates into a simpler necessary condition:

Property 1. *Distances between examples of distinct classes should remain approximately constant on average throughout the network architecture.*

In what follows, we introduce regularizers that enforce this property using similarity graphs.

B) Intermediate representation graphs

First let us introduce notations and intermediate representation graphs, following the ideas in our previous study [16]. Consider a DL network architecture. Such a network is obtained by assembling layers of various types. A layer can be represented by a function $f^\ell : \mathbf{x}^\ell \mapsto \mathbf{x}^{\ell+1}$ where $\mathbf{x}^\ell$ is the intermediate representation of the input at layer ℓ. Assembling can be achieved in various ways: composition, concatenation, sums, etc. so that we obtain a global function f that associates an input tensor $\mathbf{x}^0$ to an output tensor $\mathbf{y} = f(\mathbf{x}^0)$. In practice a batch of b inputs $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}$ is processed concurrently.

Given a (meaningful) similarity measure s on tensors, we can define the similarity matrix of the intermediate representations at layer ℓ as:

$$\mathbf{M}^\ell[i,j] = s(\mathbf{x}_i^{\ell+1}, \mathbf{x}_j^{\ell+1}), \forall 1 \leq i,j \leq b, \tag{4}$$

where $\mathbf{M}^\ell[i,j]$ denotes the element at line i and column j in $\mathbf{M}^\ell$. In our experiments we mostly focus on the use of cosine similarity, which is widely used in computer vision. It is often the case that the output $\mathbf{x}^{\ell+1}$ is obtained right after using a ReLU function, that forces all its values to be nonnegative, so that all values in $\mathbf{M}^\ell$ are also nonnegative. We then use $\mathbf{M}^\ell$ to define a weighted graph $G^\ell = \langle V, \mathbf{M}^\ell \rangle$, where $V = \{1, \ldots, b\}$ is the set of vertices.

C) Smoothness of label signals

Given a weighted graph: $G^\ell = \langle V, \mathbf{M}^\ell \rangle$, the Laplacian of G^ℓ is the matrix:

$$\mathbf{L}^\ell = \mathbf{D}^\ell - \mathbf{M}^\ell = \mathbf{F}\Lambda\mathbf{F}^\top \tag{5}$$

where $\mathbf{D}^\ell$ is the diagonal degree matrix of $\mathbf{M}^\ell$. Consider a vector $\mathbf{s} \in \mathbb{R}^b$, we define $\hat{\mathbf{s}}$ the graph Fourier transform of $\mathbf{s}$ on G^ℓ as [25]:

$$\hat{\mathbf{s}} = \mathbf{F}^\top \mathbf{s}. \tag{6}$$

Assume the order of the eigenvectors ($\mathbf{F}$) is chosen so that the corresponding eigenvalues are in ascending order. If only the first few entries of $\hat{\mathbf{s}}$ are nonzero then $\mathbf{s}$ is low frequency (smooth). In the extreme case where only the first entry of $\hat{\mathbf{s}}$ is nonzero we have that $\mathbf{s}$ is constant (maximum smoothness). From this definition we derive a measure called *smoothness* $\sigma^\ell(\mathbf{s})$ of a signal $\mathbf{s}$. The smoothness is measured using the Laplacian quadratic form:

$$\sigma^\ell(\mathbf{s}) = \mathbf{s}^\top \mathbf{L}^\ell \mathbf{s} = \sum_{i,j=1}^{b} \mathbf{M}^\ell[i,j](\mathbf{s}[i] - \mathbf{s}[j])^2 \tag{7}$$

where we note that $\mathbf{s}$ is smoother when $\sigma^\ell(\mathbf{s})$ is smaller.

In this paper we are particularly interested in smoothness of the label signals. We call *label signal* $\mathbf{s}_c$ associated with class c a binary ($\{0,1\}$) vector whose nonzero coordinates are the ones corresponding to input vectors of class c. In other words, for any i, $1 \leq i \leq b$:

$$\mathbf{s}_c[i] = 1 \Leftrightarrow (\mathbf{x}_i \text{ is in class c}). \tag{8}$$

From equation (7), we can see that the smoothness of label signal $\mathbf{s}_c$ is the sum of similarities between examples in distinct classes (since $(\mathbf{s}[i] - \mathbf{s}[j])$ is zero when i and j have the same label). Thus, a total smoothness of 0 means that all examples in distinct classes have 0 similarity.

Next we introduce a regularizer that limits how much σ^ℓ can vary from one layer to the next, thus leading to a network that is more in line with Property 1. This will be shown to improve robustness in Section III.

D) Proposed regularizer

1) Definition

We propose to measure the deformation induced by a given layer ℓ by computing the difference between label signal smoothness before and after the layer for all labels:

$$\delta_\sigma^\ell = \sum_c \left|\sigma^\ell(\mathbf{s}_c) - \sigma^{\ell-1}(\mathbf{s}_c)\right|. \tag{9}$$

These quantities are used to regularize modifications made by each of the layers during the learning process. The pseudo-code of Algorithm 1 describes how we use the proposed regularizer to compute the loss.

Illustrative example:

In Fig. 1 we depicted a toy illustrative example to motivate the proposed regularizer. We consider here a one-dimensional two-class problem. To linearly separate circles and crosses, it is necessary to group all circles. Without regularization the resulting embedding is likely to either considerably increase the distance between examples in different classes (case (a)), thus producing sharp transitions in the network function, or to reduce the margin (case (b)). In contrast, by penalizing large variations of the smoothness of label signals (case (c)), the average distance between examples in different classes must be preserved in the embedding domain, resulting in a more precise control of distances within the boundary region.

Remark 1. *Since we only consider label signals, we solely depend on the similarities between examples of distinct classes. As such, the regularizer only focuses on the boundary, and does not vary if the distance between examples of the same label grows or shrinks.*

Remark 2. *Compared with [11], there are key differences that characterize the proposed regularizer:*

(i) Only pairwise distances between examples are taken into account. This has the effect of controlling space deformations only in the directions of training examples.
(ii) The network is forced to maintain a minimum margin by keeping the smoothness small at each layer of the architecture, thus controlling both contraction and dilatation of space at the boundary. This is illustrated in Fig. 1, where [11] is represented by b) and our method by c).
(iii) The proposed criterion is an average (sum) over all distances, rather than a stricter criterion (e.g. maintaining a small Lipschitz constant), which would force each pair of vectors $(\mathbf{x}_i, \mathbf{x}_j)$ to obey the constraint.

