
Measuring the Robustness of Neural Networks via Minimal Adversarial Examples

Anonymous Author(s)

Affiliation

Address

email

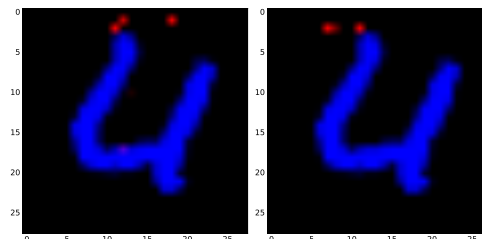
Abstract

1 Neural networks are highly sensitive to adversarial examples, which cause large
2 output deviations with only small input perturbations. However, little is known
3 *quantitatively* about the distribution and prevalence of such adversarial examples.
4 To address this issue, we propose a rigorous search method that provably finds the
5 *smallest* possible adversarial example. The key benefit of our method is that it
6 gives precise quantitative insight into the distribution of adversarial examples, and
7 guarantees the absence of adversarial examples if they are not found. The primary
8 idea is to consider the nonlinearity exhibited by the network in a small region of the
9 input space, and search exhaustively for adversarial examples in that region. We
10 show that the frequency of adversarial examples and robustness of neural networks
11 is up to twice as large as reported in previous works that use empirical adversarial
12 attacks. In addition, we provide an approach to approximate the nonlinear behavior
13 of neural networks, that makes our search method computationally feasible.

14 Neural networks are not robust: Szegedy et al. [2013] first showed that neural networks can be made
15 to incorrectly classify correctly labeled inputs by perturbing the input by a small amount. Hence,
16 there are no safety and security guarantees when applying neural networks in real-world applications
17 (e.g. autonomous vehicles). A key issue is to compute objective measures for network robustness, e.g.
18 how many adversarial examples arise (*frequency*) and the size of (the smallest) adversarial example
19 (*severity*). However, this is challenging, as state-of-the-art networks can be large and highly nonlinear.

20 To address this issue, we propose a rigorous search
21 method to construct adversarial examples for a class
22 of neural networks: fully-connected networks with
23 ReLU activations. Our method combines three in-
24 sights: 1) ReLU neural networks can be viewed as a
25 collection of logical clauses (as proposed in Bastani
26 et al. [2016]). 2) By considering the behavior of net-
27 work activations in regions around an input, finding
28 an adversarial example can be reduced to solving a
29 large system of independent linear programs. 3) We
30 propose an approximation to the nonlinear behavior
31 of network activations, as constructing the set of
32 feasible piecewise linear regions for the hierarchy
33 of network layers can be prohibitively expensive.
34 Combining these, we obtain a tractable approach to
35 construct adversarial examples.

36 The key benefits of our approach are two-fold: 1) our method provably finds the smallest adversarial
37 example for any specified output change (or guarantees the absence of such adversarial examples)
38 and hence 2) it gives unbiased quantitative measures of the robustness of the neural network.



(a) Baseline $|\Delta x| = 3.6$ (b) Ours $|\Delta x| = 2.4$

Figure 2: Adversarial examples that switch prediction from 4 to 6. Our method consistently finds smaller adversarial perturbations than baselines. Blue: original; red: Δx .

39 Several adversarial attacks have previously been proposed: Goodfellow et al. [2014], Carlini and
 40 Wagner [2016] use input space gradients, while Bastani et al. [2016] use a constraint programming
 41 based approach, similar to us. However, these methods do not provide theoretical guarantees on
 42 completeness nor optimality. [Katz et al., 2017, Ehlers, 2017, Huang et al., 2017] use formal methods
 43 to analyze the network in its entirety, but these do not scale well for larger networks. Instead, our
 44 method both guarantees minimality of the adversarial example and scales better to large networks.

