Improved Adversarial Robustness Via Abstract Interpretation

Zachary DeStefano zd2131@nyu.edu
Ildebrando Magnani im975@nyu.edu
William Merrill willm@nyu.edu

New York University

We improve existing techniques from abstract interpretation for training and certifying adversarial resistant machine learning models and apply our techniques to the problem of CIFAR-10 classification. Our modifications offer improved completeness for certification and increased accuracy from training without sacrificing soundness or performance and work independent of dimension. Starting from the hypothesis that there is a direct relationship between adversarial model robustness and margin, we explore what this implies for the shape of the decision boundary, the training process, and potential future directions for margin maximization via abstract interpretation. [1]

1 Introduction

Adversarial examples are bounded-distance perturbations on in-distribution inputs leading to misclassification for a particular machine learning model. In practice, adversarial examples enable dangerous real world attacks on models [Yin+21] [AGA21] [Zha+22], and there is even recent work showing that deliberate cryptographic adversarial examples can be secretly baked into a model [Gol+22]. Numerous explanations and solutions have been posed to determine why this phenomenon occurs [GSS14] [Sha+19] [SMB21] and how it can be remedied [Yua+17] [MGV18] [MSV19] [Fu+22] with various degrees of success.

Abstract interpretation [CC77] is a well explored class of techniques in formal methods for efficiently proving properties about programs, and through our project we examine and enhance existing techniques from abstract interpretation to *certify* the robustness of trained networks and even *modify training* to improve robustness. The central thesis of our approach is that there is a deep theoretical connection between the robustness of a model to adversarial examples and the margin theory of generalization that we frequently discussed in class. Specifically, we will show how abstract interpretation enables us to tightly measure the margin of a network around an arbitrary set of inputs. This measurement can be used as a robustness metric for trained networks or as a loss function to encourage robustness during training.

In our experiments, we abstract from a single input x to hull $B_\epsilon(x)$ of inputs that containing l^∞ perturbations on x of size at most ϵ. We use this abstraction function to explicitly optimize the model for a high margin ρ, when it exists. By training in this abstract domain, we successful defend against all adversarial examples within an $\infty-$ball with radius $\epsilon \leq \rho$ around our inputs.

During certification, the naïve abstraction has perfect soundness but limited completeness. It will never certify an input which has an adversarial example, however, because this abstraction is an over-approximation, there are adversarially robust inputs which cannot be certified.

We efficiency enhance the completeness of this method by breaking the naïve abstraction into a subquadratic number of smaller abstractions and certifying each individually before joining their results at the end, and we show that this modification empirically allows us to get a tighter bound on the abstract loss during training, resulting in a more accurate model. Finally we discuss additional theoretical ideas where we did not run experiments but that we believed merited inclusion in the paper.

2 Problem Statement

An *adversarial example* is a misclassified input $\tilde{x}_\epsilon$ within some bounded distance ϵ of an in-distribution input x. If, for a particular x, there does not exist a $\tilde{x}_\epsilon$ we say that our model is *adversarially robust* on input x. If there does not exist a $\tilde{x}$ for any $x \in D$, we say that our *model* is *adversarially robust*. In this paper we focus on the problem of adversarial examples resulting from ℓ_∞ distance bounded perturbations and discuss techniques for robustness certification and training with accompanying theoretical guarantees.

Robustness Certification. Let $L : \mathbb{R}^d \times Y \to \mathbb{R}$ be a loss function defined over inputs $x \in \mathbb{R}^d$, where Y is some label set. Let $B_\epsilon(x)$ be the ∞-ball around $x \in \mathbb{R}^d$ with radius ϵ. The local robustness certification problem is defined as the task of upper bounding the loss L within a ball around a fixed training point. In other words, want to compute $L^*_\epsilon(x, y)$ such that

$$L^*_\epsilon(x, y) \geq \sup_{z \in B_\epsilon(x)} L(z, y).$$

We consider $L^*_\epsilon(x, y)$ to be a good certification function based on its tightness, i.e., we want it to be as close to $\sup_{z \in B_\epsilon(x)} L(z, y)$ as possible while still being an upper bound.