In summary, by enforcing small variations of smoothness across the layers of the network, the proposed regularizer maintains a large enough r so that equation (3) can hold, while also controlling dilatation. Combining it with Parseval [11] would allow for a better control of the α parameter in the other directions of the input space.

E) Relations to label signal bandwidth and powers of the Laplacian

Recent study [17] develops an asymptotic analysis of the bandwidth of label signals, $BW(\mathbf{s})$, where bandwidth is defined as the highest non-zero graph frequency of $\mathbf{s}$, i.e. the nonzero entry of $\hat{\mathbf{s}}$ with the highest index. An estimate of the bandwidth can be obtained by computing:

$$BW_m(\mathbf{s}) = \left(\frac{\mathbf{s}^\top \mathbf{L}^m \mathbf{s}}{\mathbf{s}^\top \mathbf{s}}\right)^{(1/m)} \tag{10}$$

for large m. This can be viewed as a generalization of the smoothness metric of (7). Anis *et al.* [17] show that, as the

Algorithm 1 Loss function of the regularized network

```
1: Inputs:
2: x: list of all the representations of the network.
3: ReLUs, the list containing the positions of all the ReLU activations on $f$.
4: y, the output of the network
5: s, the label signal of the batch, i.e., the ground truth labels of the examples of the batch
6: $m$, the power of the Laplacian for which we wish to compute the smoothness;
7: $\gamma$, the scaling coefficient of the regularizer loss.
8:
9: procedure Loss($\mathbf{x}$, ReLUs, $\mathbf{y}$, $\mathbf{s}$, $m$, $\gamma$)
10:     for $\ell \in$ ReLUs do
11:         $\sigma^\ell \leftarrow$ Smoothness($\mathbf{x}^\ell$, $\mathbf{s}$, $m$)
12:     end for
13:     $\Delta \leftarrow \frac{\sum_{\ell \in \text{ReLUs}} \delta_\sigma^\ell}{||\text{ReLUs}||-1}$ (cf. equation (9))
14:     return CategoricalCrossEntropy($\mathbf{s}$, $\mathbf{y}$) $+ \gamma\Delta$
15: end procedure
16: procedure SMOOTHNESS($\mathbf{x}^\ell$, $\mathbf{s}$, $m$)
17:     $\mathbf{M}^\ell \leftarrow$ Pairwise similarity of $\mathbf{x}^\ell$ (we use cosine similarity in our study)
18:     $\mathbf{D}^\ell \leftarrow$ Diagonal degree matrix of $\mathbf{M}^\ell$
19:     $\mathbf{L}^\ell \leftarrow \mathbf{D}^\ell - \mathbf{M}^\ell$
20:     $\sigma^\ell \leftarrow$ Trace($\mathbf{s}^\top (L^\ell)^m \mathbf{s}$)
21:     return $\sigma^\ell$
22: end procedure
```

number of labeled points $\mathbf{x}$ (assumed drawn from a distribution $p(\mathbf{x})$) grows asymptotically, the bandwidth of the label signal converges in probability to the supremum of $p(\mathbf{x})$ in the region of overlap between classes. This motivates our study in three ways.

First, it provides theoretical justification to use $\sigma^\ell(\mathbf{s})$ for regularization, since lower values of $\sigma^\ell(\mathbf{s})$ are indicative of better separation between classes. Second, the asymptotic analysis suggests that using higher powers of the Laplacian would lead to better regularization, since estimating bandwidth using $BW_m(\mathbf{s})$ becomes increasingly accurate as m increases. Finally, our regularization can be seen to be protective against specializing by preventing $\sigma^\ell(\mathbf{s})$ from decreasing "too fast." For most problems of interest, given a sufficiently large amount of labeled data available, it would be reasonable to expect the bandwidth of $\mathbf{s}$ not to be arbitrarily small, because the classes cannot be exactly separated, and thus a network that reduces the bandwidth too much could be unreliable (i.e. biased by the training set). After trying multiple values of m, we found that using $m = 2$ usually led to close to the best performance in our experiments.

III. EXPERIMENTS

In the following subsections, we evaluate the proposed method using various tests. We use the well-known CIFAR-10 dataset [26] as a first benchmark and we demonstrate that our proposed regularizer can improve robustness as defined in Section II.A.

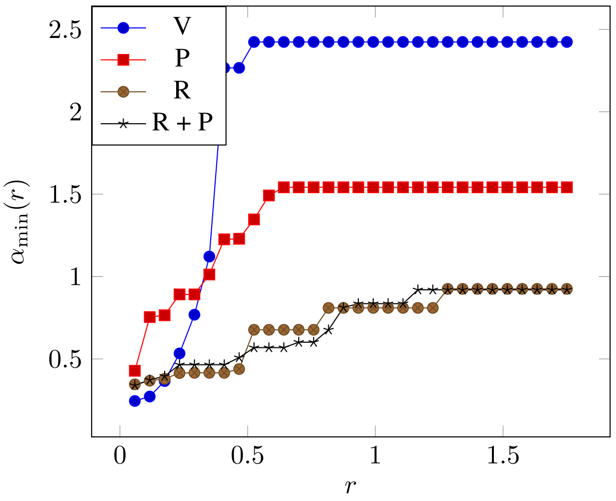

Fig. 2. Estimations of $\alpha_{\min}(r)$ obtained for different radii r over training examples. The proposed regularizer allows for smaller α values when r increases.

In summary, in subsection A) we first verify that the proposed regularizer favors Definition 1. We then show in subsection B) that by using the proposed regularizer we are able to increase robustness for random perturbations and weak adversarial attacks. In subsection C), we challenge our method on more competitive benchmarks. Finally, in subsection D) we extend the analysis to CIFAR-100 [26] and Imagenet32x32 [27] to validate the generality of the method. These experiments demonstrate that DNNs trained with the proposed regularizer lead to improved robustness.

