Improving DNN Robustness to Adversarial Attacks using Jacobian Regularization

Daniel Jakubovitz[0000−0001−7368−2370] and Raja Giryes[0000−0002−2830−0297]

School of Electrical Engineering,
Tel Aviv University, Israel
danielshaij@mail.tau.ac.il, raja@tauex.tau.ac.il

Abstract. Deep neural networks have lately shown tremendous performance in various applications including vision and speech processing tasks. However, alongside their ability to perform these tasks with such high accuracy, it has been shown that they are highly susceptible to *adversarial attacks*: a small change in the input would cause the network to err with high confidence. This phenomenon exposes an inherent fault in these networks and their ability to generalize well. For this reason, providing robustness to adversarial attacks is an important challenge in networks training, which has led to extensive research. In this work, we suggest a theoretically inspired novel approach to improve the networks' robustness. Our method applies regularization using the Frobenius norm of the Jacobian of the network, which is applied as post-processing, after regular training has finished. We demonstrate empirically that it leads to enhanced robustness results with a minimal change in the original network's accuracy.

Keywords: Deep Learning · Neural Networks · Adversarial Examples · Data Perturbation · Jacobian Regularization · Classification Robustness.

1 Introduction

Deep neural networks (DNNs) are a widespread machine learning technique, which has shown state-of-the-art performance in many domains such as natural language processing, computer vision and speech processing [7]. Alongside their outstanding performance, deep neural networks have recently been shown to be vulnerable to a specific kind of attacks, most commonly referred to as *Adversarial Attacks*. These cause significant failures in the networks' performance by performing just minor changes in the input data that are barely noticeable by a human observer and are not expected to change the prediction [8]. These attacks pose a possible obstacle for mass deployment of systems relying on deep learning in sensitive fields such as security or autonomous driving, and expose an inherent weakness in their reliability.

In adversarial attacks, very small perturbations in the network's input data are performed, which lead to classifying an input erroneously with a high confidence. Even though these small changes in the input cause the model to err with high probability, they are unnoticeable to the human eye in most cases. In addition, it has been shown in [32] that such adversarial attacks tend to generalize well across models. This transferability trait only increases the possible susceptibility to attacks since an attacker might not need to know the structure of the specific attacked network in order to fool it. Thus, black-box attacks are highly successful as well. This inherent vulnerability of DNNs is somewhat counter intuitive since it exposes a fault in the model's ability to generalize well in very particular cases.

Lately, this phenomenon has been the focus of substantial research, which has focused on effective attack methods, defense methods and theoretical explanations to this inherent vulnerability of the model. Attack methods aim to alter the network's input data in order to deliberately cause it to fail in its task. Such methods include DeepFool [19], Fast Gradient Sign Method (FGSM) [8], Jacobian-based Saliency Map Attack (JSMA) [23], Universal Perturbations [20], Adversarial Transformation Networks [2], and more [3].

Several defense methods have been suggested to increase deep neural networks' robustness to adversarial attacks. Some of the strategies aim at detecting whether an input image is adversarial or not (e.g., [17, 12, 13, 35, 16, 6]). For example, the authors in [35] suggested to detect adversarial examples using feature squeezing, whereas the authors in [6] proposed to detect adversarial examples based on density estimates and Bayesian uncertainty estimates. Other strategies focus on making the network more robust to perturbed inputs. The latter, which is the focus of this work, aims at increasing the network's accuracy in performing its original task even when it is being fed with perturbed data, intended to mislead it. This increased model robustness has been shown to be achieved by several different methods.

These defense methods include, among others, Adversarial Training [8] which adds perturbed inputs along with their correct labels to the training dataset; Defensive Distillation [24], which trains two networks, where the first is a standard classification network and the second is trained to achieve an output similar to the first network in all classes; the Batch Adjusted Network Gradients (BANG) method [26], which balances gradients in the training batch by scaling up those that have lower magnitudes; Parseval Networks [4] which constrain the Lipschitz constant of each hidden layer in a DNN to be smaller than 1; the Ensemble method [31], which takes the label that maximizes the average of the output probabilities of the classifiers in the ensemble as the predicted label; a Robust Optimization Framework [27], which uses an alternating minimization-maximization procedure in which the loss of the network is minimized over perturbed examples that are generated at each parameter update; Virtual Adversarial Training (VAT) [18], which uses a regularization term to promote the smoothness of the model distribution; Input Gradient Regularization [25] which regularizes the gradient of the cross-entropy loss, and Cross-Lipschitz Regularization [9], which regularizes all the combinations of differences of the gradients of a network's output w.r.t its input. In another recent work [29], the authors suggested an adversarial training procedure that achieves robustness with guarantees on its statistical performance.

In addition to these works, several theoretical explanations for adversarial examples have been suggested. In [8], the authors claim that linear behavior in high-dimensional spaces creates this inherent vulnerability to adversarial examples. In [22], a game theoretical framework is used to study the relationship between attack and defense strategies in recognition systems in the context of adversarial attacks. In [33], the authors examine the transferability of adversarial examples between different models and find that adversarial examples span a contiguous subspace of large dimensionality. The authors also provide an insight into the decision boundaries of DNNs. In [14], the authors claim that first order attacks are universal and suggest the Projected Gradient Descent (PGD) attack which relies on this notion. They also claim that networks require a significantly larger capacity in order to be more robust to adversarial attacks. In another recent work [28], the authors show that the gradient of a network's objective function grows with the dimension of its input and conclude that the adversarial vulnerability of a network increases with the dimension of its input.