Robust Training. As defined above, robustness certification gives us a way to measure how robust a network will be against adversarial attacks. A related problem is improving the degree to which the network can be certified. One way to do this is by training the network with the loss $L^*_\epsilon(x, y)$ instead of $L(x, y)$. Minimizing L^* around all training points will make the network more robust around all the training points, and hopefully will improve the robustness of the network to adversarial attacks on the test set as well. In our results, we show that this relationship is more nuanced than initial expected.

Significance. What is the value of this view of robustness? If $L^*_\epsilon(x, y)$ is low, then most points near x will receive the same label. Intuitively, this suggests a nontrivial decision boundary that is not simply memorizing the training data.

We can formalize the value of minimizing $L^*_\epsilon(x, y)$ in terms of margin bounds derived for neural networks [BFT17]. If all points in the ∞-ball around x belong to the same class, then the margin

[1] Code available at `https://github.com/viking-sudo-rm/diffai`.

ρ is at least ϵ. Their Theorem 1.1 [BFT17] presents a generalization bound for neural networks where the complexity term goes to zero with $1/\sqrt{\rho}$. Increasing ρ will decrease this complexity term, showing why agreement in the neighborhood around x suggests more robust generalization.

3 Background

Some background on abstract interpretation is necessary for understanding how we address these challenges. For this overview, we default to the notation from Mirman, Gehr, and Vechev [MGV18] and Mirman, Singh, and Vechev [MSV19] which was originally developed by Cousot [CC77].

Abstract Interpretation. Abstract interpretation is a technique from formal methods to approximate a program's semantics, maintaining soundness while trading completeness for efficiency. The semantics of a program, which in our case is the semantics of the neural network, are defined over the *concrete domain* which we denote $\mathcal{D}$ and captures all possible program traces — all possible neural network executions. Abstract interpretation enables us to move from reasoning about the execution of the network on individual inputs to reasoning about its behavior for sets of inputs.

In contrast to the concrete domain, we also define an *abstract domain* $\mathcal{D}^{\#}$, an *abstraction function*: $\alpha : \mathcal{P}(\mathcal{D}) \to \mathcal{D}^{\#}$, and a *concretization function* $\gamma : \mathcal{D}^{\#} \to \mathcal{P}(\mathcal{D})$. Each element in this abstract domain captures a set of traces, and α and γ are suitably defined so that they form a Galois Connection:

$$\langle \mathcal{D}, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{D}^{\#}, \leqslant \rangle.$$

This rigorous treatment of the semantics of programs, these domains, and functions between them, allows us to soundly reason about the general behavior of the neural network for sets of inputs. For all $x \in \mathcal{D}$, our concrete domain, when we abstract and then concretize, we arrive at a sound over-approximation of x (because $\gamma \circ \alpha$ is extensive). Formally this is $x \sqsubseteq \gamma(\alpha(x))$, and this extensivity is preserved when we take any function T in our network, and convert it into an *abstract transformer* $T^{\#}$ which preserves $T(x) \sqsubseteq \gamma(T^{\#}(\alpha(x)))$.

Abstract Interpretation for Neural Networks. The composition of abstract transformers are abstract transformers, so we can convert the model $N : \mathbb{R}^d \to \mathbb{R}^k$ into an abstract transformer $N^{\#} : \mathcal{P}(\mathbb{R}^d) \to \mathcal{P}(\mathbb{R}^k)$. Now that we have established the brief but necessary background on abstract interpretation, it is clear to see this is a natural fit for addressing the problem of testing if our model is *adversarially robust* on an arbitrary input x, and a natural extension of this allows for training an adversarially robust model on a ball of inputs with radius ϵ around x in one evaluation of the abstract transformer.

