Measuring Neural Net Robustness*

Osbert Bastani[1], Yani Ioannou[2], Leonidas Lampropoulos[3], Dimitrios Vytiniotis[4], Aditya V. Nori[4], and Antonio Criminisi[4]

[1] MIT
obastani@csail.mit.edu
[2] University of Cambridge
yai20@cam.ac.uk
[3] University of Pennsylvania
llamp@seas.upenn.edu
[4] Microsoft Research
{dimitris, adityan, antcrim}@microsoft.com

Abstract. Neural nets have been shown to be susceptible to adversarial examples, where a small perturbation to an input can cause it to become mislabeled. We propose metrics for measuring the robustness of a neural net and devise a novel algorithm for approximating them. We show how existing approaches to improving robustness "overfit" to adversarial examples generated using a specific algorithm.

Recent work [6] shows that it is often possible to construct an input mislabeled by a neural net by perturbing a correctly labeled input by a tiny amount in a carefully chosen direction. Lack of robustness can be problematic in a variety of settings, such as changing camera lens or lighting conditions, successive frames in a video, or adversarial attacks in security-critical applications [5].

Approaches have since been proposed to improve robustness [2]. However, work in this direction has been handicapped by the lack of objective measures of robustness. A typical approach to improving the robustness of a neural net f is to use an algorithm $\mathcal{A}$ to find adversarial examples, augment the training set with these examples, and train a new neural net f' [2]. Robustness is then evaluated by using the *same* algorithm $\mathcal{A}$ to find adversarial examples for f'—if $\mathcal{A}$ discovers fewer adversarial examples for f' than for f, then f' is concluded to be more robust than f. However, f' may have *overfit* to adversarial examples generated by $\mathcal{A}$—in particular, a *different* algorithm $\mathcal{A}'$ may find as many adversarial examples for f' as for f. Having an objective robustness measure is vital not only to reliably compare different algorithms, but also to understand robustness of production neural nets—e.g., when deploying a login system based on face recognition, a security team may need to evaluate the risk of an attack using adversarial examples.

Thus, it is critical that we develop algorithms for measuring the robustness of neural nets [4,3]. In this paper, we propose scalable algorithms for measuring robustness. Using our techniques, we show evidence that existing algorithms

* This paper is based on [1].

designed to improve the robustness of neural nets can overfit to adversarial examples identified a specific algorithm.

Defining robustness. We begin by formalizing the notion of robustness in [2,6]. Consider a classifier $f : \mathcal{X} \to \mathcal{L}$, where $\mathcal{X} \subseteq \mathbb{R}^n$ is the input space and $\mathcal{L} = \{1, ..., L\}$ are labels. Intuitively, f is robust at $\mathbf{x}_* \in \mathcal{X}$ if a "small" perturbation to $\mathbf{x}_*$ does not affect the assigned label. We are interested in perturbations sufficiently small that they do not affect human classification; an established condition is $\|\mathbf{x} - \mathbf{x}_*\|_\infty \leq \epsilon$ for some parameter ϵ. We say f is $(\mathbf{x}_*, \epsilon)$-*robust* if for every $\mathbf{x}$ such that $\|\mathbf{x} - \mathbf{x}_*\|_\infty \leq \epsilon$, $f(\mathbf{x}) = f(\mathbf{x}_*)$. Then, the *pointwise robustness* $\rho(f, \mathbf{x}_*)$ of f at $\mathbf{x}_*$ is the minimum ϵ for which f fails to be $(\mathbf{x}_*, \epsilon)$-robust:

$$\rho(f, \mathbf{x}_*) \stackrel{\text{def}}{=} \inf\{\epsilon \geq 0 \mid f \text{ is not } (\mathbf{x}_*, \epsilon)\text{-robust}\}. \tag{1}$$

Finally, the *adversarial frequency*

$$\phi(f, \epsilon) \stackrel{\text{def}}{=} \Pr_{\mathbf{x}_* \sim \mathcal{D}}[\rho(f, \mathbf{x}_*) \leq \epsilon]$$

measures how often f fails to be $(\mathbf{x}_*, \epsilon)$-robust. In other words, if f has high adversarial frequency, then it fails to be $(\mathbf{x}_*, \epsilon)$-robust for many inputs $\mathbf{x}_*$.