In [5], the authors showed the relationship between a network's sensitivity to additive adversarial perturbations and the curvature of the classification boundaries. In addition, they propose a method to discriminate between the original input and perturbed inputs. In [21], the link between a network's robustness to adversarial perturbations and the geometry of the decision boundaries of this network is further developed. Specifically, it is shown that when the decision boundary is positively curved, small universal perturbations are more likely to fool the classifier. However, a direct application of this insight to increase the networks' robustness to adversarial examples is, to the best of our knowledge, still unclear.

In a recent work [30], a relationship between the norm of the Jacobian of the network and its generalization error has been drawn. The authors have shown that by regularizing the Frobenius norm of the Jacobian matrix of the network's classification function, a lower generalization error is achieved. In [34] the authors show that using the Jacobian matrix computed at the logits (before the softmax operation) instead of the probabilities (after the softmax operation) yields better generalization results.

Inspired by the work in [30], we take this notion further and show that using Jacobian regularization as post-processing, i.e. applying it for a second phase of additional training after regular training has finished, also increases deep neural networks' robustness to adversarial perturbations. Besides the relationship to the generalization error, we show also that the Forbenius norm of the Jacobian at a given point is related to its distance to the closest adversarial example and to the curvature of the network's decision boundaries. All these connections provide a theoretical justification to the usage of the Jacobian regularization for decreasing the vulnerability of a network to adversarial attacks.

We apply the Jacobian regularization as post-processing to the regular training, after the network is stabilized with a high test accuracy, thereby allowing to use our strategy with existing pre-trained networks and improve their robustness. In addition, using the Jacobian regularization requires only little additional computational resources as it makes a single additional back-propagation step in each training step, as opposed to other methods that are very computationally demanding such as Distillation [24] which requires the training of two networks.

Two close techniques to our strategy are the Input Gradient regularization technique proposed in [25] and the Cross-Lipschitz regularization proposed in [9]. Our approach differs from the former work by the fact that we regularize the Frobenius norm of the Jacobian matrix of the network itself, and not the norm of the gradient of the cross-entropy loss. Our work differs from the latter work by the fact that we regularize the gradients of the network themselves and not all combinations of their differences, which yields better results at a lower computational cost, as will be later shown.

We compare the methods mentioned above and adversarial training [8] to Jacobian regularization on the MNIST, CIFAR-10 and CIFAR-100 datasets, demonstrating the advantage of our strategy in the form of high robustness to the DeepFool [19], FGSM [8], and JSMA [23] attack methods. Our method surpasses the results of the other strategies on FGSM and DeepFool and achieves competitive performance on JSMA. We also show that using Jacobian regularization combined with adversarial training further improves the robustness results.

This paper is organized as follows. Section 2 introduces the Jacobian regularization method and related strategies. Section 3 shows its connection to some theory of adversarial examples. The relationships drawn in this section suggest that regularizing the Jacobian of deep neural networks can improve their robustness to adversarial examples. In Section 4 we demonstrate empirically the advantages of this approach. Section 5 concludes our paper. In the supplementary material, which consists of eight appendices, we provide more theoretical insight and additional experimental results.

2 Jacobian Regularization for Adversarial Robustness

Adversarial perturbations are essentially small changes in the input data which cause large changes in the network's output. In order to prevent this vulnerability, during the post-processing training phase we penalize large gradients of the classification function with respect to the input data. Thus, we encourage the network's learned function to be more robust to small changes in the input space. This is achieved by adding a regularization term in the form of the Frobenius norm of the network's Jacobian matrix evaluated on the input data. The relation between the Frobenius norm and the ℓ_2 (spectral) norm of the Jacobian matrix has been shown in [30], and lays the justification for using the Frobenius norm of the network's Jacobian as a regularization term. We emphasize that we apply this regularization as additional post-processing training which is done after the regular training has finished.

To describe the Jacobian regularization more formally, we use the following notation. Let us denote the network's input as a D-dimensional vector, its output as a K-dimensional vector, and let us assume the training dataset X consists of N training examples. We use the index $l = 1, ..., L$ to specify a certain layer in a network with L layers. $z^{(l)}$ is the output of the l^{th} layer of the network and $z_k^{(l)}$ is the output of the k^{th} neuron in this layer. In addition, let us denote by λ the hyper-parameter which controls the weight of our regularization penalty in the loss function. The input to the network is

$$x_i \in \mathbb{R}^D, \qquad i = 1 \dots N, \qquad X = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} \in \mathbb{R}^{N \times D}, \tag{1}$$

and its output is $f(x_i) \in \mathbb{R}^K$, where the predicted class k_i^* for an input x_i is $k_i^* = \text{argmax}_k\, f_k(x_i)$, $k = 1, ..., K$.

