Formal Guarantees on the Robustness of a Classifier against Adversarial Manipulation

Matthias Hein and Maksym Andriushchenko
Department of Mathematics and Computer Science
Saarland University, Saarbrücken Informatics Campus, Germany

Abstract

Recent work has shown that state-of-the-art classifiers are quite brittle, in the sense that a small adversarial change of an originally with high confidence correctly classified input leads to a wrong classification again with high confidence. This raises concerns that such classifiers are vulnerable to attacks and calls into question their usage in safety-critical systems. We show in this paper for the first time formal guarantees on the robustness of a classifier by giving instance-specific *lower bounds* on the norm of the input manipulation required to change the classifier decision. Based on this analysis we propose the Cross-Lipschitz regularization functional. We show that using this form of regularization in kernel methods resp. neural networks improves the robustness of the classifier with no or small loss in prediction performance.

1 Introduction

The problem of adversarial manipulation of classifiers has been addressed initially in the area of spam email detection, see e.g. [5, 16]. The goal of the spammer is to manipulate the spam email (the input of the classifier) in such a way that it is not detected by the classifier. In deep learning the problem was brought up in the seminal paper by [24]. They showed for state-of-the-art deep neural networks, that one can manipulate an originally correctly classified input image with a *non-perceivable* small transformation so that the classifier now misclassifies this image with high confidence, see [7] or Figure 4 for an illustration. This property calls into question the usage of neural networks and other classifiers showing this behavior in safety critical systems, as they are vulnerable to attacks. On the other hand this also shows that the concepts learned by a classifier are still quite far away from the visual perception of humans. Subsequent research has found fast ways to generate adversarial samples with high probability [7, 12, 19] and suggested to use them during training as a form of data augmentation to gain more robustness. However, it turns out that the so-called adversarial training does not settle the problem as one can yet again construct adversarial examples for the final classifier. Interestingly, it has recently been shown that there exist universal adversarial changes which when applied lead, for every image, to a wrong classification with high probability [17]. While one needs access to the neural network model for the generation of adversarial changes, it has been shown that adversarial manipulations generalize across neural networks [18, 15, 14], which means that neural network classifiers can be attacked even as a black-box method. The most extreme case has been shown recently [15], where they attack the commercial system Clarifai, which is a black-box system as neither the underlying classifier nor the training data are known. Nevertheless, they could successfully generate adversarial images with an existing network and fool this commercial system. This emphasizes that there are indeed severe security issues with modern neural networks. While countermeasures have been proposed [8, 7, 26, 18, 12, 2], none of them provides a guarantee of preventing this behavior [3]. One might think that generative adversarial neural networks should be resistant to this problem, but it has recently been shown [13] that they can also be attacked by adversarial manipulation of input images.

In this paper we show for the first time instance-specific formal guarantees on the robustness of a classifier against adversarial manipulation. That means we provide *lower bounds* on the norm of the change of the input required to alter the classifier decision or said otherwise: we provide a guarantee that the classifier decision does not change in a certain ball around the considered instance. We exemplify our technique for two widely used family of classifiers: kernel methods and neural networks. Based on the analysis we propose a new regularization functional, which we call *Cross-Lipschitz Regularization*. This regularization functional can be used in kernel methods and neural networks. We show that using Cross-Lipschitz regularization improves both the formal guarantees of the resulting classifier (lower bounds) as well as the change required for adversarial manipulation (upper bounds) while maintaining similar prediction performance achievable with other forms of regularization. While there exist fast ways to generate adversarial samples [7, 12, 19] without constraints, we provide algorithms based on the first order approximation of the classifier which generate adversarial samples satisfying box constraints in $O(d \log d)$, where d is the input dimension.

2 Formal Robustness Guarantees for Classifiers

In the following we consider the multi-class setting for K classes and d features where one has a classifier $f : \mathbb{R}^d \to \mathbb{R}^K$ and a point x is classified via $c = \mathop{\arg\max}\limits_{j=1,\ldots,K} f_j(x)$. We call a classifier robust at x if small changes of the input do not alter the decision. Formally, the problem can be described as follows [24]. Suppose that the classifier outputs class c for input x, that is $f_c(x) > f_j(x)$ for $j \neq c$ (we assume the decision is unique). The problem of generating an input $x + \delta$ such that the classifier decision changes, can be formulated as

$$\min_{\delta \in \mathbb{R}^d} \left\|\delta\right\|_p, \qquad \text{s.th.} \quad \max_{l \neq c} \; f_l(x+\delta) \geq f_c(x+\delta) \;\; \text{and} \;\; x+\delta \in C, \tag{1}$$

where C is a constraint set specifying certain requirements on the generated input $x + \delta$, e.g., an image has to be in $[0, 1]^d$. Typically, the optimization problem (1) is non-convex and thus intractable. The so generated points $x + \delta$ are called *adversarial samples*. Depending on the p-norm the perturbations have different characteristics: for $p = \infty$ the perturbations are small and affect all features, whereas for $p = 1$ one gets sparse solutions up to the extreme case that only a single feature is changed. In [24] they used $p = 2$ which leads to more spread but still localized perturbations. The striking result of [24, 7] was that for most instances in computer vision datasets, the change δ necessary to alter the decision is astonishingly small and thus clearly the label should not change. However, we will see later that our new regularizer leads to robust classifiers in the sense that the required adversarial change is so large that now also the class label changes (we have found the correct decision boundary), see Fig 4. Already in [24] it is suggested to add the generated adversarial samples as a form of data augmentation during the training of neural networks in order to achieve robustness. This is denoted as *adversarial training*. Later on fast ways to approximately solve (1) were proposed in order to speed up the adversarial training process [7, 12, 19]. However, in this way, given that the approximation is successful, that is $\mathop{\arg\max}\limits_j f_j(x + \delta) \neq c$, one gets just upper bounds on the perturbation necessary to change the classifier decision. Also it was noted early on, that the final classifier achieved by adversarial training is again vulnerable to adversarial samples [7]. Robust optimization has been suggested as a measure against adversarial manipulation [12, 21] which effectively boils down to adversarial training in practice. It is thus fair to say that up to date no mechanism exists which prevents the generation of adversarial samples nor can defend against it [3].

In this paper we focus instead on robustness guarantees, that is we show that the classifier decision does not change in a small ball around the instance. Thus our guarantees hold for any method to generate adversarial samples or input transformations due to noise or sensor failure etc. Such formal guarantees are in our point of view absolutely necessary when a classifier becomes part of a safety-critical technical system such as autonomous driving. In the following we will first show how one can achieve such a guarantee and then explicitly derive bounds for kernel methods and neural networks. We think that such formal guarantees on robustness should be investigated further and it should become standard to report them for different classifiers alongside the usual performance measures.

2.1 Formal Robustness Guarantee against Adversarial Manipulation

The following guarantee holds for any classifier which is continuously differentiable with respect to the input in each output component. It is instance-specific and depends to some extent on the confidence in the decision, at least if we measure confidence by the relative difference $f_c(x) - \max_{j\neq c} f_j(x)$ as it is typical for the cross-entropy loss and other multi-class losses. In the following we use the notation $B_p(x,R) = \{y \in \mathbb{R}^d \mid \|x-y\|_p \leq R\}$.

Theorem 2.1. *Let $x \in \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}^K$ be a multi-class classifier with continuously differentiable components and let $c = \underset{j=1,\dots,K}{\arg\max} f_j(x)$ be the class which f predicts for x. Let $q \in \mathbb{R}$ be defined as $\frac{1}{p} + \frac{1}{q} = 1$, then for all $\delta \in \mathbb{R}^d$ with*

$$\|\delta\|_p \;\leq\; \max_{R>0} \min \left\{ \min_{j\neq c} \frac{f_c(x) - f_j(x)}{\max\limits_{y\in B_p(x,R)} \|\nabla f_c(y) - \nabla f_j(y)\|_q},\; R \right\} := \alpha,$$

it holds $c = \underset{j=1,\dots,K}{\arg\max} f_j(x+\delta)$, that is the classifier decision does not change on $B_p(x,\alpha)$.

Proof. By the main theorem of calculus, it holds that

$$f_j(x+\delta) = f_j(x) + \int_0^1 \langle \nabla f_j(x+t\delta), \delta\rangle \, dt, \qquad \text{for } j = 1,\dots,K.$$

Thus, in order to achieve $f_j(x+\delta) \geq f_c(x+\delta)$, it has to hold that

$$\begin{aligned} 0 \leq f_c(x) - f_j(x) &\leq \int_0^1 \langle \nabla f_j(x+t\delta) - \nabla f_c(x+t\delta), \delta\rangle \, dt \\ &\leq \|\delta\|_p \int_0^1 \|\nabla f_j(x+t\delta) - \nabla f_c(x+t\delta)\|_q \, dt, \end{aligned}$$

where the first inequality holds as $f_c(x) \geq f_j(x)$ for all $j = 1,\dots,K$ and in the last step we have used Hölder inequality together with the fact that the q-norm is dual to the p-norm, where q is defined via $\frac{1}{p} + \frac{1}{q} = 1$. Thus the minimal norm of the change δ required to change the classifier decision from c to j satisfies

$$\|\delta\|_p \geq \frac{f_c(x) - f_j(x)}{\int_0^1 \|\nabla f_j(x+t\delta) - \nabla f_c(x+t\delta)\|_q \, dt}.$$

We upper bound the denominator over some fixed ball $B_p(x,R)$. Note that by doing this, we can only make assertions of perturbations $\delta \in B_p(0,R)$ and thus the upper bound in the guarantee is at most R. It holds

$$\sup_{\delta\in B_p(0,R)} \int_0^1 \|\nabla f_j(x+t\delta) - \nabla f_c(x+t\delta)\|_q \, dt \;\leq\; \max_{y\in B_p(x,R)} \|\nabla f_j(y) - \nabla f_c(y)\|_q .$$

Thus we get the lower bound for the minimal norm of the change δ required to change the classifier decision from c to j,

$$\|\delta\|_p \geq \min \left\{ R, \frac{f_c(x) - f_j(x)}{\max_{y\in B_p(x,R)} \|\nabla f_j(y) - \nabla f_c(y)\|_q} \right\} := \alpha.$$

As we are interested in the worst case, we take the minimum over all $j \neq c$. Finally, the result holds for any fixed $R > 0$ so that we can maximize over R which yields the final result. $\square$

Note that the bound requires in the denominator a bound on the local Lipschitz constant of all cross terms $f_c - f_j$, which we call local cross-Lipschitz constant in the following. However, we do not require to have a global bound. The problem with a global bound is that the ideal robust classifier is basically piecewise constant on larger regions with sharp transitions between the classes. However, the global Lipschitz constant would then just be influenced by the sharp transition zones and would not yield a good bound, whereas the local bound can adapt to regions where the classifier is approximately constant and then yields good guarantees. In [24, 4] they suggest to study the global Lipschitz constant[1] of each f_j, $j = 1, \ldots, K$. A small global Lipschitz constant for all f_j implies a good bound as

$$\|\nabla f_j(y) - \nabla f_c(y)\|_q \le \|\nabla f_j(y)\|_q + \|\nabla f_c(y)\|_q , \tag{2}$$

but the converse does not hold. As discussed below it turns out that our local estimates are significantly better than the suggested global estimates which implies also better robustness guarantees. In turn we want to emphasize that our bound is tight, that is the bound is attained, for linear classifiers $f_j(x) = \langle w_j, x\rangle$, $j = 1, \ldots, K$. It holds

$$\|\delta\|_p = \min_{j \neq c} \frac{\langle w_c - w_j, x\rangle}{\|w_c - w_j\|_q}.$$

In Section 4 we refine this result for the case when the input is constrained to $[0, 1]^d$. In general, it is possible to integrate constraints on the input by simply doing the maximum over the intersection of $B_p(x, R)$ with the constraint set e.g. $[0, 1]^d$ for gray-scale images.