Computing robustness. We give a high-level overview of how we compute robustness; see [1] for details. We compute $\rho(f, \epsilon)$ by expressing (1) as a system $\mathcal{C}$ of linear constraints. For a neural net f with ReLU activations, the constraint $f(\mathbf{x}) = \ell$ can be expressed as constraints $\mathcal{C}_f(\mathbf{x}, \ell)$; i.e., $f(\mathbf{x}) = \ell$ if and only if $\mathcal{C}_f(\mathbf{x}, \ell)$ is satisfiable. Then, $\rho(f, \mathbf{x}_*)$ can be computed as follows:

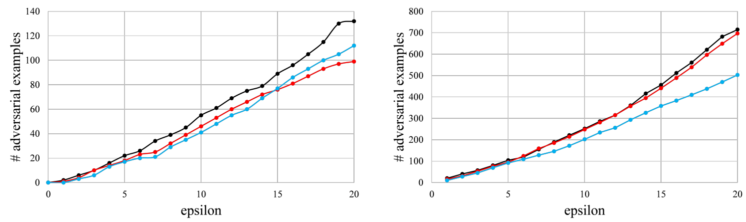
$$\rho(f, \mathbf{x}_*) = \min_{\ell \neq \ell_*} \rho(f, \mathbf{x}_*, \ell) \tag{2}$$

$$\rho(f, \mathbf{x}_*, \ell) \stackrel{\text{def}}{=} \inf\{\epsilon \geq 0 \mid \mathcal{C}_f(\mathbf{x}, \ell) \wedge \|\mathbf{x} - \mathbf{x}_*\|_\infty \leq \epsilon \text{ satisfiable}\}. \tag{3}$$

Solving (3) is typically intractable. To recover tractability, we approximate (3) by constraining the search to a convex region $\mathcal{Z}(\mathbf{x}_*)$ around $\mathbf{x}_*$, which we call a *convex restriction*. Furthermore, we devise an iterative approach to solving the resulting linear program that produces an order of magnitude speed-up.

Improving robustness. We can use our algorithm to compute adversarial examples. Given $\mathbf{x}_*$, the value of $\mathbf{x}$ computed by the optimization procedure used to solve (3) is an adversarial example for $\mathbf{x}_*$ with $\|\mathbf{x} - \mathbf{x}_*\|_\infty = \hat{\rho}(f, \mathbf{x}_*)$. Then, we use *fine-tuning* to reduce a neural net's susceptibility to adversarial examples [2]. First, we use an algorithm $\mathcal{A}$ to compute adversarial examples for each $\mathbf{x}_* \in X_{\text{train}}$ and add them to the training set. Then, we continue training f on a the augmented training set at a reduced training rate.

Empirical results. We evaluate our approach on a deep convolutional neural net f_0 for MNIST, comparing our algorithm $\mathcal{A}_{\text{LP}}$ to the baseline $\mathcal{A}_{\text{L-BFGS}}$ from [6]. First, we improve the robustness of f_0 using adversarial examples computed by



(a) (b)

Fig. 1: The (unnormalized) adversarial frequency $\phi(f, \epsilon)$, for f_0 (black), $f_{\text{L-BFGS}}$ (red), and f_{LP} (blue), estimated using (a) $\mathcal{A}_{\text{L-BFGS}}$ and (b) $\mathcal{A}_{\text{LP}}$.

$\mathcal{A}_{\text{L-BFGS}}$ and $\mathcal{A}_{\text{LP}}$ to obtain $f_{\text{L-BFGS}}$ and f_{LP}, respectively. In Figure 1, we plot the adversarial frequency $\phi(f, \epsilon)$ as a function of ϵ, estimated using (a) $\mathcal{A}_{\text{L-BFGS}}$ and (b) $\mathcal{A}_{\text{LP}}$, for each f_0 (black), f_{LP} (red), and $f_{\text{L-BFGS}}$ (blue).

According to the baseline estimate of $\phi(f, \epsilon)$ in Figure 1 (a), $f_{\text{L-BFGS}}$ is similarly robust to f_{LP}, and both are more robust than f_0. However, according to our estimate of $\phi(f, \epsilon)$ in Figure 1 (b), f_{LP} is substantially more robust than $f_{\text{L-BFGS}}$. In particular, the neural net $f_{\text{L-BFGS}}$ fine-tuned using the baseline algorithm does not learn the adversarial examples found by our algorithm, whereas the neural net f_{LP} fine-tuned using our algorithm learns both the adversarial examples found by our algorithm *and* those found by the baseline algorithm.

Finally, we have implemented our approach for the CIFAR-10 network-in-network (NiN) neural net, which has an accuracy of 91.3%. NiN suffers severely from adversarial examples—our estimate of its adversarial frequence is 61.5%. For NiN fine-tuned using our algorithm, adversarial frequency is reduced to 59.6%, though accuracy is also reduced to 90.4%.

Conclusion We have shown how to formulate and measure robustness of neural nets. Future work includes devising better approaches for improving robustness, and studying robustness properties beyond pointwise robustness.

References

1. Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *NIPS*, 2016.
2. Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.
3. Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *CAV*, 2017.
4. Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *CAV*, 2017.
5. Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ASIACCS*, 2017.

6. Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.