$f(x_i) = \text{softmax}\{z^{(L)}(x_i)\}$ is the network's output after the softmax operation where $z^{(L)}(x_i)$ is the output of the last fully connected layer in the network for the input x_i. The term $\nabla_x z^{(L)}(x_i)$ is the Jacobian matrix of layer L evaluated at the point x_i, i.e. $J^{(L)}(x_i) = \nabla_x z^{(L)}(x_i)$. Correspondingly, $J_k^{(L)}(x_i) = \nabla_x z_k^{(L)}(x_i)$ is the k^{th} row in the matrix $J^{(L)}(x_i)$.

A network's Jacobian matrix is given by

$$J(x_i) \triangleq J^{(L)}(x_i) = \begin{bmatrix} \frac{\partial z_1^{(L)}(x_i)}{\partial x_{(1)}} & \cdots & \frac{\partial z_1^{(L)}(x_i)}{\partial x_{(D)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_K^{(L)}(x_i)}{\partial x_{(1)}} & \cdots & \frac{\partial z_K^{(L)}(x_i)}{\partial x_{(D)}} \end{bmatrix} \in \mathbb{R}^{K \times D}, \tag{2}$$

where $x = (x_{(1)} \dots x_{(D)})^T$. Accordingly, the Jacobian regularization term for an input sample x_i is

$$||J(x_i)||_F^2 = \sum_{d=1}^{D} \sum_{k=1}^{K} \left(\frac{\partial}{\partial x_d} z_k^{(L)}(x_i) \right)^2 = \sum_{k=1}^{K} ||\nabla_x z_k^{(L)}(x_i)||_2^2. \tag{3}$$

Combining the regularization term in (3) with a standard cross-entropy loss function on the training data, we get the following loss function for training:

$$Loss = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i) + \lambda \sqrt{\sum_{d=1}^{D} \sum_{k=1}^{K} \sum_{i=1}^{N} \left(\frac{\partial}{\partial x_d} z_k^{(L)}(x_i) \right)^2}, \tag{4}$$

where $y_i \in \mathbb{R}^K$ is a one-hot vector representing the correct class of the input x_i.

The Input Gradient regularization method from [25] uses the following regularization term:

$$\sum_{d=1}^{D} \sum_{i=1}^{N} \left(\frac{\partial}{\partial x_d} \sum_{k=1}^{K} -y_{ik} \log f_k(x_i) \right)^2. \tag{5}$$

The Cross-Lipschitz regularization method from [9] uses the following regularization term:

$$\sum_{i=1}^{N} \sum_{j,k=1}^{K} ||\nabla_x z_k^{(L)}(x_i) - \nabla_x z_j^{(L)}(x_i)||_2^2. \tag{6}$$

The adversarial training method [8] adds perturbed inputs along with their correct labels to the training dataset, so that the network learns the correct labels of perturbed inputs during training. This helps the network to achieve a higher accuracy when it is being fed with new perturbed inputs, meaning the network becomes more robust to adversarial examples.

On the computational complexity aspect, Jacobian regularization introduces an overhead of one additional back-propagation step in every iteration. This step involves the computation of mixed partial derivatives, as the first derivative is w.r.t the input, and the second is w.r.t. the model parameters. However, one should keep in mind that Jacobian regularization is applied as a post-processing phase, and not throughout the entire training, which is computationally beneficial. Moreover, it is also more efficient than the Cross-Lipschitz regularization technique [9], which requires the computation of the norm of $\frac{1}{2}K(K-1)$ terms as opposed to our method that only requires the calculation of the norm of K different gradients. This makes Jacobian regularization more scalable for datasets with a large K.

3 Theoretical Justification

3.1 The Jacobian matrix and adversarial perturbations

In essence, for a network performing a classification task, an adversarial attack (a fooling method) aims at making a change as small as possible, which changes the network's decision. In other words, finding the smallest perturbation that causes the output function to cross a decision boundary to another class, thus making a classification error. In general, an attack would seek for the closest decision boundary to be reached by an adversarial perturbation in the input space. This makes the attack the least noticeable and the least prone to being discovered [8].

To gain some intuition for our proposed defense method, we start with a simple informal explanation on the relationship between adversarial perturbations and the Jacobian matrix of a network. Let x be a given input data sample; x_{same} a data sample close to x from the same class that was not perturbed by an adversarial attack; and x_{pert} another data sample, which is the result of an adversarial perturbation of x that keeps it close to it but with a different predicted label. Therefore, we have that for the ℓ_2 distance metric in the input and output of the network

$$\frac{||x_{pert} - x||_2}{||x_{same} - x||_2} \approx 1 \quad \text{and} \quad 1 < \frac{||z^{(L)}(x_{pert}) - z^{(L)}(x)||_2}{||z^{(L)}(x_{same}) - z^{(L)}(x)||_2}, \tag{7}$$

with a high probability. Therefore,

$$\frac{||z^{(L)}(x_{same}) - z^{(L)}(x)||_2}{||x_{same} - x||_2} < \frac{||z^{(L)}(x_{pert}) - z^{(L)}(x)||_2}{||x_{pert} - x||_2}. \tag{8}$$

Let $[x, x_{pert}]$ be the D-dimensional line in the input space connecting x and x_{pert}. According to the mean value theorem there exists some $x' \in [x, x_{pert}]$ such that

$$\frac{||z^{(L)}(x_{pert}) - z^{(L)}(x)||_2^2}{||x_{pert} - x||_2^2} \leq \sum_{k=1}^{K} ||\nabla_x z_k^{(L)}(x')||_2^2 = ||J(x')||_F^2. \tag{9}$$

This suggests that a lower Frobenius norm of the network's Jacobian matrix encourages it to be more robust to small changes in the input space. In other words, the network is encouraged to yield similar outputs for similar inputs.