2.2 Evaluation of the Bound for Kernel Methods

Next, we discuss how the bound can be evaluated for different classifier models. For simplicity we restrict ourselves to the case $p = 2$ (which implies $q = 2$) and leave the other cases to future work. We consider the class of kernel methods, that is the classifier has the form

$$f_j(x) = \sum_{r=1}^{n} \alpha_{jr} k(x_r, x),$$

where $(x_r)_{r=1}^n$ are the n training points, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive definite kernel function and $\alpha \in \mathbb{R}^{K\times n}$ are the trained parameters e.g. of a SVM. The goal is to upper bound the term $\max_{y\in B_2(x,R)} \|\nabla f_j(y) - \nabla f_c(y)\|_2$ for this classifier model. A simple calculation shows

$$0 \le \|\nabla f_j(y) - \nabla f_c(y)\|_2^2 = \sum_{r,s=1}^{n} (\alpha_{jr} - \alpha_{cr})(\alpha_{js} - \alpha_{cs}) \langle \nabla_y k(x_r, y), \nabla_y k(x_s, y)\rangle \tag{3}$$

#How did this equation come?

It has been reported that kernel methods with a Gaussian kernel are robust to noise. Thus we specialize now to this class, that is $k(x, y) = e^{-\gamma\|x-y\|_2^2}$. In this case

$$\langle \nabla_y k(x_r, y), \nabla_y k(x_s, y)\rangle = 4\gamma^2 \langle y - x_r, y - x_s\rangle\, e^{-\gamma\|x_r-y\|_2^2} e^{-\gamma\|x_s-y\|_2^2}.$$

We will now derive lower and upper bounds on this term uniformly over $B_2(x, R)$ which allows us to derive the guarantee.

Lemma 2.1. *Let* $M = \min\left\{\frac{\|2x-x_r-x_s\|_2}{2}, R\right\}$, *then*

$$\max_{y\in B_2(x,R)} \langle y - x_r, y - x_s\rangle = \langle x - x_r, x - x_s\rangle + R\,\|2x - x_r - x_s\|_2 + R^2$$

$$\min_{y\in B_2(x,R)} \langle y - x_r, y - x_s\rangle = \langle x - x_r, x - x_s\rangle - M\,\|2x - x_r - x_s\|_2 + M^2$$

$$\max_{y\in B_2(x,R)} e^{-\gamma\|x_r-y\|_2^2} e^{-\gamma\|x_s-y\|_2^2} = e^{-\gamma\left(\|x-x_r\|_2^2+\|x-x_s\|_2^2-2M\|2x-x_r-x_s\|_2+2M^2\right)}$$

$$\min_{y\in B_2(x,R)} e^{-\gamma\|x_r-y\|_2^2} e^{-\gamma\|x_s-y\|_2^2} = e^{-\gamma\left(\|x-x_r\|_2^2+\|x-x_s\|_2^2+2R\|2x-x_r-x_s\|_2+2R^2\right)}$$

[1]The Lipschitz constant L wrt to p-norm of a piecewise continuously differentiable function is given as $L = \sup_{x\in\mathbb{R}^d} \|\nabla f(x)\|_q$. Then it holds, $|f(x) - f(y)| \le L\,\|x - y\|_p$.

Proof. For the first part we use

$$\begin{aligned}\max_{y\in B_2(x,R)} \langle y-x_r, y-x_s\rangle &= \max_{h\in B_2(0,R)} \langle x-x_r+h, x-x_s+h\rangle \\ &= \langle x-x_r, x-x_s\rangle + \max_{h\in B_2(0,R)} \langle h, 2x-x_r-x_s\rangle + \|h\|_2^2 \\ &= \langle x-x_r, x-x_s\rangle + R\,\|2x-x_r-x_s\|_2 + R^2,\end{aligned}$$

where the last equality follows by Cauchy-Schwarz and noting that equality is attained as we maximize over the Euclidean unit ball. For the second part we consider

$$\begin{aligned}\min_{y\in B_2(x,R)} \langle y-x_r, y-x_s\rangle &= \min_{h\in B_2(0,R)} \langle x-x_r+h, x-x_s+h\rangle \\ &= \langle x-x_r, x-x_s\rangle + \min_{h\in B_2(0,R)} \langle h, 2x-x_r-x_s\rangle + \|h\|_2^2 \\ &= \langle x-x_r, x-x_s\rangle + \min_{0\le\alpha\le R} -\alpha\,\|2x-x_r-x_s\|_2 + \alpha^2 \\ &= \langle x-x_r, x-x_s\rangle - \min\left\{\frac{\|2x-x_r-x_s\|_2}{2}, R\right\}\|2x-x_r-x_s\|_2 \\ &\quad + \left(\min\left\{\frac{\|2x-x_r-x_s\|_2}{2}, R\right\}\right)^2,\end{aligned}$$

where in the second step we have separated direction and norm of the vector, optimization over the direction yields with Cauchy-Schwarz the result. Finally, the constrained convex one-dimensional optimization problem can be solved explicitly as $\alpha = \min\{\frac{\|2x-x_r-x_s\|_2}{2}, R\}$. The proof of the other results follows analogously noting that

Donno how to apply constrained Convex optimisation

$$e^{-\gamma\|x+h-x_r\|_2^2} e^{-\gamma\|x+h-x_s\|_2^2} = e^{-\gamma\left(\|x-x_r\|_2^2+\|x-x_s\|_2^2+2\langle h,2x-x_r-x_s\rangle+2\|h\|_2^2\right)}.$$

□

Using this lemma it is easy to derive the final result.

Proposition 2.1. *Let* $\beta_r = \alpha_{jr} - \alpha_{cr}$, $r = 1, \dots, n$ *and define* $M = \min\left\{\frac{\|2x-x_r-x_s\|_2}{2}, R\right\}$ *and* $S = \|2x-x_r-x_s\|_2$. *Then*

$$\begin{aligned}&\max_{y\in B_2(x,R)} \|\nabla f_j(y) - \nabla f_c(y)\|_2 \le 2\gamma \\ &\Bigg(\sum_{\substack{r,s=1\\ \beta_r\beta_s\ge 0}}^{n} \beta_r\beta_s\Big[\max\{\langle x-x_r, x-x_s\rangle + RS + R^2,\, 0\}e^{-\gamma\left(\|x-x_r\|_2^2+\|x-x_s\|_2^2-2MS+2M^2\right)} \\ &\qquad + \min\{\langle x-x_r, x-x_s\rangle + RS + R^2,\, 0\}e^{-\gamma\left(\|x-x_r\|_2^2+\|x-x_s\|_2^2+2RS+2R^2\right)}\Big] \\ &+ \sum_{\substack{r,s=1\\ \beta_r\beta_s< 0}}^{n} \beta_r\beta_s\Big[\max\{\langle x-x_r, x-x_s\rangle - MS + M^2,\, 0\}e^{-\gamma\left(\|x-x_r\|_2^2+\|x-x_s\|_2^2+2RS+2R^2\right)} \\ &\qquad + \min\{\langle x-x_r, x-x_s\rangle - MS + M^2,\, 0\}e^{-\gamma\left(\|x-x_r\|_2^2+\|x-x_s\|_2^2-2MS+2M^2\right)}\Big]\Bigg)^{\frac{1}{2}}\end{aligned}$$

Proof. We bound each term in the sum in Equation 3 separately using that $ac \le bd$ if $b, c \ge 0$, $a \le b$ and $c \le d$ or $b \le 0$, $d \ge 0$, $a \le b$ and $c \ge d$, where c, d correspond to the exponential terms and a, b to the upper bounds of the inner product. Similarly, $ab \ge cd$ if $b, c \ge 0$, $a \ge b$ and $c \ge d$ or $a \le 0$, $d \ge 0$, $a \ge b$ and $c \le d$. The individual upper and lower bounds are taken from Lemma 2.1. □

While the bound leads to non-trivial estimates as seen in Section 5, the bound is not very tight. The reason is that the sum is bounded elementwise, which is quite pessimistic. We think that better bounds are possible but have to postpone this to future work.

2.3 Evaluation of the Bound for Neural Networks

We derive the bound for a neural network with one hidden layer. In principle, the technique we apply below can be used for arbitrary layers but the computational complexity increases rapidly. The problem is that in the directed network topology one has to consider almost each path separately to derive the bound. Let U be the number of hidden units and w, u are the weight matrices of the output resp. input layer. We assume that the activation function σ is continuously differentiable and assume that the derivative σ' is monotonically increasing. Our prototype activation function we have in mind and which we use later on in the experiment is the differentiable approximation, $\sigma_\alpha(x) = \frac{1}{\alpha}\log(1+e^{\alpha x})$ of the ReLU activation function $\sigma_{\text{ReLU}}(x) = \max\{0, x\}$. Note that $\lim_{\alpha\to\infty}\sigma_\alpha(x) = \sigma_{\text{ReLU}}(x)$ and $\sigma'_\alpha(x) = \frac{1}{1+e^{-\alpha x}}$. The output of the neural network can be written as

$$f_j(x) = \sum_{r=1}^{U} w_{jr}\,\sigma\Big(\sum_{s=1}^{d} u_{rs}x_s\Big), \quad j = 1, \ldots, K,$$

where for simplicity we omit any bias terms, but it is straightforward to consider also models with bias. A direct computation shows that

$$\left\|\nabla f_j(y) - \nabla f_c(y)\right\|_2^2 = \sum_{r,m=1}^{U} (w_{jr} - w_{cr})(w_{jm} - w_{cm})\sigma'(\langle u_r, y\rangle)\sigma'(\langle u_m, y\rangle)\sum_{l=1}^{d} u_{rl}u_{ml}, \quad (4)$$

where $u_r \in \mathbb{R}^d$ is the r-th row of the weight matrix $u \in \mathbb{R}^{U\times d}$. The resulting bound is given in the following proposition.

Proposition 2.2. *Let σ be a continuously differentiable activation function with σ' monotonically increasing. Define $\beta_{rm} = (w_{jr} - w_{cr})(w_{jm} - w_{cm})\sum_{l=1}^{d} u_{rl}u_{ml}$. Then*

$$\begin{aligned}\max_{y\in B_2(x,R)} &\left\|\nabla f_j(y) - \nabla f_c(y)\right\|_2\\ &\leq \Big[\sum_{r,m=1}^{U} \max\{\beta_{rm}, 0\}\sigma'\big(\langle u_r, x\rangle + R\left\|u_r\right\|_2\big)\sigma'\big(\langle u_m, x\rangle + R\left\|u_m\right\|_2\big)\\ &\qquad + \min\{\beta_{rm}, 0\}\sigma'\big(\langle u_r, x\rangle - R\left\|u_r\right\|_2\big)\sigma'\big(\langle u_m, x\rangle - R\left\|u_m\right\|_2\big)\Big]^{\frac{1}{2}}\end{aligned}$$

Proof. The proof is based on the fact that due the monotonicity of σ' and with Cauchy-Schwarz,

$$\max_{y\in B_2(x,R)}\sigma'(\langle u_r, y\rangle) = \max_{h\in B_2(0,R)}\sigma'(\langle u_r, x\rangle + \langle u_r, h\rangle) = \sigma'(\langle u_r, x\rangle + R\left\|u_r\right\|_2).$$

Similarly, one gets

$$\min_{y\in B_2(x,R)}\sigma'(\langle u_r, y\rangle) = \min_{h\in B_2(0,R)}\sigma'(\langle u_r, x\rangle + \langle u_r, h\rangle) = \sigma'(\langle u_r, x\rangle - R\left\|u_r\right\|_2).$$

The rest of the result follows by element-wise bounding the terms in the sum in Equation 4. □

As discussed above the global Lipschitz bounds of the individual classifier outputs, see (2), lead to an upper bound of our desired local cross-Lipschitz constant. In the experiments below our local bounds on the Lipschitz constant are up to 8 times smaller, than what one would achieve via the global Lipschitz bounds of [24]. This shows that their global approach is much too rough to get meaningful robustness guarantees.