To measure accuracy, we average over 10 runs each time, unless mentioned otherwise. In all reports, P stands for Parseval [11] trained networks, R for networks trained with the proposed regularizer, and V for Vanilla (i.e. baseline) networks. Note that the network architecture is the same for all networks and that Vanilla refers to the network architecture trained without any regularizer. The network architecture is depicted in Fig. A1. Training hyperparameters and details of our implementation of the Parseval method can be found in the Appendix. The corresponding code is available at https://github.com/cadurosar/laplacian_networks.

A) Robustness of trained architectures

First we verify that the proposed regularizer improves robustness as defined in Section II.A. For various values of r, we estimate $\alpha_{\min}(r) = \arg\min_\alpha \{f \in Robust_\alpha(r)\}$. We use 1000 training examples and generate 100 uniform noises to estimate $\alpha_{\min}(\cdot)$. Results are shown in Fig. 2. We observe that networks trained with the proposed regularizer allow for smaller α values when the radius r increases. The Parseval method achieves better Lipschitz constant than Vanilla, as suggested by the large values of r. However, we observe that $\alpha_{\min}$ grows fast when using Parseval, suggesting that sharp transitions are allowed in the vicinity of trained examples.

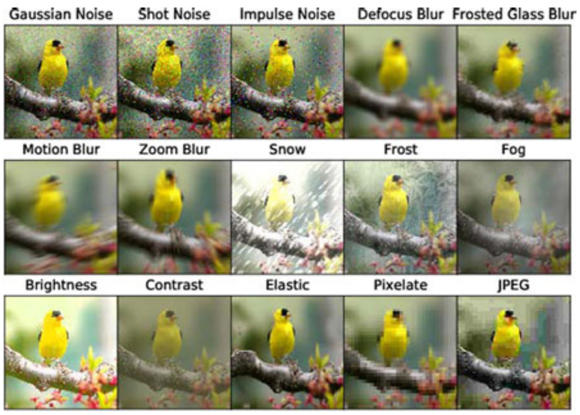

Fig. 3. Illustration of the 15 perturbations from [7]. Best viewed in color.

B) Experiments on perturbations and adversarial attacks

In this subsection we verify the ability of the proposed regularizer to increase robustness, while retaining acceptable accuracy on the clean test set, on the CIFAR-10 dataset without any type of data augmentation.

1) Clean test set

Before checking the robustness of the network, we first test the performance on clean examples. In the second column of Table 1, we show the baseline accuracy of the models on the clean CIFAR-10 test set (no perturbation is added at this point). These experiments agree with the claim from [11] where the authors show that they are able to increase the performance of the network on the clean test set. We observe that the proposed method leads to a minor decrease of performance on this test. However, we see in the following experiments that this is compensated by an increased robustness to perturbations. The fact that it is difficult to avoid such a trade-off between robustness and accuracy has already been discussed in the literature [28].

2) Perturbation robustness

In order to assess the effectiveness of the various methods when subject to perturbations, we use the benchmark proposed in [7] which consists of 15 different perturbations, with five levels of severity each (note that they are referred to as "corruptions" in [7]). Perturbations test the robustness of the network to noise when compared to its clean test set performance. We illustrate each perturbation in Fig. 3.

In more detail, we are interested in the mean relative error inflation (mREI). To define it, consider $E_{\texttt{net}}^{\texttt{per,sev}}$ the error rate of a network net (V,P,R or P+R), under perturbation type per and severity sev. Denote $E_{\texttt{net}}$ the error rate of the network net on the clean set. We first define Error Inflation (EI) as:

$$EI_{\texttt{net}}^{\texttt{per,sev}} = \frac{E_{\texttt{net}}^{\texttt{per,sev}}}{E_{\texttt{net}}}.$$

Then the REI is defined as:

$$REI_{\texttt{net}}^{\texttt{per,sev}} = EI_{\texttt{net}}^{\texttt{per,sev}} - EI_{\texttt{V}}^{\texttt{per,sev}}.$$

Table 1. Network mREI under different types of perturbations

			Noise			Blur				Weather				Digital			
Network	Clean set	**mREI**	Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
Vanilla (V)	11.9%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Parseval (P)	10.3%	0.29	0.71	0.48	0.57	0.10	1.01	0.15	0.11	0.16	0.17	−0.02	0.04	0.13	0.14	0.25	0.28
Regularizer (R)	13.2%	−0.29	−1.12	−0.86	0.10	−0.03	−0.65	−0.09	−0.19	−0.30	−0.61	0.17	0.01	0.40	−0.15	−0.50	−0.50
P+R	12.8%	−0.35	−1.33	−1.00	0.05	−0.09	−0.75	−0.18	−0.31	−0.41	−0.67	0.13	0.04	0.48	−0.17	−0.47	−0.53
	mCD 10^{-3}		18	16	37	5	24	15	17	15	20	51	14	57	14	6	3

Bottom line represents the corresponding median Cosine Distance (mCD) (at the highest perturbation severity) between corrupted and clean images.

Finally, mREI is obtained by averaging over all severities.

The results are described in Table 1 for the CIFAR-10 dataset. The raw error rates under each type of perturbation can be found in the Appendix. We observe that Parseval alone is not able to help with the mREI, despite reducing the clean set error. On the other hand, the proposed regularizer and its combination with Parseval training decreases the clean set accuracy but increases the relative performance under perturbations by a significant amount.

This experiment supports the fact that the proposed regularizer can significantly improve robustness to most types of perturbations introduced in [7]. It is worth pointing out that this finding does not hold for Impulse Noise, Fog, and Contrast. Looking more into detail, we observe that Impulse noise shifts some values on the image to either its maximum possible value or the minimum possible value, while Fog and Contrast perform a re-normalization of the image. In those cases perturbations have the effect of creating noisy inputs that are far away (in terms of the cosine distance) from the original images, as supported by the last line of the table. This is in contrast to the other types of perturbations in the experiment. Because they can be far away, these perturbations do not fulfill the definition of equation (2), where there is a maximum radius r for which robustness is enforced around the examples. In other words, our robustness definition is focusing on small deviations/distances as those are more likely to characterize noise (i.e we focus on distances that are too small to change the class of the image).