We empirically examined the average values of the Frobenius norm of the Jacobian matrix of networks trained with various defense methods on the MNIST dataset. The network architecture is described in Section 4. Table 1 presents these values for both the original inputs and the ones which have been perturbed by DeepFool [19]. For "regular" training with no defense, it can be seen that as predicted, the aforementioned average norm is significantly larger on perturbed inputs. Interestingly enough, using adversarial training, which does not regularize the Jacobian matrix directly, decreases the average Frobenius norm of the Jacobian matrix evaluated on perturbed inputs (second row of Table 1). Yet, when Jacobian regularization is added (with $\lambda = 0.1$), this norm is reduced much more (third and fourth rows of Table 1). Thus, it is expected to improve the robustness of the network even further. Indeed, this behavior is demonstrated in Section 4.

Table 1: Average Frobenius norm of the Jacobian matrix at the original data and the data perturbed by DeepFool. DNN is trained on MNIST with various defense methods

Defense method	$\frac{1}{N}\sum_{i=1}^{N} \|\|J(x_i)\|\|_F$	$\frac{1}{N}\sum_{i=1}^{N} \|\|J(x_{i_{pert}})\|\|_F$
No defense	0.14	0.1877
Adversarial Training	0.141	0.143
Jacobian regularization	0.0315	0.055
Jacobian regularization & Adversarial Training	0.0301	0.0545

3.2 Relation to classification decision boundaries

As shown in [19], we may locally treat the decision boundaries as hyper-surfaces in the K-dimensional output space of the network. Let us denote $g(x) = w^T x +$

$b = 0$ as a hyper-plane tangent to such a decision boundary hyper-surface in the input space. Using this notion, the following lemma approximates the distance between an input and a perturbed input classified to be at the boundary of a hyper-surface separating between the class of x, k_1, and another class k_2.

Lemma 1 *The first order approximation for the distance between an input x, with class k_1, and a perturbed input classified to the boundary hyper-surface separating the classes k_1 and k_2 for an ℓ_2 distance metric is given by*

$$d = \frac{|z_{k_1}^{(L)}(x) - z_{k_2}^{(L)}(x)|}{||\nabla_x z_{k_1}^{(L)}(x) - \nabla_x z_{k_2}^{(L)}(x)||_2}. \tag{10}$$

This lemma is given in [19]. For completeness, we present a short sketch of the proof in Appendix A. Based on this lemma, the following corollary provides a proxy for the minimal distance that may lead to fooling the network.

Corollary 2 *Let k^* be the correct class for the input sample x. Then the ℓ_2 norm of the minimal perturbation necessary to fool the classification function is approximated by*

$$d^* = \min_{k \neq k^*} \frac{|z_{k^*}^{(L)}(x) - z_{k}^{(L)}(x)|}{||\nabla_x z_{k^*}^{(L)}(x) - \nabla_x z_{k}^{(L)}(x)||_2}. \tag{11}$$

To make a direct connection to the Jacobian of the network, we provide the following proposition:

Proposition 3 *Let k^* be the correct class for the input sample x. Then the first order approximation for the ℓ_2 norm of the minimal perturbation necessary to fool the classification function is lower bounded by*

$$d^* \geq \frac{1}{\sqrt{2}||J^{(L)}(x)||_F} \min_{k \neq k^*} |z_{k^*}^{(L)}(x) - z_{k}^{(L)}(x)|. \tag{12}$$

The proof of Proposition 3 is given in Appendix B. The term $|z_{k^*}^{(L)}(x) - z_{k}^{(L)}(x)|$ in (12) is maximized by the minimization of the cross-entropy term of the loss function, since a DNN aspires to learn the correct output with the largest confidence possible, meaning the largest possible margin in the output space between the correct classification and the other possible classes. The term $||J^{(L)}(x)||_F$ in the denominator is the Frobenius norm of the Jacobian of the last fully connected layer of the network. It is minimized due to the Jacobian regularization part in the loss function. This is essentially a min-max problem, since we wish to maximize the minimal distance necessary to fool the network, d^*. For this reason, applying Jacobian regularization during training increases the minimal distance necessary to fool the DNN, thus providing improved robustness to adversarial perturbations. One should keep in mind that it is important not to deteriorate the network's original test accuracy. This is indeed the case as shown in Section 4.

An important question is whether the regularization of the Jacobian at earlier layers of the network would yield better robustness to adversarial examples. To this end, we examined imposing the regularization on the $L-1$ and the $L-2$ layers of the network. Both of these cases generally yielded degraded robustness results compared to imposing the regularization on the last layer of the network. Thus, throughout this work we regularize the Jacobian of the whole network. The theoretical details are given in Appendix C and the corresponding experimental results are given in Appendix D.