3 The Cross-Lipschitz Regularization Functional

We have seen in Section 2 that if

$$\max_{j\neq c}\max_{y\in B_p(x,R)}\left\|\nabla f_c(y) - \nabla f_j(y)\right\|_q, \quad (5)$$

is small and $f_c(x) - f_j(x)$ is large, then we get good robustness guarantees. The latter property is typically already optimized in a multi-class loss function. We consider for all methods in this paper the cross-entropy loss so that the differences in the results only come from the chosen function class (kernel methods versus neural networks) and the chosen regularization functional. The cross-entropy loss $L : \{1, \ldots, K\} \times \mathbb{R}^K \to \mathbb{R}$ is given as

$$L(y, f(x)) = -\log\Big(\frac{e^{f_y(x)}}{\sum_{k=1}^{K} e^{f_k(x)}}\Big) = \log\Big(1 + \sum_{k \neq y}^{K} e^{f_k(x) - f_y(x)}\Big).$$

In the latter formulation it becomes apparent that the loss tries to make the difference $f_y(x) - f_k(x)$ as large as possible for all $k = 1, \ldots, K$.

As our goal are good robustness guarantees it is natural to consider a proxy of the quantity in (5) for regularization. We define the **Cross-Lipschitz Regularization** functional as

$$\Omega(f) = \frac{1}{nK^2} \sum_{i=1}^{n} \sum_{l,m=1}^{K} \|\nabla f_l(x_i) - \nabla f_m(x_i)\|_2^2, \tag{6}$$

where the $(x_i)_{i=1}^n$ are the training points. The goal of this regularization functional is to make the *differences* of the classifier functions at the data points as constant as possible. In total by minimizing

$$\frac{1}{n} \sum_{i=1}^{n} L\big(y_i, f(x_i)\big) + \lambda \Omega(f), \tag{7}$$

over some function class we thus try to maximize $f_c(x_i) - f_j(x_i)$ and at the same time keep $\|\nabla f_l(x_i) - \nabla f_m(x_i)\|_2^2$ small uniformly over all classes. This automatically enforces robustness of the resulting classifier. It is important to note that this regularization functional is coherent with the loss as it shares the same degrees of freedom, that is adding the same function g to all outputs: $f_j'(x) = f_j(x) + g(x)$ leaves loss and regularization functional invariant. This is the main difference to [4], where they enforce the global Lipschitz constant to be smaller than one.

3.1 Cross-Lipschitz Regularization in Kernel Methods

In kernel methods one uses typically the regularization functional induced by the kernel which is given as the squared norm of the function, $f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$, in the corresponding reproducing kernel Hilbert space H_k, $\|f\|_{H_k}^2 = \sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j)$. In particular, for translation invariant kernels one can make directly a connection to penalization of derivatives of the function f via the Fourier transform, see [20]. However, penalizing higher-order derivatives is irrelevant for achieving robustness. Given the kernel expansion of f, one can write the Cross-Lipschitz regularization function as

$$\Omega(f) = \frac{1}{nK^2} \sum_{i,j=1}^{n} \sum_{l,m=1}^{K} \sum_{r,s=1}^{n} (\alpha_{lr} - \alpha_{mr})(\alpha_{ls} - \alpha_{ms}) \left\langle \nabla_y k(x_r, x_i), \nabla_y k(x_s, x_i) \right\rangle$$

Ω is convex in $\alpha \in \mathbb{R}^{K \times n}$ as $k'(x_r, x_s) = \langle \nabla_y k(x_r, x_i), \nabla_y k(x_s, x_i) \rangle$ is a positive definite kernel for any x_i and with the convex cross-entropy loss the learning problem in (7) is convex.

3.2 Cross-Lipschitz Regularization in Neural Networks

The standard way to regularize neural networks is *weight decay*; that is, the squared Euclidean norm of all weights is added to the objective. More recently dropout [22], which can be seen as a form of stochastic regularization, has been introduced. Dropout can also be interpreted as a form of regularization of the weights [22, 10]. It is interesting to note that classical regularization functionals which penalize derivatives of the resulting classifier function are not typically used in deep learning, but see [6, 11]. As noted above we restrict ourselves to one hidden layer neural networks to simplify notation, that is, $f_j(x) = \sum_{r=1}^{U} w_{jr}\, \sigma\big(\sum_{s=1}^{d} u_{rs} x_s\big), \quad j = 1, \ldots, K$. Then we can write the Cross-Lipschitz regularization as

$$\Omega(f) = \frac{2}{nK^2} \sum_{r,s=1}^{U} \Big(\sum_{l=1}^{K} w_{lr} w_{ls} - \sum_{l=1}^{K} w_{lr} \sum_{m=1}^{K} w_{ms} \Big) \sum_{i,j=1}^{n} \sigma'(\langle u_r, x_i \rangle) \sigma'(\langle u_s, x_i \rangle) \sum_{l=1}^{d} u_{rl} u_{sl}$$

which leads to an expression which can be fast evaluated using vectorization. Obviously, one can also implement the Cross-Lipschitz Regularization also for all standard deep networks.

4 Box Constrained Adversarial Sample Generation

The main emphasis of this paper are robustness guarantees without resorting to particular ways how to generate adversarial samples. On the other hand while Theorem 2.1 gives lower bounds on the required input transformation, efficient ways to approximately solve the adversarial sample generation in (1) are helpful to get upper bounds on the required change. Upper bounds allow us to check how tight our derived lower bounds are. As all of our experiments will be concerned with images, it is reasonable that our adversarial samples are also images. However, up to our knowledge, the current main techniques to generate adversarial samples [7, 12, 19] integrate box constraints by clipping the results to $[0,1]^d$. We provide in the following fast algorithms to generate adversarial samples which lie in $[0,1]^d$. The strategy is similar to [12], where they use a linear approximation of the classifier to derive adversarial samples with respect to different norms. Formally,

$$f_j(x+\delta) \approx f_j(x) + \langle \nabla f_j(x), \delta \rangle, \quad j = 1, \ldots, K.$$

Assuming that the linear approximation holds, the optimization problem (1) integrating box constraints for changing class c into j becomes

$$\begin{aligned} \min_{\delta \in \mathbb{R}^d} \quad & \|\delta\|_p \\ \text{sbj. to:} \quad & f_j(x) - f_c(x) \geq \langle \nabla f_c(x) - \nabla f_j(x), \delta \rangle \\ & 0 \leq x_j + \delta_j \leq 1 \end{aligned} \tag{8}$$

In order to get the minimal adversarial sample we have to solve this for all $j \neq c$ and take the one with minimal $\|\delta\|_p$. This yields the minimal adversarial change for linear classiifers. Note that (8) is a convex optimization problem, which can be reduced to a one-parameter problem in the dual. This allows to derive the following result (proofs and algorithms are in the supplement).

Proposition 4.1. *Let $p \in \{1, 2, \infty\}$, then (8) can be solved in $O(d \log d)$ time.*

We start with problem for $p = 2$ which is given as:

$$\begin{aligned} \min_{\delta \in \mathbb{R}^d} \quad & \|\delta\|_2 \\ \text{sbj. to:} \quad & f_j(x) - f_c(x) \geq \langle \nabla f_c(x) - \nabla f_j(x), \delta \rangle \\ & 0 \leq x_j + \delta_j \leq 1 \end{aligned} \tag{9}$$

Lemma 4.1. *Let $m = \arg\max_j f_j(x)$ and define $v = \nabla f_m(x) - \nabla f_j(x)$ and $0 > c = f_j(x) - f_m(x)$. If a solution of problem (9) exists, then it is given as*

$$\delta_r = \begin{cases} -\lambda v_r, & \text{if } -x_r \leq -\lambda v_r \leq 1 - x_r, \\ 1 - x_r, & \text{if } -\lambda v_r \geq 1 - x_r, \\ -x_r, & \text{if } -x_r \geq -\lambda v_r. \end{cases}$$

The optimal $\lambda \geq 0$ can be obtained by solving

$$c = \langle v, \delta \rangle = -\lambda \sum_{-x_r \leq -\lambda v_r \leq 1 - x_r} v_r^2 + \sum_{-\lambda v_r > 1 - x_r} v_r (1 - x_r) - \sum_{-\lambda v_r < -x_r} v_r x_r.$$

If Problem 9 is infeasible, then this equation has no solution for $\lambda \geq 0$. In both the feasible and infeasible case the solution can be found in $O(d \log d)$. The algorithm is given in Algorithm 1.

Proof. The Lagrangian is given by

$$L(\delta,\lambda,\alpha,\beta)=\frac{1}{2}\left\|\delta\right\|_2^2+\lambda\big(\left\langle v,\delta\right\rangle-c\big)+\left\langle \alpha, x+\delta-\mathbf{1}\right\rangle-\left\langle \beta, x+\delta\right\rangle.$$

#How did this equation come?