3) Adversarial robustness

We next evaluate robustness to adversarial inputs, which are specifically built to fool the network function. Such adversarial inputs can be generated and evaluated in multiple ways. Here we implement three approaches: first a mean case of adversarial noise, where the adversary can only use one forward and one backward pass to generate the perturbations, second a worst case scenario, where the adversary can use multiple forward and backward passes to try to find the smallest perturbation that will fool the network, third a compromise between the mean case and the worst case, where the adversary can do a predefined number of forward and backward passes with a perturbation threshold limit.

For the first approach, we add the scaled gradient sign (called the fast gradient signal method or FGSM) to the input [13], so that we obtain a target signal-to-noise ratio (SNR) of 33. This is in line with previous studies [11]. Obtained results are introduced in the left and center plots of Fig. 4. In the left plot the noise is added after normalizing the input, whereas on the middle plot it is added before normalizing it. As with the perturbation tests, a combination of the Parseval method and our proposed approach yields the most robust architecture.

With regard to the second approach, where a worst case scenario is considered, we use the Foolbox [29] implementation of DeepFool [14]. Due to time constraints we sample only 1/10 of the test set images for this test. The conclusions we can draw are similar (right plot of Fig. 4) to those obtained for the first adversarial attack approach. Finally,

Table 2. Median test set accuracy on the CIFAR-10 dataset against the PGD attack

Model	PGD accuracy (%)
V	1.18
P	1.72
R	5.2
P+R	**5.6**

Table 3. Comparison of CIFAR-10 test set accuracy under the black box FGSM attack

	Source			
Target	V	P	R	P+R
V	X	*60.74*	61.49	72.51
P	*57.82*	X	68.21	**73.87**
R	*69.72*	74.96	X	73.56
P+R	**75.35**	**76.11**	***70.22***	X

The most robust target for a given source is bolded, while the strongest source for a target is in italic.

for the third approach we use the PGD (projected gradient descent) attack introduced in [15]. PGD is an iterative version of FGSM, which loops for a maximum number of *it* iterations. For each iteration it moves by a distance of *step* in the direction of the gradient, provided it does not move away from the original image by a distance greater than ϵ. Our experiments (described in Table 2) show that the proposed regularizer increases robustness against a PGD attack, for an epsilon corresponding to an SNR of about 33 ($it = 20, step = 0.002, \epsilon = 0.01$).

A common pitfall in evaluating robustness to adversarial attacks comes from the fact the gradient of the architecture can be masked due to the introduced method. As a consequence, generated attacks become weaker compared to those on the vanilla architecture. So, to further verify that the obtained results are not only due to gradient masking, we perform tests with black box FGSM, where the target attacked network is not the same as the source of the adversarial noise. This way, all networks are tested against the same attacks.

For this test we continue to use an SNR of about 33 with the FGSM method. We choose the network with the best performance for each of the tested methods. The results are depicted in Table 3. In our experiments, we found that the combination of our method with Parseval is the most robust to noise coming from other sources. This demonstrates that the improvements are not caused by gradient masking, but are caused by the increased robustness of the proposed method and Parseval's. Interestingly, the noise created by both Parseval and our method did not challenge the other methods as well as the one created by Vanilla, justifying *a posteriori* the interest of this experiment.

4) Robustness to parameter and activation noises

In a third series of experiments we aim at evaluating the robustness of the architecture to noise on parameters and activations. We consider two types of noises: (i) erasures of the memory (dropout), and (ii) quantization of the weights [23].

In the dropout case, we compute the test set accuracy when the network has a probability of either 25 or 40% of dropping an intermediate representation value after each block of computation in the architecture. We average over a run of 40 experiments. Results are depicted in the left and center plots of Fig. 5. It is interesting to note that the Parseval trained functions seem to collapse as soon as we reach 40% probability of dropout, providing an average accuracy smaller than the vanilla networks. In contrast, the proposed method is the most robust to these perturbations.

For the quantization of the weights, we aim at compressing the network size in memory by a factor of 6. We therefore quantize the weights using 5 bits (instead of 32) and re-evaluate the test set accuracy. The right plot of Fig. 5 shows that the proposed method is providing a better robustness to this perturbation than the tested counterparts.

Overall, these experiments confirm previous ones in the conclusion that the proposed regularizer obtains the best robustness compared to Parseval and Vanilla networks. Note that in this case (parameter and activation noise) there is a drop of performance when combining the proposed regularizer with the Parseval method. Although we do not have direct theoretical justification for this, the fact that the proposed regularizer and Parseval present conflicting strategies (smoothness versus contraction) could explain the drop in accuracy when the two methods are combined.

C) Experiments on challenging benchmarks

In this subsection we verify the ability of the proposed regularizer to increase robustness on the CIFAR-10 dataset while being combined with recent techniques of adversarial data augmentation. This is important as those methods are seen as the state of the art for adversarial robustness. Adversarial data augmentation consists of augmenting the training set during the training stage by using the same kind of attacks as those described in the last subsection. We refer to techniques using adversarial data augmentation using the letter A.

1) Tests with FGSM adversarial data augmentation

We first perform experiments with adversarial data augmentation as suggested in [13]. To be more precise we use the method they advise which is called "step1.1" using $\epsilon = 8/255$.

A first test consists of measuring the accuracy when inputs are modified with additive Gaussian noise with various SNRs. As expected, we observe in Fig. 6 that training with adversarial examples helps in this case, as it adds more variation to the training set. Yet it reduces the accuracy on the clean set (left plot). Note that combining our method with adversarial training results in the best median accuracy.

Table 4. Test set accuracy results on the CIFAR-10 dataset with PGD training

	Clean (%)	Gaussian (%)	PGD (%)	Dropout (%)
A	76.39	71.25	32.78	35.20
A + R	76.36	**72.26**	**33.72**	**55.63**

About robustness to adversarial attacks, the obtained results are depicted in Fig. 7. We observe that adding FGSM adversarial training does not generalize well to other types of attacks (which is readily seen in the literature [15]). Overall, the models using the proposed regularizer are the most robust again.