3.3 Relation to decision boundary curvature

In [21] the authors show the link between a network's robustness to adversarial perturbations and the geometry of its decision boundaries. The authors show that when the decision boundaries are positively curved the network is fooled by small universal perturbations with a higher probability. Here we show that Jacobian regularization promotes the curvature of the decision boundaries to be less positive, thus reducing the probability of the network being fooled by small universal adversarial perturbations.

Let $H_k(x) = \frac{\partial^2 z_k^{(L)}(x)}{\partial x^2}$ be the Hessian matrix of the network's classification function at the input point x for the class k. As shown in [21], the decision boundary between two classes k_1 and k_2 can be locally referred to as the hypersurface $F_{k_1,k_2}(x) = z_{k_1}^{(L)}(x) - z_{k_2}^{(L)}(x) = 0$. Relying on the work in [15] let us use the approximation $H_k(x) \approx J_k(x)^T J_k(x)$ where $J_k(x)$ is the k^{th} row in the matrix $J(x)$. The matrix $J_k(x)^T J_k(x)$ is a rank one positive semi-definite matrix. Thus, the curvature of the decision boundary $F_{k_1,k_2}(x)$ is given by $x^T(H_{k_1} - H_{k_2})x$, which using the aforementioned approximation, can be approximated by

$$x^T \left(J_{k_1}(x)^T J_{k_1}(x) - J_{k_2}(x)^T J_{k_2}(x)\right) x = \left(J_{k_1}(x)x\right)^2 - \left(J_{k_2}(x)x\right)^2 . \tag{13}$$

Thus, we arrive at the following upper bound for the curvature:

$$\left(J_{k_1}(x)x\right)^2 - \left(J_{k_2}(x)x\right)^2 \le \left(J_{k_1}(x)x\right)^2 + \left(J_{k_2}(x)x\right)^2 \tag{14}$$

$$\le \sum_{k=1}^{K} \left(J_k(x)x\right)^2 \le ||J(x)||_F^2 ||x||_2^2, \tag{15}$$

where the last inequality stems from the matrix norm inequality. For this reason the regularization of $||J(x)||_F$ promotes a less positive curvature of the decision boundaries in the environment of the input samples. This offers a geometric intuition to the effect of Jacobian regularization on the network's decision boundaries. Discouraging a positive curvature makes a universal adversarial perturbation less likely to fool the classifier.

4 Experiments

We tested the performance of Jacobian regularization on the MNIST, CIFAR-10 and CIFAR-100 datasets. The results for CIFAR-100, which are generally consistent with the results for MNIST and CIFAR-10, are given in Appendix E. As mentioned before, we use the training with Jacobian regularization as a post-processing phase to the "regular" training. Using a post-processing training phase is highly beneficial: it has a low additional computational cost as we add the regularization part after the network is already stabilized with a high test accuracy and not throughout the entire training. It also allows taking an existing network and applying the post-processing training phase to it in order to increase its robustness to adversarial examples. We obtained optimal results this way, whereas we found that applying the Jacobian regularization from the beginning of the training yields a lower final test accuracy.

The improved test accuracy obtained using post-processing training can be explained by the advantage of keeping the original training phase, which allows the network to train solely for the purpose of a high test accuracy. The subsequent post-processing training phase with Jacobian regularization introduces a small change to the already existing good test accuracy, as opposed to the case where the regularization is applied from the beginning that results in a worse test accuracy. Table 2 presents a comparison between post-processing training and "regular" training on MNIST. Similar results are obtained for CIFAR-10 and CIFAR-100.

We examine the performance of our method using three different adversarial attack methods: DeepFool [19], FGSM [8] and JSMA [23]. We also assess the performance of our defense combined with adversarial training, which is shown to be effective in improving the model's robustness. However, this comes at the cost of generating and training on a substantial amount of additional input samples as is the practice in adversarial training. We found that the amount of perturbed inputs in the training mini-batch has an impact on the overall achieved robustness. An evaluation of this matter appears in Appendix F. The results for adversarial training, shown hereafter, are given for the amount of perturbed inputs that yields the optimal results in each test case. We also compare the results to the Input Gradient regularization technique [25] and the Cross-Lipschitz regularization technique [9].

Table 2: Effect of post-processing training vs. "regular" training on MNIST, using different defense methods

Defense method	Test accuracy	$\hat{\rho}_{adv}$
No defense	99.08%	20.67 x 10^{-2}
Input Gradient regularization, "regular" training	99.25%	23.43 x 10^{-2}
Input Gradient regularization, post-processing training	99.44%	24.03 x 10^{-2}
Cross-Lipschitz regularization, "regular" training	98.64%	29.03 x 10^{-2}
Cross-Lipschitz regularization, post-processing training	98.91%	29.99 x 10^{-2}
Jacobian regularization, "regular" training	98.35%	32.89 x 10^{-2}
Jacobian regularization, post-processing training	98.44%	34.24 x 10^{-2}

For MNIST we used the network from the official TensorFlow tutorial [1]. The network consists of two convolutional layers, each followed by a max pooling layer. These layers are then followed by two fully connected layers. All these layers use the ReLU activation function, except for the last layer which is followed by a softmax operation. Dropout regularization with 0.5 keep probability is applied to the fully connected layers. The training is done using an Adam optimizer [11] and a mini-batch size of 500 inputs. With this network we obtained a test accuracy of 99.08%. Training with Jacobian regularization was done with a weight of $\lambda = 0.1$, which we found to provide a good balance between the cross-entropy loss and the Jacobian regularization.