The KKT conditions become

$$\begin{aligned}\delta+\lambda v+\alpha-\beta&=0\\ \alpha_r(x_r+\delta_r-1)&=0,\quad \forall r=1,\dots,d\\ \beta_r(x_r+\delta_r)&=0,\quad \forall r=1,\dots,d\\ \lambda(\langle v,\delta\rangle-c)&=0\\ \alpha_r&\geq 0,\quad \forall r=1,\dots,d\\ \beta_r&\geq 0,\quad \forall r=1,\dots,d\\ \lambda&\geq 0.\end{aligned}$$

We deduce that if $\beta_r>0$ then $\alpha_r=0$ which implies

$$\delta_r=-x_r=-\lambda v_r+\beta_r \quad\Longrightarrow\quad \beta_r=\max\{0,-x_r+\lambda v_r\}.$$

Similarly, if $\alpha_r>0$ then $\beta_r=0$ which implies

$$\delta_r=1-x_r=-\lambda v_r-\alpha_r \quad\Longrightarrow\quad \alpha_r=\max\{0,x_r-1-\lambda v_r\}.$$

It follows

$$\delta_r=\begin{cases}-\lambda v_r & \text{if } -x_r<-\lambda v_r<1-x_r\\ 1-x_r & \text{if } -\lambda v_r>1-x_r\\ -x_r & \text{if } -\lambda v_r<-x_r\end{cases}.$$

We can determine λ by inspecting $\langle v,\delta\rangle$ which is given as

$$\langle v,\delta\rangle=-\lambda\sum_{-x_r\leq-\lambda v_r\leq 1-x_r}v_r^2+\sum_{-\lambda v_r>1-x_r}v_r(1-x_r)-\sum_{-\lambda v_r<-x_r}v_r x_r.$$

Note that $\lambda\geq 0$ and $1-x_r\geq 0$ and thus $-\lambda v_r>1-x_r$ implies $v_r<0$, thus $v_r(1-x_r)\leq 0$ and similarly $-\lambda v_r<-x_r$ implies $v_r>0$ and thus also $-v_r x_r<0$. Note that the term $\langle v,\delta\rangle$ is monotonically decreasing as λ is increasing. Thus one can sort $\max\{\frac{x_r-1}{v_r},\frac{x_r}{v_r}\}$ in increasing order and they represent the thresholds when the summation changes. Then we compute $\langle v,\delta\rangle$ for all of these thresholds and determine the largest threshold λ^* such that $\langle v,\delta\rangle\leq c$. This fixes the index sets of all sums. Then we determine λ by

$$\lambda=\frac{\sum\limits_{-\lambda^* v_r>1-x_r}v_r(1-x_r)-\sum\limits_{-\lambda^* v_r<-x_r}v_r x_r-c}{\sum\limits_{-x_r\leq-\lambda^* v_r\leq 1-x_r}v_r^2}.$$

In total sorting takes time $O(d\log d)$ and solving for λ^* has complexity $O(d)$. □

Next we consider the case $p=1$.

$$\begin{aligned}&\min\nolimits_{\delta\in\mathbb{R}^d}\ \left\|\delta\right\|_1 \\ &\text{sbj. to: } f_j(x)-f_c(x)\geq\langle\nabla f_c(x)-\nabla f_j(x),\delta\rangle\\ &\qquad\quad 0\leq x_j+\delta_j\leq 1\end{aligned}\tag{10}$$

Lemma 4.2. *Let $m=\arg\max\limits_j f_j(x)$ and define $v=\nabla f_m(x)-\nabla f_j(x)$ and $c=f_j(x)-f_m(x)<0$, then the solution of the problem* (10) *can be found by Algorithm 2.*

Proof. The result is basically obvious but we derive it formally. First of all we rewrite (10) as a linear program.

$$\begin{aligned}&\min_{\delta\in\mathbb{R}^d}\ \sum_{i=1}^d t_i\\ &\text{sbj. to: } c\geq\langle v,\delta\rangle\\ &\qquad\quad -x_j\leq\delta_j\leq 1-x_j\\ &\qquad\quad -t_i\leq\delta_i\leq t_i\\ &\qquad\quad t_i\geq 0\end{aligned}\tag{11}$$

Algorithm 1 Computation of box-constrained adversarial samples wrt to $\|\cdot\|_2$-norm.

```
INPUT: $c = f_j(x) - f_m(x)$ and $v = \nabla f_m(x) - \nabla f_j(x)$ (j, desired class, m original class)
sort $\gamma_r = \max\{\frac{x_r - 1}{v_r}, \frac{x_r}{v_r}\}$ in increasing order $\pi$
s=0; $\rho = 0$
while $\rho > c$ do
  $s \leftarrow s + 1$
  compute $\rho = \langle v, \delta(\gamma_{\pi_s}) \rangle$, where $\delta(\lambda)$ is the function defined in Lemma 4.1
end while
if $\rho \leq c$ then
  compute $I_m = \{r \mid -x_r \leq -\gamma_{\pi_{s-1})} v_r \leq 1 - x_r\}$,
  compute $I_u = \{r \mid -\gamma_{\pi_{s-1}} v_r > 1 - x_r\}$,
  compute $I_l = \{r \mid -\gamma_{\pi_{s-1}} v_r < -x_r\}$
  $\lambda = \frac{\sum\limits_{r \in I_u} v_r(1-x_r) - \sum\limits_{r \in I_l} v_r x_r - c}{\sum\limits_{r \in I_m} v_r^2}$.
  $\delta = \max\{-x_r, \min\{-\lambda v_r, 1 - x_r\}\}$
else
  Problem has no feasible solution
end if
```

Algorithm 2 Computation of box-constrained adversarial samples wrt to $\|\cdot\|_1$-norm.

```
INPUT: $c = f_j(x) - f_m(x)$ and $v = \nabla f_m(x) - \nabla f_j(x)$ (j, desired class, m original class)
sort $|v_i|$ in decreasing order
s:=0, g:=0,
while $g > c$ do
  $s \leftarrow s + 1$
  if $v_{\pi_s} > 0$ then
    $\delta_{\pi_s} = -x_{\pi_s}$
  else
    $\delta_{\pi_s} = 1 - x_{\pi_s}$
  end if
  $g \leftarrow g + \delta_{\pi_s} v_{\pi_s}$
end while
if $g \leq c$ then
  $g \leftarrow g - v_{\pi_s} \delta_{\pi_s}$
  $\delta_{\pi_s} \leftarrow \frac{c-g}{v_{\pi_s}}$
else
  Problem has no feasible solution
end if
```

The Lagrangian of this problem is

$$\begin{aligned} L(\delta,t,\alpha,\beta,\gamma,\theta,\kappa,\lambda) &= \langle t,\mathbf{1}\rangle + \lambda(\langle v,\delta\rangle - c) - \langle \alpha,\delta+x\rangle + \langle \beta,\delta-\mathbf{1}+x\rangle - \langle \gamma,\delta+t\rangle + \langle \theta,\delta-t\rangle - \langle \kappa,t\rangle \\ &= \langle t,\mathbf{1}-\gamma-\theta-\kappa\rangle + \langle \delta,\beta-\alpha+\lambda v+\theta-\gamma\rangle + \langle \beta-\alpha,x\rangle - \langle \beta,\mathbf{1}\rangle - \lambda c \end{aligned}$$

Minimization of the Lagrangian over t resp. δ leads only to a non-trivial result if

$$\begin{aligned} \mathbf{1}-\gamma-\theta-\kappa &= 0 \\ \beta-\alpha+\lambda v+\theta-\gamma &= 0 \end{aligned}$$

We get the dual problem

$$\begin{aligned} \max_{\alpha,\beta,\theta,\gamma,\kappa,\lambda}\ & \langle \beta-\alpha,x\rangle - \langle \beta,\mathbf{1}\rangle - \lambda c && (12) \\ \text{sbj. to: }& \mathbf{1}-\gamma-\theta-\kappa = 0 \\ & \beta-\alpha+\lambda v+\theta-\gamma = 0 \\ & \alpha\geq 0,\ \beta\geq 0,\ \theta\geq 0,\ \gamma\geq 0,\ \kappa\geq 0,\ \lambda\geq 0 && (13) \end{aligned}$$

Using the equalities we can now simplify the problem by replacing α and using the fact that κ is not part of the objective, the positivity just induces an additional constraint. We get

$$\alpha = \beta+\lambda v+\theta-\gamma.$$

Plugging this into the problem (16) we get

$$\begin{aligned} \max_{\beta,\theta,\gamma,\lambda}\ & -\langle \lambda v+\theta-\gamma,x\rangle - \langle \beta,\mathbf{1}\rangle - \lambda c && (14) \\ \text{sbj. to: }& \mathbf{1}-\gamma-\theta \geq 0 \\ & \beta+\lambda v+\theta-\gamma \geq 0 \\ & \beta\geq 0,\ \theta\geq 0,\ \gamma\geq 0,\ \lambda\geq 0 && (15) \end{aligned}$$

We get the constraint $\beta \geq \max\{0, -\lambda v-\theta+\gamma\}$ (all the inequalities and functions are taken here componentwise) and thus we can explicitly maximize over β

$$\begin{aligned} \max_{\theta,\gamma,\lambda}\ & -\langle \lambda v+\theta-\gamma,x\rangle - \sum_{i=1}^{d}\max\{0,-\lambda v_i-\theta_i+\gamma_i\} - \lambda c && (16) \\ \text{sbj. to: }& \gamma+\theta \leq \mathbf{1} \\ & \theta\geq 0,\ \gamma\geq 0,\ \lambda\geq 0 && (17) \end{aligned}$$

As $0 \leq x_i \leq 1$ it holds for all $\theta\geq 0, \gamma\geq 0$

$$\langle -\lambda v-\theta+\gamma,x\rangle - \sum_{i=1}^{d}\max\{0,-\lambda v_i-\theta_i+\gamma_i\} \leq 0.$$

The maximum is attained if $\gamma_i-\theta_i-\lambda v_i = 0$ resp. with the constraints on γ_i,θ_i this is equivalent to $-1\leq \lambda v_i \leq 1$. Suppose that $\lambda v_i > 1$ then the maximum is attained for $\gamma_i = 1$ and $\theta_i = 0$, and for $\lambda v_i < -1$ the maximum is attained for $\gamma_i = 0$ and $\theta_i = 1$. Thus by solving explicitly for θ and γ we obtain

$$\begin{aligned} \max_{\lambda}\ & \sum_{\lambda v_i>1}(1-\lambda_i v_i)x_i + \sum_{\lambda v_i<-1}(1-\lambda_i v)(x_i-1) - \lambda c && (18) \\ \text{sbj. to: }& \lambda\geq 0 && (19) \end{aligned}$$

Note that the first two terms are decreasing with λ and the last term is increasing with λ. Let λ^* be the optimum, then we have the following characterization

$$\begin{aligned} -1<\lambda v_i<1 \quad&\Longrightarrow\quad \gamma_i+\theta_i<1 \quad\Longrightarrow \kappa_i>0 \quad\Longrightarrow t_i=0 \Longrightarrow \delta_i=0 \\ \lambda v_i>1 \quad&\Longrightarrow\quad \gamma_i=1,\theta_i=0,\beta_i=0,\alpha_i>0 \quad\Longrightarrow\quad \delta_i=-x_i, \\ \lambda v_i<-1 \quad&\Longrightarrow\quad \gamma_i=0,\theta_i=1,\beta_i>0 \quad\Longrightarrow\quad \delta_i=1-x_i, \end{aligned}$$

The cases $|\lambda v_i| = 1$ are undetermined but given that $\lambda > 0$ the remaining values can be fixed by solving for $c = \langle v,\delta\rangle$. The time complexity is again determined by the initial sorting step of $O(d\log d)$. The following linear scan requires $O(d)$. □

Finally, we consider the case $p = \infty$.

$$\begin{aligned}\min_{\delta \in \mathbb{R}^d} \ & \|\delta\|_\infty \\ \text{sbj. to: } & f_j(x) - f_c(x) \geq \langle \nabla f_c(x) - \nabla f_j(x), \delta \rangle \\ & 0 \leq x_j + \delta_j \leq 1\end{aligned} \tag{20}$$

Lemma 4.3. *Let $m = \arg\max_j f_j(x)$ and define $v = \nabla f_m(x) - \nabla f_j(x)$ and $c = f_j(x) - f_m(x) < 0$, then the solution of (20) can be found by solving for $t \geq 0$,*

$$\sum_{v_i > 0} v_i \max\{-t, -x_r\} + \sum_{v_i < 0} v_i \min\{t, 1 - x_r\} = c.$$

which is done in Algorithm 3.