Finally, when considering implementation related perturbations, the results depicted in Fig. 8 are consistent with the ones from the previous section, in which is shown that the proposed regularizer helps improving robustness to this type of noise.

In summary, even when adding adversarial training, the proposed regularizer is either the most robust in median, or capable of improving the robustness when combined with the other methods.

2) Tests with PGD adversarial data augmentation

Most of our adversarial tests are performed with FGSM because of its simplicity and speed, even though it has already been shown (e.g. [15]) that FGSM is weak as an attack and as a defense mechanism. Despite the fact we do not only target adversarial defense, we further stress the ability of the proposed regularizer to improve it and to combine with other methods. To this end we perform experiments against the PGD attack.

As the proposed regularizer can be combined with FGSM defense, it is natural to also test it alongside PGD training. We use the parameters advised in [15]: seven iterations with $step = 2/255$, and $\epsilon = 8/255$. The results depicted in Table 4 show that using our regularizer increases robustness of networks trained with PGD. Note that Dropout and Gaussian noise were applied 10 times to each of the networks and the results are displayed as the mean test set accuracy under these perturbations. A rate of 40% was used for dropout. The PGD attack uses the following parameters: $it = 20, step = 2/255, \epsilon = 8/255$.

D) Experiments with other datasets

In this final subsection, we test the generality of the method using the CIFAR-100 and ImageNet32x32 datasets, with a subset of the perturbations used for CIFAR-10. Gaussian noise is applied 10 times to each of the networks for a total of 30 different runs. An SNR of 33 is used for FGSM and 15 for Gaussian noise. Images are normalized in the same way as the experiments with CIFAR-10. Standard data augmentation is used for CIFAR-100.

Results on CIFAR-100 are shown in Table 5 as the mean over three different initializations. We observe that as it was

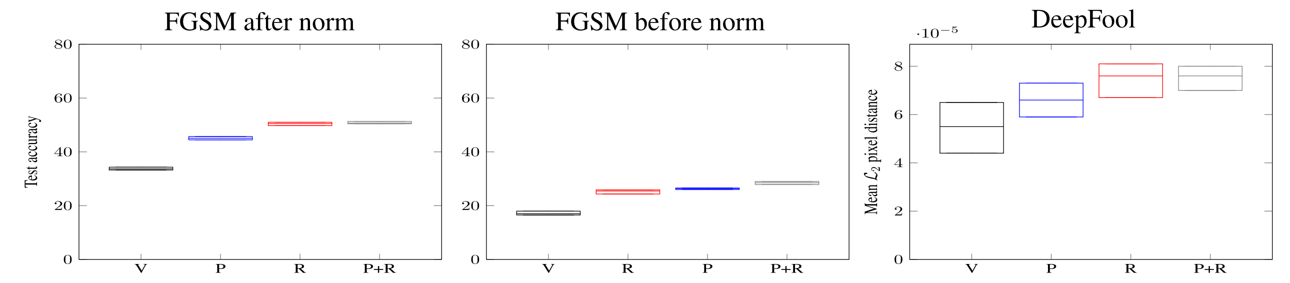

Fig. 4. Robustness against an adversary measured by the test set accuracy under FGSM attack in the left and center plots and by the mean $\mathcal{L}_2$ pixel distance needed to fool the network using DeepFool on the right plot.

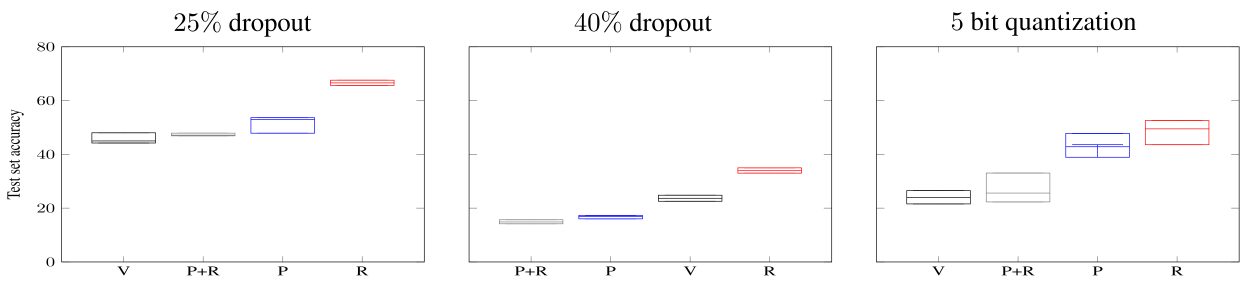

Fig. 5. CIFAR-10 test set accuracy under different types of implementation related noise.

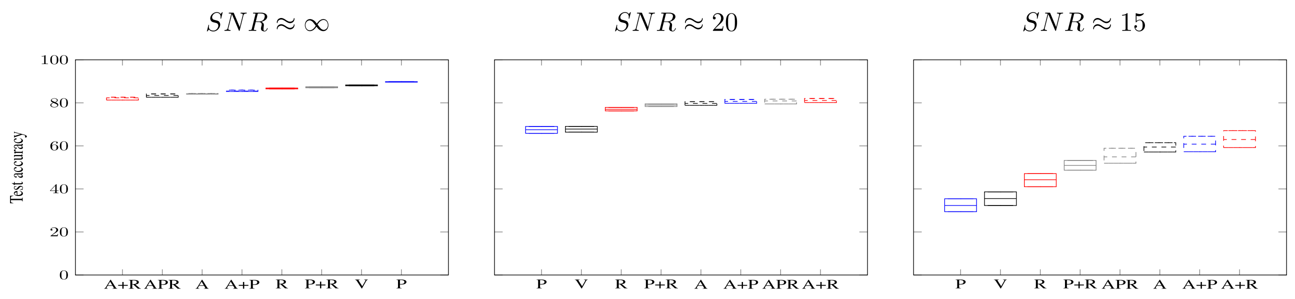

Fig. 6. Test set accuracy under Gaussian noise with varying SNRs.