For CIFAR-10 we used a convolutional neural network consisting of four concatenated sets, where each set consists of two convolutional layers followed by a max pooling layer followed by dropout with a 0.75 keep probability. After these four sets, two fully connected layers are used. For CIFAR-10, training was done with a RMSProp optimizer [10] and a mini-batch size of 128 inputs. With this network we obtained a test accuracy of 88.79%. Training with Jacobian regularization was done with a weight of $\lambda = 0.5$, which we found to provide a good balance between the cross-entropy loss and the Jacobian regularization.

The results of an ablation study regarding the influence of variation in the values of λ for MNIST and CIFAR-10 are given in Appendix G.

4.1 DeepFool evaluation

We start by evaluating the performance of our method compared to the others under the DeepFool attack. The DeepFool attack [19] uses a first order approximation of the network's decision boundaries as hyper-planes. Using this approximation, the method seeks for the closest decision boundary to be reached by a change in the input. Since the decision boundaries are not actually linear, this process continues iteratively until the perturbed input changes the network's decision. The robustness metric associated with this attack is $\hat{\rho}_{adv} = \frac{1}{N}\sum_{i=1}^{N}\frac{d_i}{||x_i||_2}$, which represents the average proportion between the ℓ_2 norm of the minimal perturbation necessary to fool the network for an input x_i and the ℓ_2 norm of x_i. This attack is optimized for the ℓ_2 metric.

Table 3 and Table 4 present the robustness measured by $\hat{\rho}_{adv}$ under a DeepFool attack for MNIST and CIFAR-10 respectively. As the results show, Jacobian regularization provides a much more significant robustness improvement compared to the other methods. Substantially smaller perturbation norms are required to fool networks that use those defense approaches compared to networks that are trained using Jacobian regularization. Moreover, combining it with adversarial training further enhances this difference in the results.

Notice that neither of the examined defense methods change the test accuracy significantly. For MNIST, the Jacobian and Cross-Lipschitz regularizations and adversarial training cause a small accuracy decrease, whereas the Input Gradient regularization technique improves the accuracy. Conversely, for CIFAR-10, the Jacobian and Cross-Lipschitz regularizations and adversarial training yield a better accuracy, whereas the Input Gradient regularization reduces the accuracy.

Table 3: Robustness to DeepFool attack for MNIST

Defense method	Test accuracy	$\hat{\rho}_{adv}$
No defense	99.08%	20.67 x 10^{-2}
Adversarial Training	99.03%	22.38 x 10^{-2}
Input Gradient regularization	99.25%	23.43 x 10^{-2}
Input Gradient regularization & Adversarial Training	98.88%	23.49 x 10^{-2}
Cross-Lipschitz regularization	98.64%	29.03 x 10^{-2}
Cross-Lipschitz regularization & Adversarial Training	98.73%	32.38 x 10^{-2}
Jacobian regularization	98.44%	34.24 x 10^{-2}
Jacobian regularization & Adversarial Training	98%	36.29 x 10^{-2}

Table 4: Robustness to DeepFool attack for CIFAR-10

Defense method	Test accuracy	$\hat{\rho}_{adv}$
No defense	88.79%	1.21 x 10^{-2}
Adversarial Training	88.88%	1.23 x 10^{-2}
Input Gradient regularization	88.56%	1.43 x 10^{-2}
Input Gradient regularization & Adversarial Training	88.49%	2.17 x 10^{-2}
Cross-Lipschitz regularization	88.91%	2.08 x 10^{-2}
Cross-Lipschitz regularization & Adversarial Training	88.49%	4.04 x 10^{-2}
Jacobian regularization	89.16%	3.42 x 10^{-2}
Jacobian regularization & Adversarial Training	88.49%	6.03 x 10^{-2}

4.2 FGSM evaluation

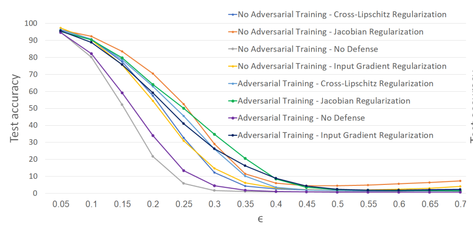
The FGSM (Fast Gradient Sign Method) attack [8] was designed to rapidly create adversarial examples that could fool the network. The method changes the network's input according to:

$$x_{pert} = x - \epsilon \cdot sign\left(\nabla_x Loss(x)\right), \tag{16}$$

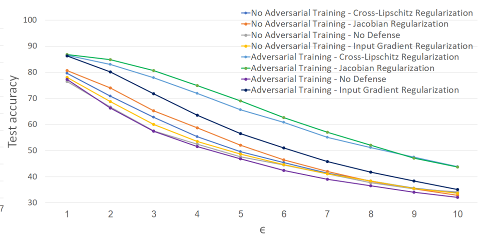
where ϵ represents the magnitude of the attack. This attack is optimized for the ℓ_∞ metric.