Proof. We can rewrite the optimization problem into a linear program

$$\begin{aligned}\min_{t \in \mathbb{R}, \delta \in \mathbb{R}^d} \ & t \\ \text{sbj. to: } & c \geq \langle v, \delta \rangle \\ & -x_j \leq \delta_j \leq 1 - x_j, \quad j = 1, \ldots, d \\ & -t \leq \delta_j \leq t, \quad j = 1, \ldots, d \\ & t \geq 0\end{aligned} \tag{21, 22}$$

Thus we have $\max\{-t, -x_r\} \leq \delta_r \leq \min\{t, 1 - x_r\}$ for $r = 1, \ldots, d$. Then it holds

$$\begin{aligned}\langle v, \delta \rangle &\geq \sum_{v_r > 0} v_r \max\{-t, -x_r\} + \sum_{v_r < 0} v_r \min\{t, 1 - x_r\} \\ &= -\sum_{v_r > 0, t \geq x_r} v_r x_r - t \sum_{v_r > 0, t < x_r} v_r + t \sum_{v_r < 0, t < 1 - x_r} v_r + \sum_{v_r < 0, t \geq 1 - x_r} v_r (1 - x_r),\end{aligned}$$

and the lower bound can be attained if $\delta_r = \min\{t, 1 - x_r\}$ for $v_r < 0$ and $\delta_r = \max\{-t, -x_r\}$ for $v_r > 0$, $\delta_r = 0$ if $v_r = 0$. Note that both terms are monotonically decreasing with t. The algorithm 3 has complexity $O(d \log d)$ due to the initial sorting step followed by steps of complexity $O(d)$. □

For nonlinear classifiers a change of the decision is not guaranteed and thus we use later on a binary search with a variable c instead of $f_c(x) - f_j(x)$.

5 Experiments

The goal of the experiments is the evaluation of the robustness of the resulting classifiers and not necessarily state-of-the-art results in terms of test error. In all cases we compute the robustness guarantees from Theorem 2.1 (lower bound on the norm of the minimal change required to change the classifier decision), where we optimize over R using binary search, and adversarial samples with the algorithm for the 2-norm from Section 4 (upper bound on the norm of the minimal change required to change the classifier decision), where we do a binary search in the classifier output difference in order to find a point on the decision boundary. Additional experiments can be found in the supplementary material.

Kernel methods: We optimize the cross-entropy loss once with the standard regularization (Kernel-LogReg) and with Cross-Lipschitz regularization (Kernel-CL). Both are convex optimization problems and we use L-BFGS to solve them. We use the Gaussian kernel $k(x, y) = e^{-\gamma \|x-y\|^2}$ where $\gamma = \frac{\alpha}{\rho^2_{\text{KNN40}}}$ and ρ_{KNN40} is the mean of the 40 nearest neighbor distances on the training set and $\alpha \in \{0.5, 1, 2, 4\}$. We show the results for MNIST (60000 training and 10000 test samples). However, we have checked that parameter selection using a subset of 50000 images from the training set and evaluating on the rest yields indeed the parameters which give the best test errors when trained on the full set. The

#https://towardsdatascience.com/limited-memory-broyden-fletcher-goldfarb-shanno-algorithm-in-ml-net-118dec066ba

Algorithm 3 Computation of box-constrained adversarial samples wrt to $\|\cdot\|_\infty$-norm.

```
INPUT: $c = f_j(x) - f_m(x)$ and $v = \nabla f_m(x) - \nabla f_j(x)$ (j, desired class, m original class)
$d_+ := |\{l \,|\, v_l > 0\}|$ and $d_- := |\{l \,|\, v_l < 0\}|$
sort $\{x_l \,|\, v_l > 0\}$ in increasing order $\pi$, sort $\{1 - x_l \,|\, v_l < 0\}$ in increasing order $\rho$
s:=1, r:=1, g:=0, t:=0, $\kappa_+ = \sum_{v_l > 0} v_l$, $\kappa_- = \sum_{v_l < 0} v_l$, $\gamma_+ = \gamma_- = 0$.
while $g > c$ AND ($s \leq d_+$ OR $t \leq d_-$) do
  if $x_{\pi_s} < 1 - x_{\rho_r}$ then
    $t = x_{\pi_s}$,   $\kappa_+ \leftarrow \kappa_+ - v_{\pi_s}$,   $\gamma_+ \leftarrow \gamma_+ - v_{\pi_s} x_{\pi_s}$,
    $s \leftarrow s + 1$
  else
    $t = 1 - x_{\rho_r}$,   $\kappa_- \leftarrow \kappa_- - v_{\rho_r}$,   $\gamma_- \leftarrow \gamma_- - v_{\rho_r}(1 - x_{\rho_r})$,
    $t \leftarrow t + 1$
  end if
  $g = \gamma_+ + \gamma_- + t\,(\kappa_+ - \kappa_-)$
end while
if $g \leq c$ then
  undo last step
  $t = (c - \gamma_+ - \gamma_-)/(\kappa_- - \kappa_+)$
  compute $\delta_r = \begin{cases} \min\{t, 1 - x_r\} & \text{for } v_r > 0 \\ \max\{-t, -x_r\} & \text{for } v_r < 0 \end{cases}$
else
  Problem has no feasible solution
end if
```

#Didnt understand?

regularization parameter is chosen in $\lambda \in \{10^{-k} | k \in \{5, 6, 7, 8\}\}$ for Kernel-SVM and $\lambda \in \{10^{-k} \,|\, k \in \{0, 1, 2, 3\}\}$ for our Kernel-CL. The results of the optimal parameters are given in the following table and the performance of all parameters is shown in Figure 1. Note that due to the high computational complexity we could evaluate the robustness guarantees only for the optimal parameters.

	test error	avg. $\|\cdot\|_2$ adv. samples	avg.$\|\cdot\|_2$ rob. guar.
No Reg. ($\lambda = 0$)	2.23%	2.39	0.037
K-SVM	1.48%	1.91	0.058
K-CL	1.44%	3.12	0.045

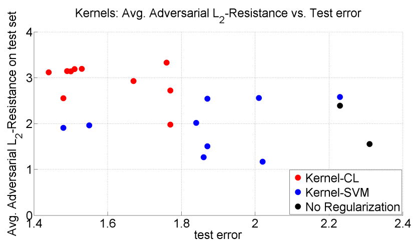


Figure 1: Kernel Methods: Cross-Lipschitz regularization achieves both better test error and robustness against adversarial samples (upper bounds, larger is better) compared to the standard regularization. The robustness guarantee is weaker than for neural networks but this is most likely due to the relatively loose bound.

#somehting that can be improved ig

Neural Networks: Before we demonstrate how upper and lower bounds improve using cross-Lipschitz regularization, we first want to highlight the importance of the usage of the local cross-Lipschitz constant in Theorem 2.1 for our robustness guarantee.

Local versus global Cross-Lipschitz constant: While no robustness guarantee has been proven before, it has been discussed in [24] that penalization of the global Lipschitz constant should improve robustness, see also [4]. For that purpose they derive the Lipschitz constants of several different layers and use the fact that the Lipschitz constant of a composition of functions is upper bounded by the product of the Lipschitz constants of the functions. In analogy, this would mean that the term $\sup_{y \in B(x,R)} \|\nabla f_c(y) - \nabla f_j(y)\|_2$, which we have upper bounded in Proposition 2.2, in the denominator in Theorem 2.1 could be replaced[2] by the global Lipschitz constant of $g(x) := f_c(x) - f_j(x)$. which is given as $\sup_{y \in \mathbb{R}^d} \|\nabla g(x)\|_2 = \sup_{x \neq y} \frac{|g(x) - g(y)|}{\|x - y\|_2}$. We have with $\|U\|_{2,2}$ being the largest singular value

[2]Note that then the optimization of R in Theorem 2.1 would be unnecessary.

MNIST (plain)				CIFAR10 (plain)			
None	Dropout	Weight Dec.	Cross Lip.	None	Dropout	Weight Dec.	Cross Lip.
0.69	0.48	0.68	0.21	0.22	0.13	0.24	0.17

Table 1: We show the average ratio $\frac{\alpha_{\text{global}}}{\alpha_{\text{local}}}$ of the robustness guarantees $\alpha_{\text{global}}, \alpha_{\text{local}}$ from Theorem 2.1 on the test data for MNIST and CIFAR10 and different regularizers. The guarantees using the local Cross-Lipschitz constant are up to eight times better than with the global one.

of U,

$$\begin{aligned}|g(x) - g(y)| = \langle w_c - w_j, \sigma(Ux) - \sigma(Uy)\rangle &\leq \|w_c - w_j\|_2 \|\sigma(Ux) - \sigma(Uy)\|_2 \\ &\leq \|w_c - w_j\|_2 \|U(x-y)\|_2 \leq \|w_c - w_j\|_2 \|U\|_{2,2} \|x-y\|_2 ,\end{aligned}$$

where we used that σ is contractive as $\sigma'(z) = \frac{1}{1+e^{-\alpha z}}$ and thus we get

$$\sup_{y \in \mathbb{R}^d} \|\nabla f_c(x) - \nabla f_j(x)\|_2 \leq \|w_c - w_j\|_2 \|U\|_{2,2} .$$

The advantage is clearly that this global Cross-Lipschitz constant can just be computed once and by using it in Theorem 2.1 one can evaluate the guarantees very quickly. However, it turns out that one gets significantly better robustness guarantees by using the local Cross-Lipschitz constant in terms of the bound derived in Proposition 2.2 instead of the just derived global Lipschitz constant. Note that the optimization over R in Theorem 2.1 is done using a binary search, noting that the bound of the local Lipschitz constant in Proposition 2.2 is monotonically decreasing in R. We have the following comparison in Table 1. We want to highlight that the robustness guarantee with the global Cross-Lipschitz constant was *always* worse than when using the local Cross-Lipschitz constant across all regularizers and data sets. Table 1 shows that the guarantees using the local Cross-Lipschitz can be up to eight times better than for the global one. As these are just one hidden layer networks, it is obvious that robustness guarantees for deep neural networks based on the global Lipschitz constants will be too coarse to be useful.

Experiments: We use a one hidden layer network with 1024 hidden units and the softplus activation function with $\alpha = 10$. Thus the resulting classifier is continuously differentiable. We compare three different regularization techniques: weight decay, dropout and our Cross-Lipschitz regularization. Training is done with SGD. For each method we have adapted the learning rate (two per method) and regularization parameters (4 per method) so that all methods achieve good performance. We do experiments for MNIST and CIFAR10 in three settings: plain, data augmentation and adversarial training. The exact settings of the parameters and the augmentation techniques are described below.The results for MNIST are shown in Figure 2 and the results for CIFAR10 are in Figure 3.For MNIST there is a clear trend that our Cross-Lipschitz regularization improves the robustness of the resulting classifier while having competitive resp. better test error. It is surprising that data augmentation does not lead to more robust models. However, adversarial training improves the guarantees as well as adversarial resistance. For CIFAR10 the picture is mixed, our CL-Regularization performs well for the augmented task in test error and upper bounds but is not significantly better in the robustness guarantees. The problem might be that the overall bad performance due to the simple model is preventing a better behavior. Data augmentation leads to better test error but the robustness properties (upper and lower bounds) are basically unchanged. Adversarial training slightly improves performance compared to the plain setting and improves upper and lower bounds in terms of robustness. We want to highlight that our guarantees (lower bounds) and the upper bounds from the adversarial samples are not too far away.