45 In this work, we present a tractable approach to find adversarial examples and analyze local network
 46 robustness via a model reduction to constraints in linear arithmetic. Our contributions are as follows:

- 47 • We propose an adversarial attack that uses a reduction of neural networks into logical clauses
 48 and an iterative search over *feasible* network activations to find adversarial examples.
- 49 • We make this search tractable via approximation bounds on the set of feasible activations.
- 50 • We show that our method provably finds the minimal adversarial example for an input x or
 51 guarantees its absence within its neighborhood.
- 52 • We validate our method by showing that ReLU neural networks can be attacked with
 53 examples with a smaller magnitude, and up to twice as frequently as previously reported.

54 1 Finding Adversarial Examples via Satisfiability Problems

55 Our goal is to find the minimal adversarial perturbation $\Delta x \in \mathbb{R}^d$ for a given neural network classifier
 56 $f(x)$ that changes f 's output to a target class j :

$$\Delta x^* = \underset{\Delta x}{\operatorname{argmin}} \{ \|\Delta x\|_p \mid f(x) = i, f(x + \Delta x) = j, i \neq j \}. \quad (1)$$

57 Bastani et al. [2016] quantified the distribution of such examples using the point-wise robustness
 58 $\rho(x, f, j)$, the smallest ball around x that does not contain adversarial examples:

$$\rho(x, f, j) = \inf \{ \epsilon \mid \exists \Delta x : f(x + \Delta x) = j, \|\Delta x\|_p \leq \epsilon \}. \quad (2)$$

59 where $p = 1, 2, \infty$ has been considered in literature. In our work we use $p = 1$, but the principles
 60 hold for the other norms as well. Given a distribution D from which data points are sampled, based
 61 on ρ , the adversarial frequency ϕ and severity μ are the prevalence and average robustness:

$$\phi(f, \delta, j) = \mathbb{P}_{x \sim D}(\rho(x, f, j) \leq \delta), \quad \mu(f, \delta, j) = \mathbb{E}_{x \sim D}(\rho(x, f, j) \mid \rho(x, f, j) \leq \delta) \quad (3)$$

62 **ReLU networks as logic constraints** Hereafter, we will consider fully-connected networks with
 63 ReLU activations. We follow Bastani et al. [2016] and describe a ReLU network as a logic formula
 64 in linear arithmetic. Here, each ReLU node y corresponds to a logical disjunction:

$$y = \max(0, w^T x) \Leftrightarrow (y = 0 \wedge w^T x \leq 0) \vee (y = w^T x \wedge y > 0), \quad (4)$$

65 where each clause represents a linear region (polytope) of its input space. A change in the input x can
 66 cause a *switch in activation*: the active clause can change from $w^T x > 0$ to $w^T x \leq 0$, or vice versa.
 67 Similarly, a network with m ReLU nodes across all layers can be represented as a single clause with
 68 m disjunctions, whose 2^m combinations of node activations each represent a linear region for the
 69 network. Finding an adversarial example then corresponds to finding an x' that satisfies this m -node
 70 clause and activates a desired target node in the output layer.

71 An iterative approach would 1) sweep through the network sequentially node-by-node, 2) check
 72 what combination of node activation states can be attained for the network and 3) search over these
 73 combinations for the minimal adversarial example. However, this is computationally expensive
 74 and the sequential nature of the search prohibits parallelization. A naive parallelized search would
 75 consider 2^m combinations of activation patterns and is prohibitively expensive.

76 **Searching for provably minimal adversarial examples** We now propose a tractable search algo-
 77 rithm (outlined in Algorithm 1) to find minimal adversarial examples in a candidate set of perturbations
 78 $B(x, \epsilon) = \{x' \mid \|x' - x\| \leq \epsilon\}$ around an input x . The main idea is to use a parallelized approach
 79 and limit the number of activations to search over. Given a set of inputs $C_l(x_l)$ for layer l , we
 80 compute a convex-relaxation $C_{l+1}(x_{l+1})$ of the image of $C_l(x_l)$ through the layer $\{\text{ReLU}(x_l)\}$.
 81 These relaxations are then used to parallelize.