We examined the discussed defense methods' test accuracy under the FGSM attack (test accuracy on the perturbed dataset) for different values of ϵ. Fig. 1 presents the results comparing Jacobian regularization to adversarial training, Input Gradient regularization and Cross-Lipschitz regularization. In all cases, the minimal test accuracy on the original test set using Jacobian regularization is 98% for MNIST and 88.49% for CIFAR-10.

Similarly to the results under the DeepFool attack, the results under the FGSM attack show that the test accuracy with the Jacobian regularization defense is higher than with the Input Gradient and Cross-Lipschitz regularizations or with adversarial training. Moreover, if adversarial training is combined with Jacobian regularization, its advantage over using the other techniques is even more distinct. This leads to the conclusion that the Jacobian regularization method yields a more robust network to the FGSM attack.



(a) MNIST



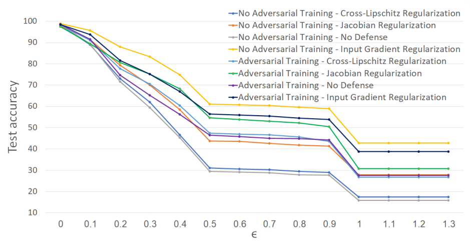
(b) CIFAR-10

Fig. 1: Test accuracy for FGSM attack on MNIST (left) and CIFAR-10 (right) for different values of ϵ

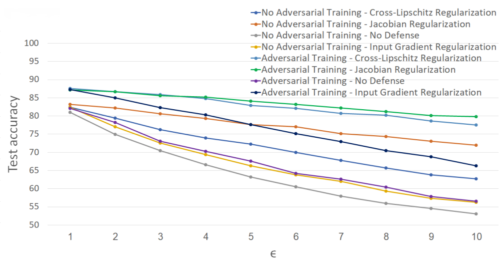
4.3 JSMA evaluation

The JSMA (Jacobian-based Saliency Map Attack) [23] attack relies on the computation of a Saliency Map, which outlines the impact of every input pixel on the classification decision. The method picks at every iteration the most influential pixel to be changed such that the likelihood of the target class is increased. We leave the mathematical details to the original paper. Similarly to FGSM, ϵ represents the magnitude of the attack. The attack is repeated iteratively, and is optimized for the ℓ_0 metric.

We examined the defense methods' test accuracy under the JSMA attack (test accuracy on the perturbed dataset) for different values of ϵ. Fig. 2 presents the results for the MNIST and CIFAR-10 datasets. The parameters of the JSMA attack are 80 epochs, 1 pixel attack for the former and 200 epochs, 1 pixel attack, for the latter. In all cases, the minimal test accuracy on the original test set using Jacobian regularization is 98% for MNIST and 88.49% for CIFAR-10.



(a) MNIST



(b) CIFAR-10

Fig. 2: Test accuracy for JSMA (1 pixel) attack on MNIST with 80 epochs (left) and CIFAR-10 with 200 epochs (right) for different values of ϵ

Our method achieves superior results compared to the other three methods on CIFAR-10. On the other hand, on MNIST we obtain an inferior performance compared to the Input Gradient regularization method, though we obtain a better performance compared to Cross-Lipschitz regularization. Thus, we conclude that our defense method is effective under the JSMA attack in some cases and presents competitive performance overall. We believe that the reason behind the failure of our method in the MNIST case can be explained by our theoretical analysis. In the formulation of the Jacobian regularization (based on the Frobenius norm of the Jacobian matrix), the metric that is being minimized is the ℓ_2 norm. Yet, in the JSMA attack, the metric that is being targeted by the perturbation is the ℓ_0 pseudo-norm as only one pixel is being changed in every epoch. We provide more details on this issue in Appendix H.

5 Discussion and Conclusions

This paper introduced the Jacobian regularization method for improving DNNs' robustness to adversarial examples. We provided a theoretical foundation for its usage and showed that it yields a high degree of robustness, whilst preserving the network's test accuracy. We demonstrated its improvement in reducing the vulnerability to various adversarial attacks (DeepFool, FGSM and JSMA) on the MNIST, CIFAR-10 and CIFAR-100 datasets. Under all three examined attack methods Jacobian regularization exhibits a large improvement in the network's robustness to adversarial examples, while only slightly changing the network's performance on the original test set. Moreover, in general, Jacobian regularization without adversarial training is better than adversarial training without Jacobian regularization, whereas the combination of the two defense methods provides even better results. Compared to the Input Gradient regularization, our proposed approach achieves superior performance under two out of the three attacks and competitive ones on the third (JSMA). Compared to the Cross-Lipschitz regularization, our proposed approach achieves superior performance under all of the three examined attacks.

We believe that our approach, with its theoretical justification, may open the door to other novel strategies for defense against adversarial attacks.

In the current form of regularization of the Jacobian, its norm is evaluated at the input samples. We empirically deduced that the optimal results are obtained by applying the Jacobian regularization on the original input samples, which is also more efficient computationally, and not on perturbed input samples or on points in the input space for which the Frobenius norm of the Jacobian matrix is maximal. A future work may analyze the reasons for that.