Stohastic gradient descent

For MNIST (all settings) the learning rate is for all methods chosen from $\{0.2, 0.5\}$. The regularization parameters for weight decay are chosen from $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, for Cross-Lipschitz from $\{10^{-5}, 10^{-4}, 5 * 10^{-4}, 10^{-3}\}$ and the dropout probabilities are taken from $\{0.4, 0.5, 0.6, 0.7\}$. For CIFAR10 the learning rate is for all methods chosen from $\{0.04, 0.1\}$, the regularization parameters for weight decay and Cross-Lipschitz are $\{10^{-5}, 10^{-4}, 5 * 10^{-4}, 10^{-3}\}$ and dropout probabilities are taken from $\{0.5, 0.6, 0.7, 0.8\}$. For CIFAR10 with data augmentation we choose the learning rate for all methods from $\{0.04, 0.1\}$, the regularization parameters for weight decay are $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$ and

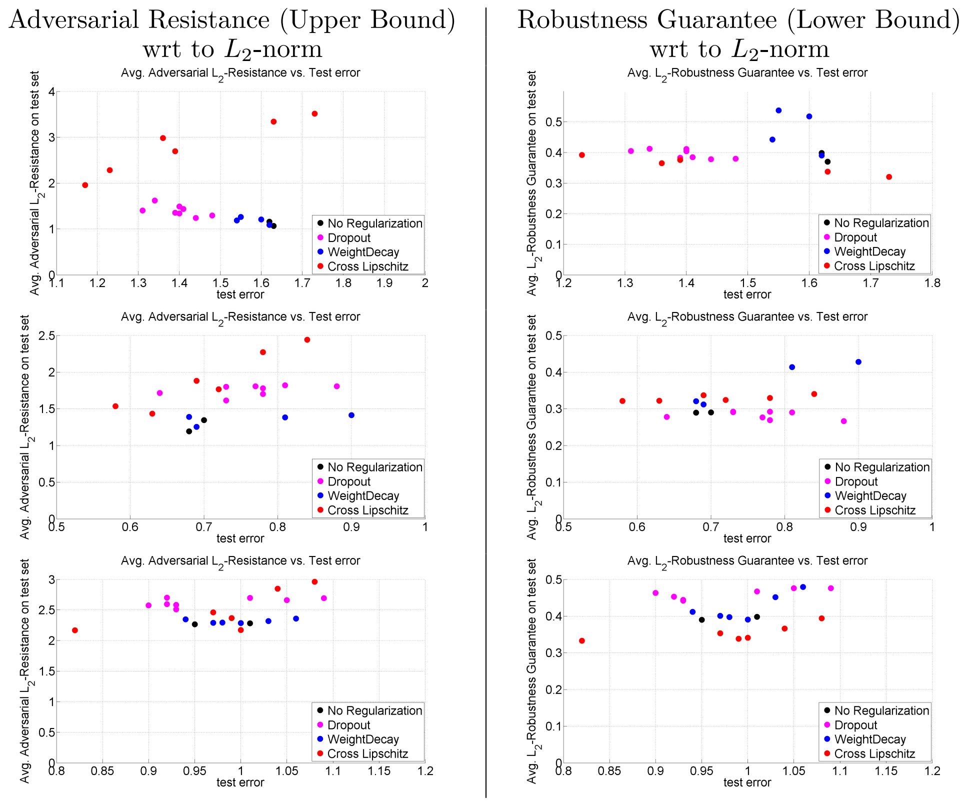


Figure 2: Neural Networks, Left: Adversarial resistance wrt to L_2-norm on MNIST. **Right:** Average robustness guarantee wrt to L_2-norm on MNIST for different neural networks (one hidden layer, 1024 HU) and hyperparameters. The Cross-Lipschitz regularization leads to better robustness with similar or better prediction performance. **Top row:** plain MNIST, **Middle:** Data Augmentation, **Bottom:** Adv. Training

for Cross-Lipschitz $\{10^{-5}, 10^{-4}, 5 * 10^{-4}, 10^{-3}\}$ and the dropout probabilities are taken from $\{0.5, 0.6, 0.7, 0.8\}$. Data augmentation for MNIST means that we apply random rotations in the angle $[-\frac{\pi}{20}, \frac{\pi}{20}]$ and random crop from 28x28 to 24x24. For CIFAR-10 we apply the same and additionally we mirror the image (left to right) with probability 0.5 and apply random brightness $[-0.1, 0.1]$ and random contrast change $[0.6, 1.4]$. In each substep we ensure that we get an image in $[0, 1]^d$ by clipping. We implemented adversarial training by generating adversarial samples wrt to the infinity norm with the code from Section 4 and replaced 50% of each batch as adversarial samples. Finally, we use for SGD batchsize 64 in all experiments.

Illustration of adversarial samples: we take one test image from MNIST and apply the adversarial generation from Section 4 wrt to the 2-norm to generate the adversarial samples for the different kernel methods and neural networks (plain setting), where we use for each method the parameters leading to best test performance. All classifiers change their originally correct decision to a “wrong” one. It is interesting to note that for Cross-Lipschitz regularization (both kernel method and neural network) the “adversarial” sample is really at the decision boundary between 1 and 8 (as predicted) and thus the new decision is actually correct. This effect is strongest for our Kernel-CL, which also requires the strongest modification to generate the adversarial sample. The situation is different for neural networks, where the classifiers obtained from the two standard regularization techniques are still vulnerable, as the adversarial sample is still clearly a 1 for dropout and weight decay. We show further examples below.

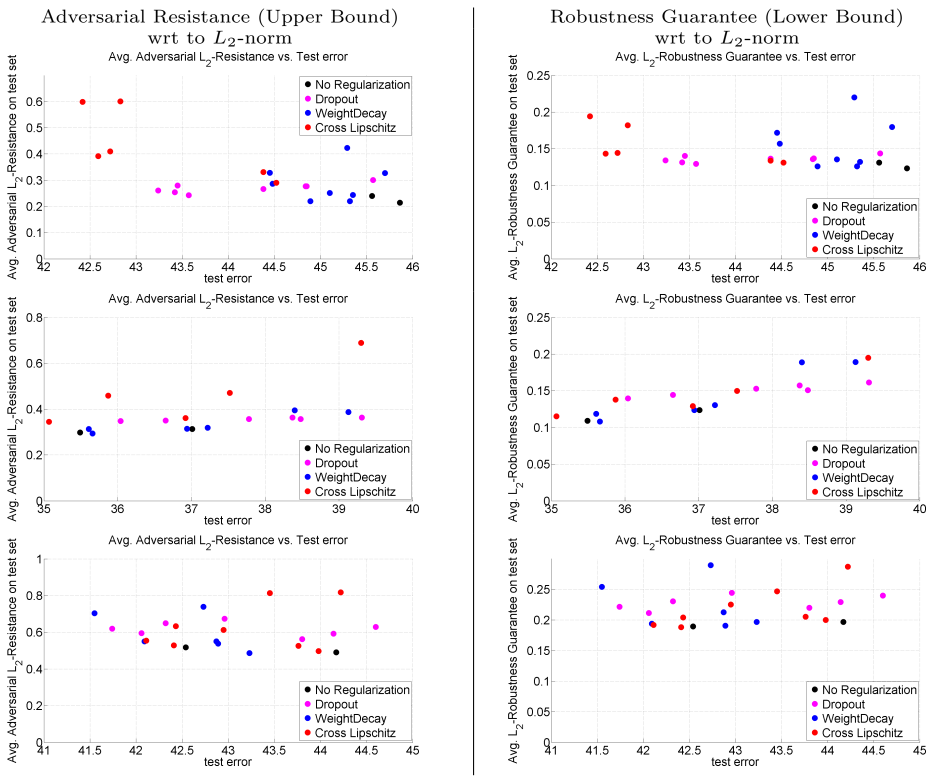


Figure 3: **Left:** Adversarial resistance wrt to L_2-norm on test set of CIFAR10. **Right:** Average robustness guarantee on the test set wrt to L_2-norm for the test set of CIFAR10 for different neural networks (one hidden layer, 1024 HU) and hyperparameters. While Cross-Lipschitz regularization yields good test errors, the guarantees are not necessarily stronger. **Top row:** CIFAR10 (plain), **Middle:** CIFAR10 trained with data augmentation, **Bottom:** Adversarial Training.

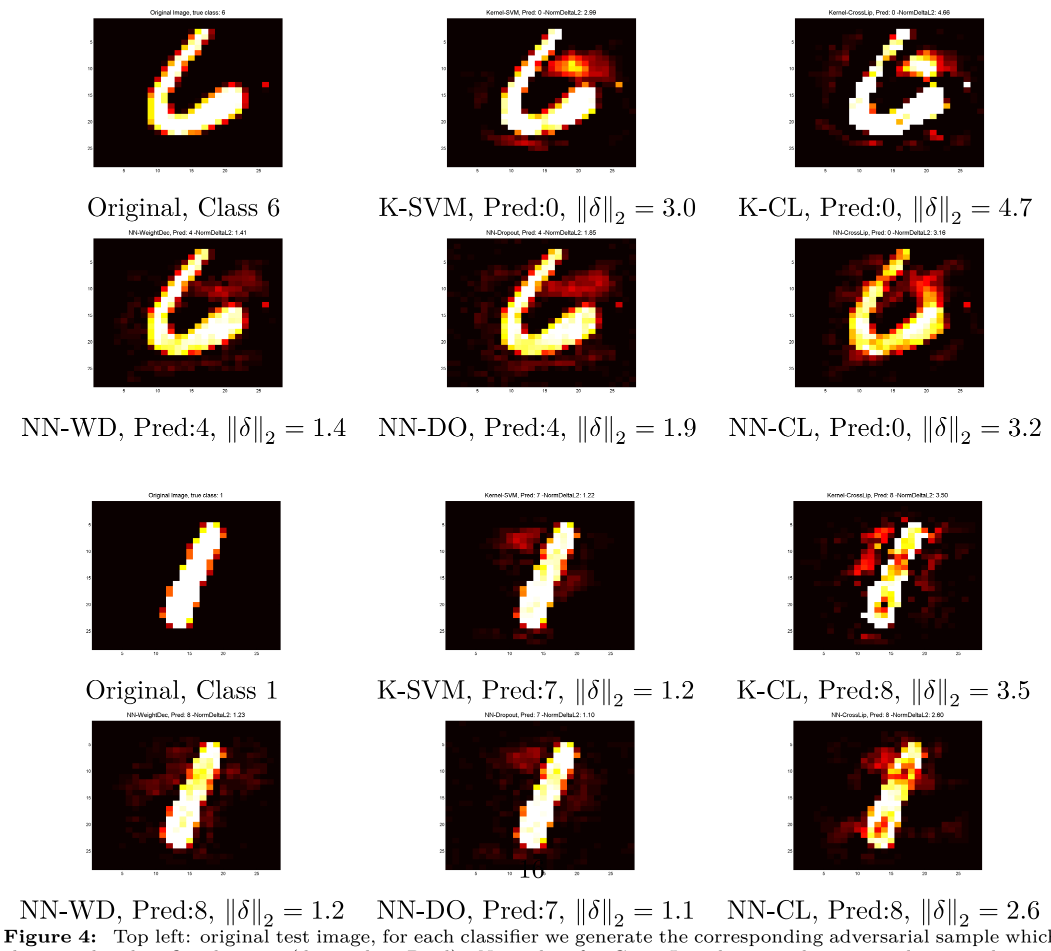
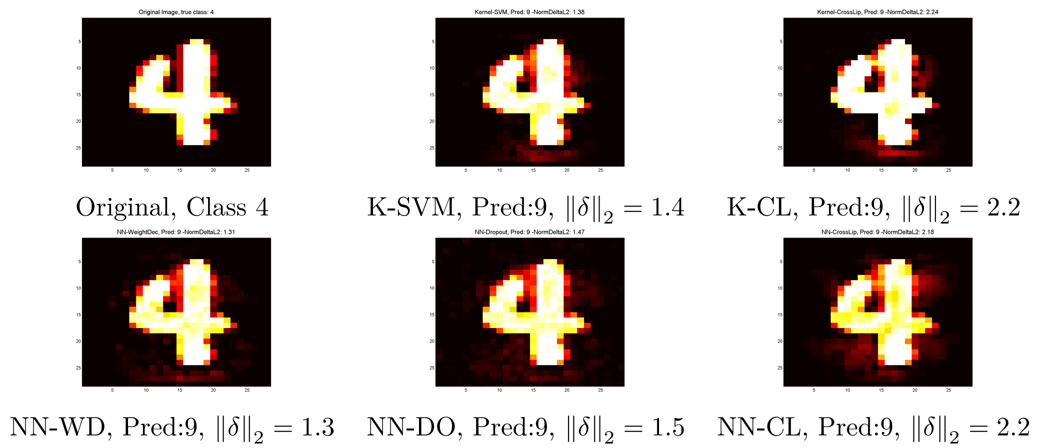