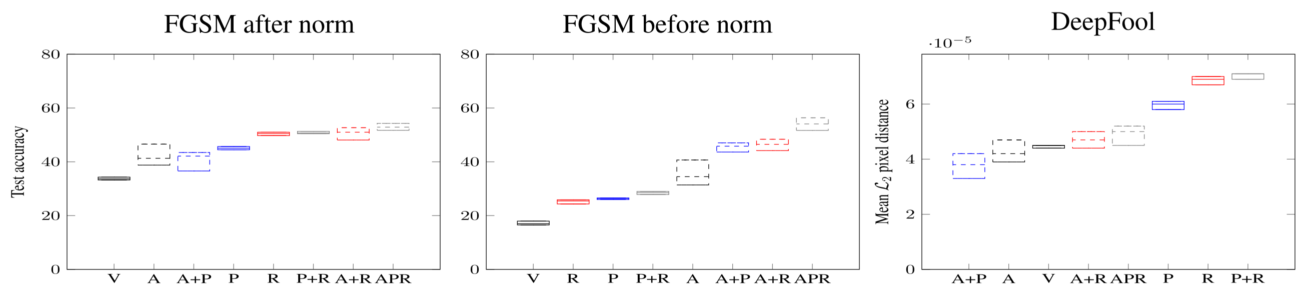

Fig. 7. Robustness against an adversary measured by the test set accuracy under FGSM attack in the left and center plots and by the mean $\mathcal{L}_2$ pixel distance needed to fool the network using DeepFool on the right plot.

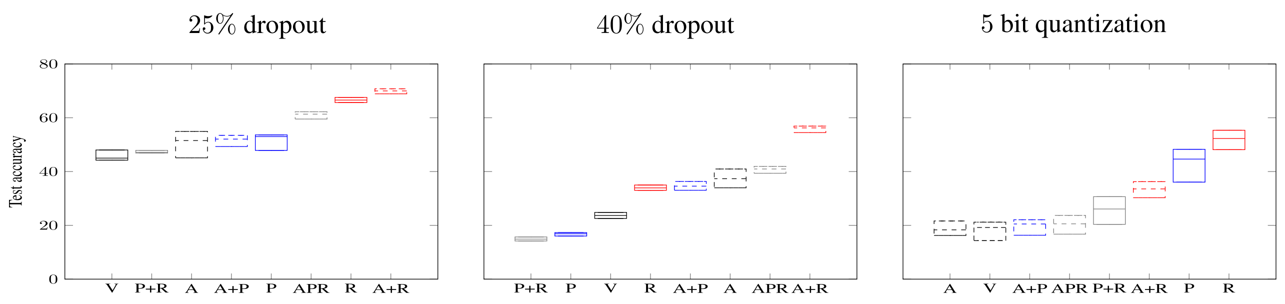

Fig. 8. Test set accuracy under different types of implementation related noise.

Table 5. Test set accuracy results on the CIFAR-100 dataset

Model	Clean set (%)	Gaussian noise (%)	FGSM (%)
Vanilla (V)	78.7	12.6	20.5
Parseval (P)	**80.1**	14.8	22.0
Regularizer (R)	79.4	15.9	23.0
P+R	79.5	**19.1**	**24.4**

Table 6. Test set accuracy results on the Imagenet32x32 dataset

Model	Clean (%)	Gaussian noise (%)	Dropout (%)
Vanilla (V)	52.1	36.8	2.3
Parseval (P)	48.1	34.10	3.71
Regularizer (R)	**52.4**	**37.4**	**7.0**
P+R	43.80	29.87	5.0

the case on CIFAR-10, the proposed method and the combination of the methods is the most robust on these test cases.

We then use Imagenet32x32, a downscaled version of Imagenet [30] which can be used as an alternative to CIFAR-10 while maintaining a similar computational budget [27]. We use the same network and training hyperparameters of the original paper. Further description can be found in the Appendix. Gaussian noise and Dropout are applied 40 times to each of the networks. Gaussian noise is applied with $SNR = 33$ whereas Dropout is applied with 15%.

Results are shown in Table 6. We observe that as it was the case on CIFAR-10 and CIFAR-100, the proposed method provides more robustness in all of these test cases. Note that we had trouble fine-tuning the β parameter for the Parseval criterion, explaining the poor performance of Parseval and its combination with our proposed regularizer.

IV. CONCLUSION

In this paper we have introduced a definition of robustness alongside an associated regularizer. The former takes into account both small variations around the training set examples and the margin. The latter enforces small variations of the smoothness of label signals on similarity graphs obtained at intermediate layers of a DL network architecture. We have empirically shown with our tests that the proposed regularizer can lead to improved robustness in various conditions compared to existing counterparts. We also demonstrated that combining the proposed regularizer with existing methods can result in even better robustness for some conditions.

Future study includes a more systematic study of the effectiveness of the method with regards to other datasets, models, and perturbations. Recent studies have shown that adversarial noise is partially transferable between models and dataset [31, 32] and therefore we are confident about the generality of the method in terms of models and datasets.

FINANCIAL SUPPORT

This study was funded in part with the support of Région Bretagne and computations were performed with the use of Nvidia GPUs, courtesy of Nvidia.

CONFLICT OF INTEREST

None.

REFERENCES

1 He, K.; Zhang, X.; Ren, S.; Sun, J.: Identity mappings in deep residual networks, in *European Conference on Computer Vision*, Springer, 2016, 630–645.

2 Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; K. Macherey, *et al.*: Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

3 LeCun, Y.; Bengio, Y.; Hinton, G.: Deep learning. *Nature*, **521** (7553) (2015), 436.

4 Szegedy, C. *et al.*.: Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

5 Goodfellow, I.J.; Shlens, J.; Szegedy, C.: Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

6 Gilmer, J.; Adams, R.P.; Goodfellow, I.; Andersen, D.; Dahl, G.E.: Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

7 Hendrycks, D.; Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

8 Papernot, N.; McDaniel, P.D.: Deep *k*-nearest neighbors: Towards confident, interpretable and robust deep learning. *CoRR*, vol. abs/1803.04765, 2018.