Notice that in the Frobenius norm, all the rows of the Jacobian matrix are penalized equally. Another possible future research direction is providing a different weight for each row. This may be achieved by either using a weighted version of the Frobenius norm or by replacing it with other norms such as the spectral one. Note, though, that the latter option is more computationally demanding compared to our proposed approach.

Acknowledgment. This work is partially supported by the ERC-StG SPADE grant and the MAGNET MDM grant.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow. org **1** (2015)
2. Baluja, S., Fischer, I.: Learning to attack: Adversarial transformation networks. In: AAAI (2018)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE Symposium on Security and Privacy (2017)
4. Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., Usunier, N.: Parseval networks: Improving robustness to adversarial examples. In: ICML (2017)
5. Fawzi, A., Moosavi-Dezfooli, S., Frossard, P., Soatto, S.: Classification regions of deep neural networks. arXiv **abs/1705.09552** (2017)
6. Feinman, R., Curtin, R.R., Shintre, S., Gardner, A.B.: Detecting adversarial samples from artifacts. arXiv **abs/1703.00410** (2017), https://arxiv.org/pdf/1703.00410.pdf
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
8. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
9. Hein, M., Andriushchenko, M.: Formal guarantees on the robustness of a classifier against adversarial manipulation. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 2266–2276. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/6821-formal-guarantees-on-the-robustness-of-a-classifier-against-adversarial-manipulation.pdf
10. Hinton, G.: Networks for machine learning - lecture 6a - overview of mini-batch gradient descent (2012)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015), http://arxiv.org/abs/1412.6980
12. Li, X., Li, F.: Adversarial examples detection in deep networks with convolutional filter statistics. In: ICCV (2017)
13. Lu, J., Issaranon, T., Forsyth, D.: Safetynet: Detecting and rejecting adversarial examples robustly. In: ICCV (2017)
14. Madryi, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv **abs/1706.06083** (2017), https://arxiv.org/pdf/1706.06083.pdf
15. Martens, J., Sutskever, I., Swersky, K.: Estimating the hessian by back-propagating curvature. In: Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012 (2012)
16. Meng, D., Chen, H.: Magnet: A two-pronged defense against adversarial examples. In: ACM SIGSAC Conference on Computer and Communications Security. pp. 135–147 (2017)
17. Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On detecting adversarial perturbations. In: ICLR (2017)
18. Miyato, T., ichi Maeda, S., Koyama, M., Nakae, K., Ishii, S.: Distributional smoothing with virtual adversarial training. In: ICLR (2016), https://arxiv.org/pdf/1507.00677.pdf
19. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: A simple and accurate method to fool deep neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2574–2582 (June 2016)

20. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: CVPR (2017)
21. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P., Soatto, S.: Analysis of universal adversarial perturbations. arXiv **abs/1705.09554** (2017)
22. Oh, S.J., Fritz, M., Schiele, B.: Adversarial image perturbation for privacy protection a game theory perspective. In: ICCV (2017)
23. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 1st IEEE European Symposium on Security and Privacy (2016)
24. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 37th IEEE Symposium on Security and Privacy (2016)
25. Ross, A.S., Doshi-Velez, F.: Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. CoRR **abs/1711.09404** (2017), http://arxiv.org/abs/1711.09404
26. Rozsa, A., Gunther, M., E. Boult, T.: Towards robust deep neural networks with bang. In: WACV (2018)
27. Shaham, U., Yamada, Y., Negahban, S.: Understanding adversarial training: Increasing local stability of neural nets through robust optimization. arXiv **abs/1511.05432** (2016), https://arxiv.org/pdf/1511.05432.pdf
28. Simon-Gabriel, C.J., Ollivier, Y., Bottou, L., Schlkopf, B., Lopez-Paz, D.: Adversarial vulnerability of neural networks increases with input dimension. arXiv **abs/1802.01421** (2018), https://arxiv.org/pdf/1802.01421.pdf
29. Sinha, A., Namkoong, H., Duchi, J.: Certifying some distributional robustness with principled adversarial training. In: ICLR (2018)
30. Sokolic, J., Giryes, R., Sapiro, G., Rodrigues, M.R.D.: Robust large margin deep neural networks. IEEE Transactions on Signal Processing **65**(16), 4265–4280 (Aug 2017)
31. Strauss, T., Hanselmann, M., Junginger, A., Ulmer, H.: Ensemble methods as a defense to adversarial perturbations against deep neural networks. arXiv **abs/1709.03423** (2018), https://arxiv.org/pdf/1709.03423.pdf
32. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014), http://arxiv.org/abs/1312.6199
33. Tramer, F., Papernot, N., Goodfellow, I., Boneh1, D., McDaniel, P.: The space of transferable adversarial examples. arXiv **abs/1704.03453** (2017), https://arxiv.org/pdf/1704.03453.pdf
34. Varga, D., Csiszarik, A., Zombori, Z.: Gradient regularization improves accuracy of discriminative models. arXiv **abs/1712.09936** (2018), https://arxiv.org/pdf/1712.09936.pdf
35. Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. In: To appear in Network and Distributed Systems Security Symposium (NDSS) (2018)