Algorithm 1 Minimal Adversarial Example Search

Input: Neural Network f with L layers, perturbation radius ϵ , target-class j .

- 1: **for** layer $l \in L$ layers **do**
 - 2: **GetInputRegions:** Construct constraints $C_l(x_l)$ using Lemma 2 that bound the inputs $\{x_l\}$.
 - 3: **GetSwitchingNodes:** Collect nodes in layer l that switch activation (equation 4).
 - 4: **end for**
 - 5: **for** layer $l \in L$ layers **do**
 - 6: **CheckAct:** Given $C_l(x_l)$, check feasible activation states for switching nodes in layer l .
 - 7: **GetFeasibleRegions:** Get constraints $\tilde{S}_l(x_l)$ for *feasible* linear-regions under $C_l(x_l)$.
 - 8: **end for**
 - 9: Construct combinations (P_1, P_2, \dots, P_t) of $\{\tilde{S}_l\}$.
 - 10: Search $\{P_i\}$ for minimal adversarial input perturbation Δx that changes output to target class j .
-

82 More specifically, our approach has the following key steps. First, we find the nodes for each layer
83 that can switch activation state (**GetSwitchingNodes**), on a layer-by-layer basis. For the first ReLU
84 layer $y = \max(0, Wx + u)$, we define the constraint $C(x) := \|\Delta x\| \leq \epsilon$. To check if a node y_i
85 switches, we check whether both the constraints $W_i^T x + u_i > 0 \wedge C(x)$ and $W_i^T x + u_i \leq 0 \wedge C(x)$
86 have satisfying assignments. If both are satisfiable, then the node can potentially switch activation for
87 some perturbation.

88 Then, for the next layer, the constraint $C(y)$ (**GetInputRegions**) is determined by bounding Δy
89 using a relaxation $\|\Delta y - A\Delta x\| \leq \|\Delta \bar{A}\|\epsilon$ (Lemma 2) and $\|\Delta x\| \leq \epsilon$. We then use $C(y)$ to find
90 the switching nodes, and repeat this layer-wise. Next, in a layer-wise manner, we compute *feasible*
91 combinations s_{switch}^l of node activations for the nodes that can switch in each layer (**CheckAct**).
92 Such combinations determine a linear region \tilde{S}_l for the layer output, which can be represented as
93 $s_{\text{switch}}^l \wedge y = Ax + b$ (**GetFeasibleRegions**).

94 Finally, we construct linear regions P_i by combining regions defined in $\{\tilde{S}_l\}$ across layers and search
95 for the smallest adversarial example δ_i within P_i . Since we have searched over all the linear-regions
96 of the network that can be activated by an input in $B(x, \epsilon)$, we have:

97 **Lemma 1.** *If $\Delta x^* = \min\{\Delta x_i : i = 1, 2, \dots, t\} \leq \epsilon$, then Δx_i is the globally minimal adversarial*
98 *example for the targeted attack. Otherwise, there exists no adversarial examples in $B(x, \epsilon)$.*

99 **Parallelization** This approach can be parallelized at two levels: 1) Given a set of inputs for a layer
100 l , the set of switching nodes for a layer can be determined in parallel across nodes. 2) After we have
101 computed the over-approximation of the image of $B(x, \epsilon)$ at each layer, we can compute the possible
102 activation patterns for each of the layers in parallel.

103 **Approximating perturbations** In order to improve search efficiency, in Algorithm 1 we consider
104 a relaxation of the network’s nonlinear behavior over the input region $B(x, \epsilon)$. To do so, we derive an
105 upper bound on the output perturbation Δy given a set of perturbed inputs $x' = x + \Delta x$ for a single
106 ReLU layer $y = \max(0, Wx + u)$:

107 **Lemma 2.** *Consider a ReLU layer $y = \max(0, Wx + u)$ whose local linear behavior at a given x*
108 *can be represented as $y = Ax + b$. For any input perturbation $\|\Delta x\| \leq \epsilon$, Δy satisfies*

$$\|\Delta y - A\Delta x\| \leq \epsilon \|\Delta \bar{A}\|. \quad (5)$$

109 *where the rows of $\Delta \bar{A}$ are the weights for the nodes that can switch for a perturbation $\|\Delta x\| \leq \epsilon$.*

110 *Proof.* Given $x' = x + \Delta x$, the ReLU output changes as:

$$y + \Delta y = \max(0, W(x + \Delta x) + u) = (A + \Delta A)x + (b + \Delta b), \quad (6)$$

111 where $\Delta A, \Delta b$ capture the nonlinear behavior of the activation function. Consider a node y_i that can
112 be made to switch with some such perturbation Δx . There exists some $0 \leq \alpha_i \leq 1$, such that

$$(A_i + \Delta A_i)(x + \alpha_i \Delta x) + (b_i + \Delta b_i) = 0 = A_i(x + \alpha_i \Delta x) + b_i, \quad (7)$$

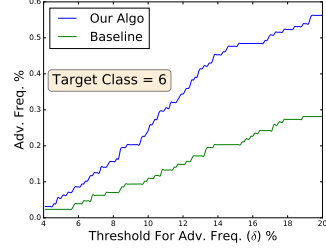
113 Combining the relations above, we get

$$\Delta y_i = (A + (1 - \alpha_i)\Delta A_i)\Delta x_i = (A + \Psi_i \Delta \bar{A}_i)\Delta x_i. \quad (8)$$

Method ($\delta = 5$)	Frequency ρ	Metric ($\delta = 20$)	Method / Target class	5	6
Baseline	26.85%	Severity μ	Baseline	12.16	17.52
Our approach	36.55%	Severity μ	Ours	8.35	13.82

(a) Frequency ρ in %.

Target class	0	1	2	3	4	5	6	7	8	9
Baseline ($\delta = 5$)	1.8	3.4	6.0	13.4	2.9	17.6	2.4	9.5	7.2	11.5
Our approach ($\delta = 5$)	2.2	3.9	9.6	18.0	5.1	23.8	6.6	14.8	11.9	15.7
Baseline ($\delta = 10$)	7.9	7.7	8.6	47.0	10.1	42.5	11.5	22.6	38.2	23.9
Our approach ($\delta = 10$)	16.8	11.1	60.0	63.2	18.5	62.8	27.3	40.0	56.4	48.8

(b) Severity μ in net pixel change.(c) Frequency ρ in %.Figure 5: Adversarial frequency for various thresholds δ .

114 For non-switching nodes, we have $\Delta y_i = A_i \Delta x + \Delta b_i$. Note that for the ReLU activation $\Delta A_i =$
115 $\pm \mathbf{W}_i$. Hence, for any given perturbation Δx , we have $(1 - \alpha_i) \Delta A_i \Delta x_i = \Psi_i \Delta \bar{A}_i \Delta x_i$, with $\Psi_i =$
116 $\pm(1 - \alpha_i)$ for switching nodes and $\Psi_i = 1 - \alpha_i = 0$ for nodes that do not switch. Equivalently, we can
117 write $\Delta y - A \Delta x = \Psi \Delta \bar{A} \Delta x$, where Ψ is a diagonal matrix with Ψ_i as its i th diagonal entry. Since
118 $\|\Psi\|_p \leq 1$ for $p = 1, 2, \infty$, equation (8) reduces to $\|\Delta y - A \Delta x\| \leq \|\Delta \bar{A}\| \cdot \|\Delta x\| \leq \|\Delta \bar{A}\| \epsilon$. \square

119 We also enforce $y + \Delta y \geq 0$. Here, we considered a convex-relaxation of the set of feasible
120 perturbations ΔA . Note that Lemma 2 can be applied iteratively across layers and is tighter than the
121 bound used in Szegedy et al. [2013], which used the operator norm of W i.e $\|\Delta y\| \leq \|W\| \cdot \|\Delta x\|$.