Figure 4: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for Cross-Lipschitz regularization this new decision makes (often) sense, whereas for the neural network models (weight decay/dropout) the change is so small that

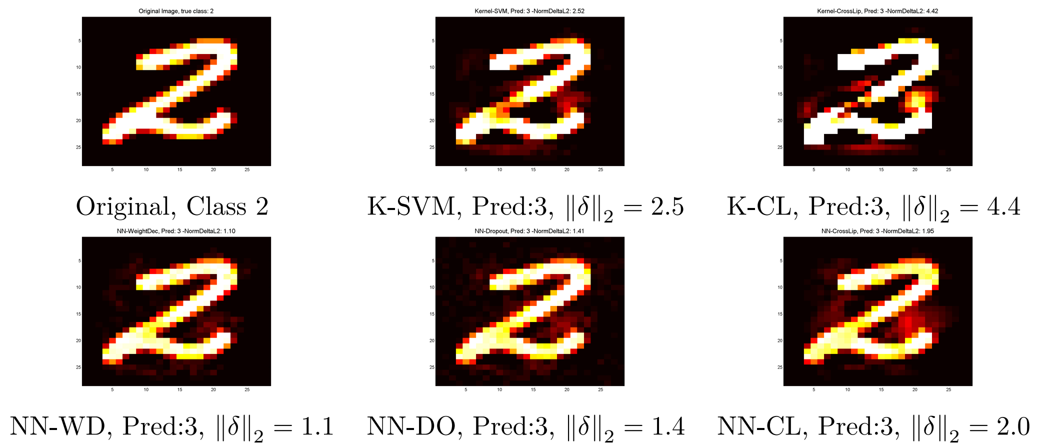
Figure 5: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for the kernel methods this new decision makes sense, whereas for all neural network models the change is so small that the new decision is clearly wrong.


Original Image, true class: 4
Kernel-SVM, Pred: 9 -NormDeltaL2: 1.38
Kernel-CrossLip, Pred: 9 -NormDeltaL2: 2.24
NN-WeightDec, Pred: 9 -NormDeltaL2: 1.31
NN-Dropout, Pred: 9 -NormDeltaL2: 1.47
NN-CrossLip, Pred: 9 -NormDeltaL2: 2.18

Original, Class 4 K-SVM, Pred:9, $\|\delta\|_2 = 1.4$ K-CL, Pred:9, $\|\delta\|_2 = 2.2$

NN-WD, Pred:9, $\|\delta\|_2 = 1.3$ NN-DO, Pred:9, $\|\delta\|_2 = 1.5$ NN-CL, Pred:9, $\|\delta\|_2 = 2.2$

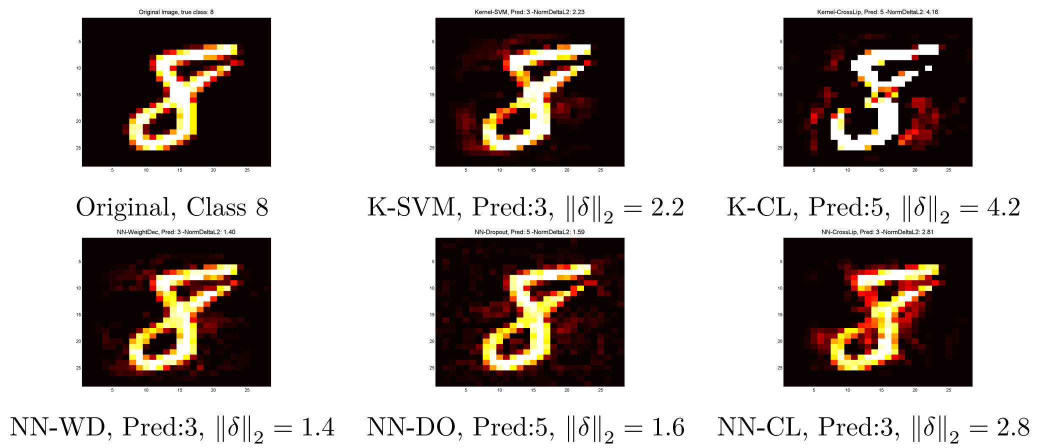
Figure 6: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for the kernel methods this new decision makes sense, whereas for all neural network models the change is so small that the new decision is clearly wrong.


Original Image, true class: 2
Kernel-SVM, Pred: 3 -NormDeltaL2: 2.52
Kernel-CrossLip, Pred: 3 -NormDeltaL2: 4.42
NN-WeightDec, Pred: 3 -NormDeltaL2: 1.10
NN-Dropout, Pred: 3 -NormDeltaL2: 1.41
NN-CrossLip, Pred: 3 -NormDeltaL2: 1.95

Original, Class 2 K-SVM, Pred:3, $\|\delta\|_2 = 2.5$ K-CL, Pred:3, $\|\delta\|_2 = 4.4$

NN-WD, Pred:3, $\|\delta\|_2 = 1.1$ NN-DO, Pred:3, $\|\delta\|_2 = 1.4$ NN-CL, Pred:3, $\|\delta\|_2 = 2.0$

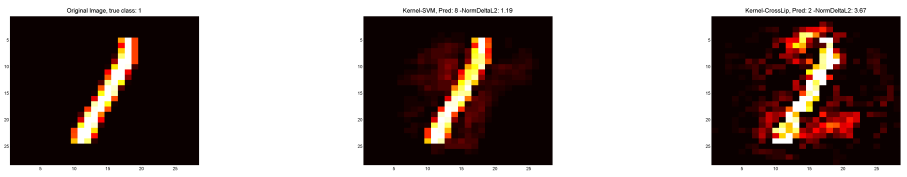
Figure 7: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for the kernel methods this new decision makes sense, whereas for all neural network models the change is so small that the new decision is clearly wrong.


Original Image, true class: 8
Kernel-SVM, Pred: 3 -NormDeltaL2: 2.23
Kernel-CrossLip, Pred: 5 -NormDeltaL2: 4.16
NN-WeightDec, Pred: 3 -NormDeltaL2: 1.40
NN-Dropout, Pred: 5 -NormDeltaL2: 1.59
NN-CrossLip, Pred: 3 -NormDeltaL2: 2.81

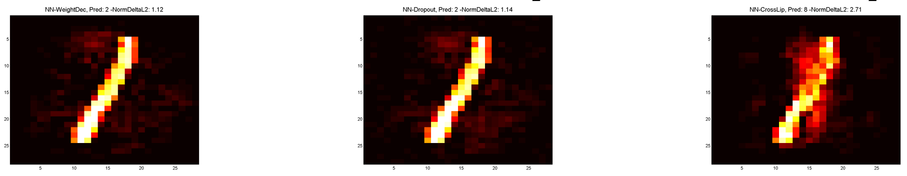
Original, Class 8 K-SVM, Pred:3, $\|\delta\|_2 = 2.2$ K-CL, Pred:5, $\|\delta\|_2 = 4.2$

NN-WD, Pred:3, $\|\delta\|_2 = 1.4$ NN-DO, Pred:5, $\|\delta\|_2 = 1.6$ NN-CL, Pred:3, $\|\delta\|_2 = 2.8$

Figure 8: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for the kernel methods this new decision makes sense, whereas for all neural network models the change is so small that the new decision is clearly wrong.

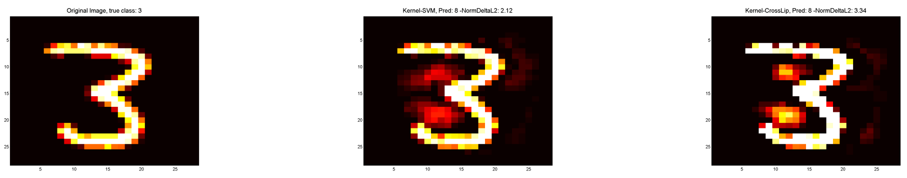

Original Image, true class: 1
Kernel-SVM, Pred: 8 -NormDeltaL2: 1.19
Kernel-CrossLip, Pred: 2 -NormDeltaL2: 3.67

Original, Class 1 K-SVM, Pred:8, $\|\delta\|_2 = 1.2$ K-CL, Pred:2, $\|\delta\|_2 = 3.7$

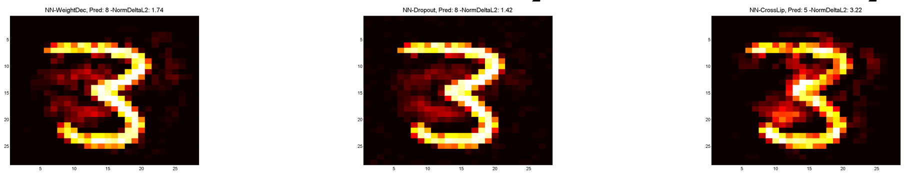

NN-WeightDec, Pred: 2 -NormDeltaL2: 1.12
NN-Dropout, Pred: 2 -NormDeltaL2: 1.14
NN-CrossLip, Pred: 8 -NormDeltaL2: 2.71

NN-WD, Pred:2, $\|\delta\|_2 = 1.1$ NN-DO, Pred:2, $\|\delta\|_2 = 1.1$ NN-CL, Pred:8, $\|\delta\|_2 = 2.7$

Figure 9: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for the kernel methods this new decision makes sense, whereas for all neural network models the change is so small that the new decision is clearly wrong.