9 Gu, S.; Rigazio, L.: Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

10 Mallat, S.: Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, **374** (2065) (2016), 20150203.

11 Cisse, M.; Bojanowski, P.; Grave, E.; Dauphin, Y.; Usunier, N.: Parseval networks: improving robustness to adversarial examples, in *International Conference on Machine Learning*, 2017, 854–863.

12 Pezeshki, M.; Fan, L.; Brakel, P.; Courville, A.; Bengio, Y.: Deconstructing the ladder network architecture, in *International Conference on Machine Learning*, 2016, 2368–2376.

13 Kurakin, A.; Goodfellow, I.; Bengio, S.: Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

14 Moosavi Dezfooli, S.M.; Fawzi, A.; Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks, in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

15 Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A.: Towards deep learning models resistant to adversarial attacks, in *International Conference on Learning Representations*, 2018.

16 Gripon, V.; Ortega, A.; Girault, B.: An inside look at deep neural networks using graph signal processing, in *Proceedings of ITA*, February 2018.

17 Anis, A.; ElGamal, A.; Avestimehr, S.; Ortega, A.: A sampling theory perspective of graph-based semi-supervised learning. *IEEE*

Transactions on Information Theory, **65** (4) (2019), 2322–2342. doi:10.1109/TIT.2018.2879897.

18 Lassance, C.; Gripon, V.; Tang, J.; Ortega, A.: Structural robustness for deep learning architectures, in *2019 IEEE Data Science Workshop (DSW)*, June 2019, 125–129.

19 Lassance, C.E.R.K.; Gripon, V.; Ortega, A.: Laplacian power networks: Bounding indicator function smoothness for adversarial defense'. *CoRR*, vol. abs/1805.10133, 2018.

20 Anirudh, R.; Thiagarajan, J.J.; Sridhar, R.; Bremer, T.: Influential sample selection: A graph signal processing approach'. *arXiv preprint arXiv:1711.05407*, 2017.

21 Svoboda, J.; Masci, J.; Monti, F.; Bronstein, M.M.; Guibas, L.: Peernets: Exploiting peer wisdom against adversarial attacks'. *arXiv preprint arXiv:1806.00088*, 2018.

22 Bronstein, M.M.; Bruna, J.; LeCun, Y.; Szlam, A.; Vandergheynst, P.: Geometric deep learning going beyond Euclidean data. *IEEE Signal Process. Mag.*, **34** (4) (2017), 18–42.

23 Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; Bengio, Y.: Quantized neural networks Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, **18** (2017), 187–1.

24 Goodfellow, I.; Bengio, Y.; Courville, A.: Deep learning, **1**, MIT Press, Massachusetts, USA, 2017.

25 Shuman, D.I.; Narang, S.K.; Frossard, P.; Ortega, A.; Vandergheynst, P.: The emerging field of signal processing on graphs. Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, **30** (3) (2013), 83–98.

26 Krizhevsky, A.; Hinton, G.: Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf, 2009.

27 Chrabaszcz, P.; Loshchilov, I.; Hutter, F.: A downsampled variant of imagenet as an alternative to the CIFAR datasets. *arXiv preprint arXiv:1707.08819*, 2017.

28 Fawzi, A.; Fawzi, O.; Frossard, P.: Analysis of classifiers' robustness to adversarial perturbations. *Mach. Learn.*, **107** (3) (2018), 481–508.

29 Rauber, J.; Brendel, W.; Bethge, M.: Foolbox: a python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

30 Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L.: ImageNet: a Large-Scale Hierarchical Image Database, in *CVPR09*, 2009.

31 Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; Frossard, P.: Universal adversarial perturbations. *arXiv preprint*, 2017.

32 Papernot, N.; McDaniel, P.; Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

33 Kovačević, J.; Chebira, A.: An introduction to frames. *Found. Trends Signal Process.*, **2** (1) (2008), 1–94.

34 Zagoruyko, S.; Komodakis, N.: Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

APPENDIX

PARSEVAL TRAINING

We compare our results with those obtained using the method described in [11]. There are three modifications to the normal training procedure: orthogonality constraint, convolutional renormalization, and convexity constraint.

For the orthogonality constraint we enforce *Parseval tightness* [33] as a layer-wise regularizer:

$$R_\beta(W^\ell) = \frac{\beta}{2}\|W^{\ell\top}W^\ell - I\|_2^2, \tag{A.1}$$

where W_ℓ is the weight tensor at layer ℓ. This function can be approximately optimized with gradient descent by doing the operation:

$$W^\ell \leftarrow (1+\beta)W^\ell - \beta W^\ell W^{\ell\top} W^\ell. \tag{A.2}$$

Given that our network is smaller we can apply the optimization to the entirety of the W, instead of 30% as per the original paper, this increases the strength of the Parseval tightness.

For the convolutional renormalization, each matrix W^ℓ is reparameterized before being applied to the convolution as $W^\ell/\sqrt{2k_l k_w + 1}$, where k_w is the kernel width and k_h is the kernel height.

For our architecture the inputs from a layer come from either one or two different layers. In the case where the inputs come from only one layer, α the convexity constraint parameter is set to 1. When the inputs come from the sum of two layers we use $\alpha = 0.5$ as the value for both of them, which constraints our Lipschitz constant, this is softer than the convexity constraint from the original paper.

HYPERPARAMETERS

We train our networks using classical stochastic gradient descent with momentum (0.9), with batch size of $b = 100$ images and using a L2-norm weight decay with a coefficient of $\lambda = 0.0005$.

We use the mean of the difference of smoothness between successive layers in our loss function. Therefore in our loss function we have:

$$\mathcal{L} = \text{CE} + \lambda\text{WD} + \gamma\Delta \tag{A.3}$$

where $\Delta = (1/(d-1))\sum_{\ell=1}^{d}|\delta_\sigma^\ell|$, CE is the cross entropy function and WD is weight decay.

CIFAR-10

For the CIFAR-10 dataset we do a 100 epoch training. Our learning rate starts at 0.1. After half of the training (50 epochs) the learning rate decreases to 0.001. We do not add data augmentation, as it can interfere with some of the perturbations (elastic transformations for example).