122 2 Experiments

123 We trained a 3-layer network with 60-30-10 hidden units to an accuracy of 97% on MNIST [LeCun
124 and Cortes, 2010]. For 300 randomly sampled examples, we compute the adversarial frequency for
125 targeted and untargeted attacks, and severity and robustness for targeted attacks. We use the l_1 -norm
126 to measure distances: changing one pixel from black to white corresponds to an l_1 -norm of 1. In our
127 experiments, we use the l_1 -ball $B(x, \epsilon = 5.0)$ around each sample x to find the minimal adversarial
128 example and compare all results to Bastani et al. [2016], which is a recent state-of-the-art approach to
129 measuring robustness.

130 We find that on all metrics, our method outperforms the baseline. First, our method achieves
131 significantly higher frequencies for both targeted (Table 3c) and untargeted (Table 3a) attacks. Here,
132 our method includes linear-regions feasible with $\epsilon = 5$, but we compare frequencies against baseline
133 for examples with threshold $\Delta x \leq \delta = 5, 10$. For the $\epsilon < \delta = 10.0$ case, our method still succeeds
134 more frequently even though we consider fewer candidate perturbations than strictly allowed for. For
135 this case, the guarantees regarding unbiasedness of the measurements do not hold.

136 This trend is consistent as δ changes: Figure 5 shows our method achieves higher adversarial
137 frequency for all δ , for two representative target classes; we see a similar trend in Tables 3b and 3c.
138 Moreover, our method finds smaller adversarial examples: the severity (see Table 3b) is significantly
139 smaller. Hence, network is significantly less robust than estimated by baselines (see Table 3b). Note
140 that, for fair comparison, we compute severity only with those examples where the baseline attack
141 succeeds with $\delta < 20.0$.

142 Figure 2 shows a representative instance where our attack finds a smaller adversarial perturbation
143 than the baseline. We see that the perturbation found is different in nature from the baseline.

144 3 Future work

145 The constraints corresponding to a linear region and the output being assigned a specific label
146 represent a decision boundary for the network. This could allow us to compute population statistics,
147 such as the local density of adversarial samples. To scale to large networks, efficient algorithms to
148 construct *interpolants* in linear-arithmetic [Albarghouthi and McMillan, 2013, Lynch and Tang, 2008]
149 can be used to approximate the boundaries. Moreover, our approach could aid the design of robust
150 networks and analysis of defenses against adversarial attack.

151 **References**

- 152 C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing
153 properties of neural networks. *ArXiv e-prints*, December 2013.
- 154 Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and
155 Antonio Criminisi. Measuring neural net robustness with constraints. In *Advances in Neural
156 Information Processing*, pages 2613–2621, 2016.
- 157 I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *ArXiv
158 e-prints*, December 2014.
- 159 N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. *ArXiv e-prints*,
160 August 2016.
- 161 G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An Efficient SMT Solver for
162 Verifying Deep Neural Networks. *ArXiv e-prints*, 2017.
- 163 Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. *CoRR*,
164 abs/1705.01320, 2017. URL <http://arxiv.org/abs/1705.01320>.
- 165 Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural
166 networks. In Rupak Majumdar and Viktor Kunčák, editors, *Computer Aided Verification: 29th
167 International Conference, CAV, 2017, Proceedings, Part I*, pages 3–29, Cham, 2017. Springer
168 International Publishing.
- 169 Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL [http://yann.
170 lecun.com/exdb/mnist/](http://yann.lecun.com/exdb/mnist/).
- 171 Aws Albarghouthi and Kenneth L. McMillan. Beautiful interpolants. In *Proceedings of the 25th
172 International Conference on Computer Aided Verification, CAV'13*, pages 313–329, 2013.
- 173 Christopher Lynch and Yuefeng Tang. Interpolants for linear arithmetic in smt. In *ATVA 2008, Seoul,
174 Korea, October 20-23, 2008. Proceedings*, pages 156–170, 2008.