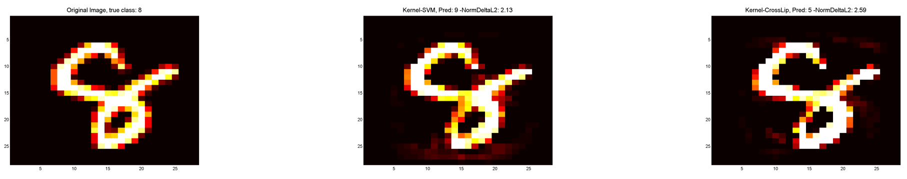

Original Image, true class: 3
Kernel-SVM, Pred: 8 -NormDeltaL2: 2.12
Kernel-CrossLip, Pred: 8 -NormDeltaL2: 3.34

Original, Class 3 K-SVM, Pred:8, $\|\delta\|_2 = 2.1$ K-CL, Pred:8, $\|\delta\|_2 = 3.3$

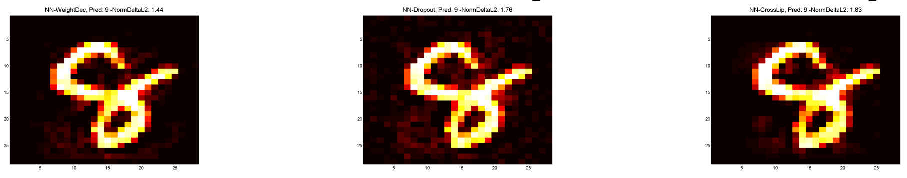

NN-WeightDec, Pred: 8 -NormDeltaL2: 1.74
NN-Dropout, Pred: 8 -NormDeltaL2: 1.42
NN-CrossLip, Pred: 5 -NormDeltaL2: 3.22

NN-WD, Pred:8, $\|\delta\|_2 = 1.7$ NN-DO, Pred:8, $\|\delta\|_2 = 1.4$ NN-CL, Pred:5, $\|\delta\|_2 = 3.2$

Figure 10: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for the kernel methods this new decision makes sense, whereas for all neural network models the change is so small that the new decision is clearly wrong.


Original Image, true class: 8
Kernel-SVM, Pred: 9 -NormDeltaL2: 2.13
Kernel-CrossLip, Pred: 5 -NormDeltaL2: 2.59

Original, Class 8 K-SVM, Pred:9, $\|\delta\|_2 = 2.1$ K-CL, Pred:5, $\|\delta\|_2 = 2.6$


NN-WeightDec, Pred: 9 -NormDeltaL2: 1.44
NN-Dropout, Pred: 9 -NormDeltaL2: 1.76
NN-CrossLip, Pred: 9 -NormDeltaL2: 1.83

NN-WD, Pred:9, $\|\delta\|_2 = 1.4$ NN-DO, Pred:9, $\|\delta\|_2 = 1.8$ NN-CL, Pred:9, $\|\delta\|_2 = 1.8$

Figure 11: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for the kernel methods this new decision makes sense, whereas for all neural network models the change is so small that the new decision is clearly wrong.

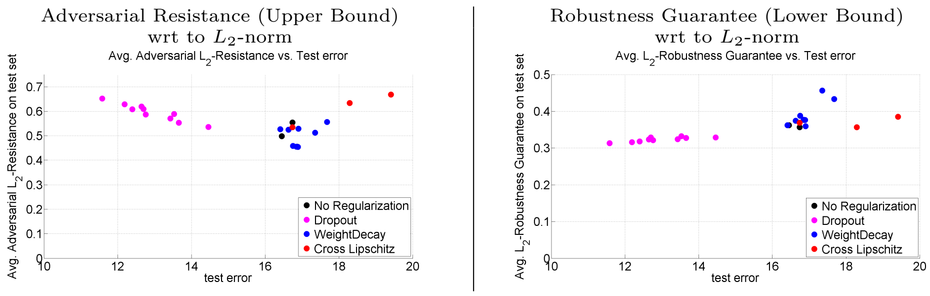
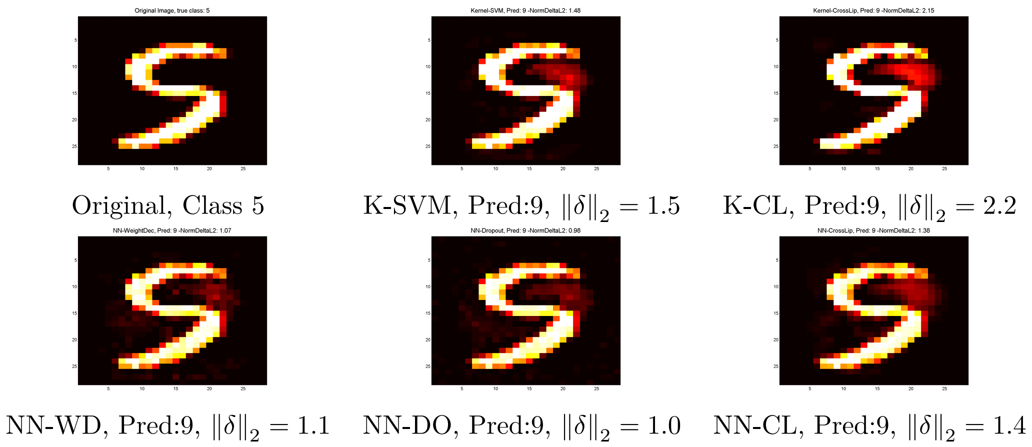


Figure 13: **Left:** Adversarial resistance wrt to L_2-norm on test set of the german traffic sign benchmark (GTSB) in the plain setting. **Right:** Average robustness guarantee on the test set wrt to L_2-norm for the test set of GTSB for different neural networks (one hidden layer, 1024 HU) and hyperparameters. Here dropout performs very well both in terms of performance and robustness.



Original, Class 5
K-SVM, Pred:9, $\|\delta\|_2 = 1.5$
K-CL, Pred:9, $\|\delta\|_2 = 2.2$
NN-WD, Pred:9, $\|\delta\|_2 = 1.1$
NN-DO, Pred:9, $\|\delta\|_2 = 1.0$
NN-CL, Pred:9, $\|\delta\|_2 = 1.4$

Figure 12: Top left: original test image, for each classifier we generate the corresponding adversarial sample which changes the classifier decision (denoted as Pred). Note that for the kernel methods this new decision makes sense, whereas for all neural network models the change is so small that the new decision is clearly wrong.

German Traffic Sign Benchmark: As a third dataset we used the German Traffic Sign Benchmark (GTSB) [23], which consists of images of german traffic signs, which has 43 classes with 34209 training and 12630 test samples. The results are shown in Figure 13. For this dataset Cross-Lipschitz regularization improves the upper bounds compared to weight decay but dropout achieves significantly better prediction performance and has similar upper bounds. The robustness guarantees for weight decay and Cross-Lipschitz are slightly better than for dropout.

Residual Networks: All experiments so far were done with one hidden layer neural networks so that we can evaluate lower and upper bounds. Now we want to demonstrate that Cross-Lipschitz regularization can also successfully be used for deep networks. We use residual networks proposed in [9] with 32 parameter layers and non-bottleneck residual blocks. We follow basically their setting, apart from that we did not subtract the per-pixel mean so that all images are in $[0, 1]^d$ and use random crop but without any padding as in [9]. Similar to [9], we train for 160 epochs, and the learning rate is divided by 10 on the 115-th and 140-th epochs. For the experiments with dropout we followed the recommendation of [25], inserting a dropout layer between convolutional layers inside each residual block. For Cross-Lipschitz regularization we use automatic differentiation in TensorFlow [1] to calculate the derivative with respect to the input, which slows done the training by a factor of 10.

For the plain setting the learning rate for all methods is chosen from $\{0.2, 0.5\}$, except for the runs without regularization, for which it is from $\{0.08, 0.1, 0.2, 0.4, 0.6, 0.8\}$. For weight decay the regularization parameter is chosen from $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, for Cross-Lipschitz from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, and for dropout the probabilities are from $\{0.5, 0.6, 0.7, 0.8\}$. For the data augmentation setting the only difference was in the higher learning rates: no

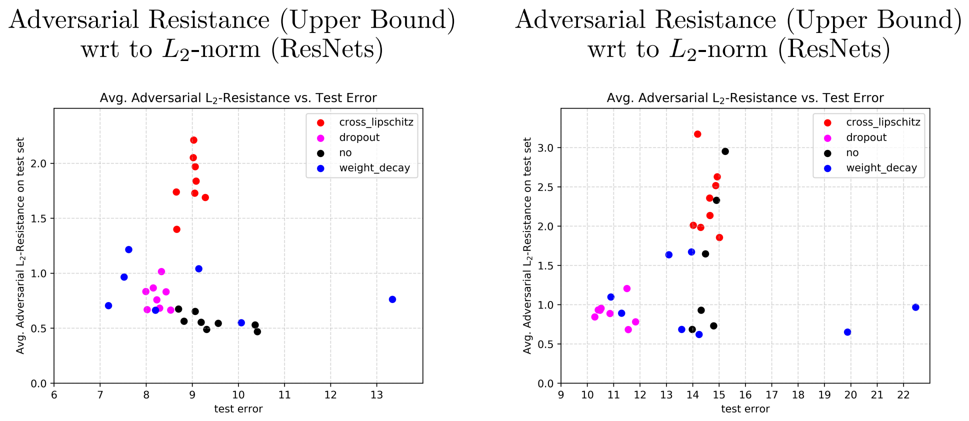


Figure 14: Results on CIFAR10 for a residual network with different regularizers. As we only have lower bounds for one hidden layer networks, we can only show upper bounds for adversarial resistance. **Left:** with data augmentation similar to [25] **Right:** plain setting

regularization - $\{0.2, 0.5, 0.8, 1.0, 1.5, 2.0, 3.0, 4.0\}$, weight decay - $\{0.1, 0.4\}$, Cross-Lipschitz - $\{0.2, 1.0\}$. The results are shown in Figure 14. Cross-Lipschitz regularization improves the upper bounds on the robustness against adversarial manipulation compare to weight decay and dropout by a factor of 2 to 3 both in the plain setting (right) and with data augmentation (left). This comes at a price of a slightly worse test performance. However, it shows that Cross-Lipschitz regularization is also effective for deep neural networks. It remains interesting future work to come up also with interesting instance-specific lower bounds (robustness guarantees) for deep neural networks.

Outlook Formal guarantees on machine learning systems are becoming increasingly more important as they are used in safety-critical systems. We think that there should be more research on robustness guarantees (lower bounds), whereas current research is focused on new attacks (upper bounds). We have argued that our instance-specific guarantees using our local Cross-Lipschitz constant is more effective than using a global one and leads to lower bounds which are up to 8 times better. A major open problem is to come up with tight lower bounds for deep networks.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016.

[2] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi. Measuring neural net robustness with constraints. In *NIPS*, 2016.

[3] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017.

[4] M. Cisse, P. Bojanowksi, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. In *ICML*, 2017.

[5] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *KDD*, 2004.

[6] H. Drucker and Y. Le Cun. Double backpropagation increasing generalization performance. In *IJCNN*, 1992.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[8] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. In *ICLR Workshop*, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[10] D. P. Helmbold and P. Long. On the inductive bias of dropout. *Journal of Machine Learning Research*, 16:3403–3454, 2015.

[11] S. Hochreiter and J. Schmidhuber. Simplifying neural nets by discovering flat minima. In *NIPS*, 1995.

[12] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvari. Learning with a strong adversary. In *ICLR*, 2016.

[13] J. Kos, I. Fischer, and D. Song. Adversarial examples for generative models. In *ICLR Workshop*, 2017.

[14] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *ICLR Workshop*, 2017.

[15] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.

[16] D. Lowd and C. Meek. Adversarial learning. In *KDD*, 2005.

[17] S.M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *CVPR*, 2017.

[18] N. Papernot, P. McDonald, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep networks. In *IEEE Symposium on Security & Privacy*, 2016.

[19] P. Frossard S.-M. Moosavi-Dezfooli, A. Fawzi. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582, 2016.

[20] B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

[21] U. Shaham, Y. Yamada, and S. Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. In *NIPS*, 2016.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[23] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, pages 2503–2511, 2014.

[25] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, pages 87.1–87.12.

[26] S. Zheng, Y. Song, T. Leung, and I. J. Goodfellow. Improving the robustness of deep neural networks via stability training. In *CVPR*, 2016.