We tested multiple parameters of β, the Parseval tightness parameter, γ the weight for the smoothness difference cost, and m the power of the Laplacian. We found that the best values for this specific architecture, dataset, and training scheme were: $\beta = 0.01, \gamma = 0.0001, m = 2$. The PreActResNet18 network depicted in Fig. A1 is used in all experiments for CIFAR-10.

CIFAR-100

We perform experiments using the WideResNet28-10 [34] architecture, and we added standard data augmentation (random crops and random horizontal flipping) and dropout with probability of 30% after the first convolution of each residual

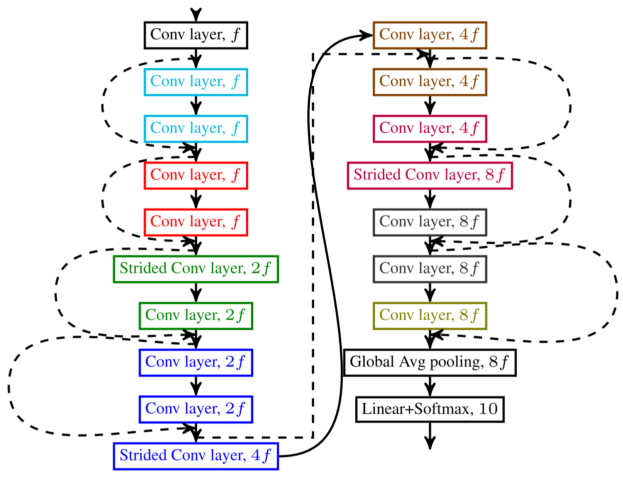

Fig. A1. Depiction of the studied network.

block. We train for 200 epochs, starting with a learning rate of 0.1 and divide the learning rate by 5 in epochs 60, 120, and 160. Momentum of 0.9 is used and weight decay of 5×10^{-4}.

We tested multiple parameters of β, the Parseval tightness parameter, γ the weight for the smoothness difference cost, and m the power of the Laplacian. We found that the best values for this specific architecture, dataset and training scheme were: $\beta = 0.0003, \gamma = 0.01, m = 2$. FGSM is applied after normalization of the input.

IMAGENET32x32

We perform experiments using the WideResNet28-5 [34] architecture, and we added standard data augmentation (random crops and random horizontal flipping). We follow the training procedure from [27], where training is done for 30 epochs, starting with a learning rate of 0.1 and divide the learning rate by 5 in epochs 10 and 20. Momentum of 0.9 is used and weight decay of 5×10^{-4}.

We tested multiple parameters of β, the Parseval tightness parameter, γ the weight for the smoothness difference cost, and m the power of the Laplacian. We found that the best values for this specific architecture, dataset and training scheme were: $\beta = 10^{-6}, \gamma = 0.01, m = 2$.

DEPICTION OF THE NETWORK

Figure A1 depicts the network we call PreActResNet18, $f = 64$ is the filter size of the first layer of the network. Conv layers are 3×3 layers and are always preceded by batch normalization and ReLU (except for the first layer which receives just the input). The smoothness is calculated after each ReLU.

PERTURBATION ROBUSTNESS

Table A1 contains the mean test error under each type of perturbation.

Table A1. Networks error under different types of perturbations

			Noise			Blur				Weather				Digital			
Network	Error	Mean Error	Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
V	11.9%	32	48	39	41	24	56	30	30	29	32	20	13	31	25	31	25
P	10.3%	31	50	40	42	22	59	28	27	27	30	17	12	30	23	30	24
R	13.2%	31	39	32	47	26	54	32	31	28	26	25	15	41	25	26	21
P+R	12.8%	29	35	29	45	24	51	30	28	25	25	23	15	41	24	26	20

Table A2. Comparison of the total time that it takes to train each method

Network	Time per epoch (s)	Time to train the network (min)	Normalized time per epoch
V	47.4	79	1.00
P	55.3	92	1.17
R	81	135	1.71
A	95	158	2.00
PGD	369	615	7.78

TIME REQUIRED TO TRAIN THE NETWORKS

The proposed regularizer requires the computation of a similarity matrix and therefore increases the overall training time. The same happens for the other methods that we compare to in this study (Parseval, FGSM, and PGD trainings).

To better understand the impact of each considered method on the training time, we report in Table A2 the training times. We note that the proposed regularizer has a greater computation cost than the Parseval method, but it is significantly faster than the adversarial training methods. Overall, the added training time of the proposed regularizer remains very limited.

Carlos Lassance received an engineering double-degree from PUC-Rio and IMT Atlantique in 2017, and he received his Ph.D. degree in machine learning and graphs from IMT Atlantique in 2020. He was recently an intern at Mila (2018–19) and started reviewing for IEEE journals. His main research interests are GSP and deep neural networks.

Vincent Gripon is a permanent researcher with IMT Atlantique, a French top technical university. He obtained his M.S. from École Normale Supérieure Paris-Saclay in 2008 and his Ph.D. from IMT Atlantique in 2011. He spent 1 year as a visiting scientist at McGill University between 2011 and 2012 and 1 year as an invited Professor at Mila and Université de Montréal. His research mainly focuses on efficient implementation of artificial neural networks, graph signal processing, deep learning robustness and associative memories. He has co-authored more than 70 papers in these domains in prestigious venues.

Antonio Ortega received his undergraduate and doctoral degrees from the Universidad Politécnica de Madrid, Madrid, Spain and Columbia University, New York, NY, respectively. In 1994 he joined the Electrical and Computer Engineering department at the University of Southern California (USC), where he is currently a Professor and has served as Associate Chair. He is a Fellow of the IEEE and EURASIP, and a member of ACM and APSIPA. He was the inaugural Editor-in-Chief (EiC) of the APSIPA Transactions is the EiC of the IEEE Transactions of Signal and Information Processing over Networks. He recently served as a member of the Board of Governors of the IEEE Signal Processing Society. He has received several paper awards, including the 2016 Signal Processing Magazine award. His recent research work is focusing on graph signal processing, machine learning, multimedia compression, and wireless sensor networks. Over 40 Ph.D. students have completed their thesis under his supervision and his work has led to over 400 publications in international conferences and journals, as well as several patents.