In this way we can convert our input image $x \in \mathcal{D}$ into a ball $B_\epsilon(x)$ and evaluate α to get $\mathcal{A}_\epsilon(x) \in \mathcal{D}^{\#}$, then evaluate the network abstract transformer $N^{\#}$, and concretize to get a sound over-approximation on the range of output values from every input $\tilde{x} \in B_\epsilon(x)$ in the concrete. For our approach and analysis, we use the notation of Mirman, Gehr, and Vechev and write $L^*(B_\epsilon(x), y)$ to mean the $\gamma \circ L^{\#} \circ \alpha(\langle x, y \rangle)$ where α maps $\langle x, y \rangle$ to $\langle \mathcal{A}_\epsilon(x), y \rangle$ and operates under the assumption that, $\forall \tilde{x} \in B_\epsilon(x)$, the correct classification of $\tilde{x}$ is y.

Additionally, for a particular network N with parameters θ, $\mathcal{A}_\epsilon(x)$ the tightest over-approximation of $B_\epsilon(x)$ in $D^{\#}$, and an test point $\langle x, y \rangle$, we write $L_{N_\theta}^{\mathcal{A}_\epsilon}(x, y)$ as a more explicit form of $L^*(B_\epsilon(x), y)$ without any details or arguments omitted.

These are powerful techniques for reasoning about sets of inputs, but the precision from the technique is inversely correlated with the time complexity of the evaluation of the abstract domain chosen and the abstract transformer. The model were are extending implements the the box domain which abstracts a set of points to their bounding box, and the zonotope domain which is a tighter convex abstraction of these points that requires more time but produces better over-approximations.

4 Approach

For a region $X \subseteq \mathbb{R}^d$, define $L^*(X, y)$ as the upper bound of the loss over that region given by abstract interpretation for a fixed abstract domain. Thus, $L_\epsilon^*(x, y) := L^*(B_\epsilon(x), y)$.

We expect that the bound $L_\epsilon^*(x, y)$ given by abstract interpretation becomes less tight as ϵ increases. In order to get a tighter certification, we propose estimating this quantity by first partitioning $B_\epsilon(x)$ into n smaller regions $\{B_i\}_{i=1}^n$ and computing $L^*(B_i, y)$ for region. We can then compute the potentially tighter, sound upper bound $L_{\epsilon,n}^*(x, y)$ defined as:

$$L_\epsilon^*(x, y) \geq L_{\epsilon,n}^*(x, y) = \max_{1 \leq i \leq n} L^*(B_i, y).$$

If each smaller region B_i has smaller approximation error than B, $L_{\epsilon,n}^*(B, y)$ will be a tighter bound on the adversarial loss than $L_\epsilon^*(B, y)$.

How then do we define the smaller regions B_i in terms of $B_\epsilon(x)$? A naive approach would be to divide $B_\epsilon(x)$ into k equal intervals on *each* axis. This produces a set of smaller ∞-balls partitioning $B_\epsilon(x)$. The number of balls n is k^d. Since we would like our method to work in high dimensional spaces, the exponential dependence on d makes this idea computationally infeasible.

We propose a simple extension of this idea with non-exponential dependence on d. Specifically, we assume a fixed budget n of the number of queries to L^*. First, define

$$d' \leftarrow \min\{\lfloor \log_k n \rfloor, d\}.$$

We then only cut $B_\epsilon(x)$ into smaller intervals on d' different axes rather than on all d axes. We will fix k, the number of splits per axis, to 2, in order to guarantee that as many axes are split as possible. This yields the pseudo-code in Algorithm 1.

Query Complexity. This algorithm will make at most n queries to L^*, and thus respects our desired budget constraints. This is because the partition over $B_\epsilon(x)$ has $2^{d'}$ regions. Based on the way d' is set, $2^{d'}$ is upper bounded by

$$2^{\log_2 n} = n.$$

Thus, we will perform at most n queries.

With $n = 1$, our algorithm is equivalent to the baseline of not partitioning $B_\epsilon(x)$. On the other hand, with an exponential query budget (i.e., $n \geq 2^d$) our algorithm reduces to partitioning on every axis. Thus, our method can be understood to interpolate between the single-query baseline and intractable exponential-query approach depending on the number of queries that its budget n allows.

5 Experimental Results

All experiments were completed on the CIFAR-10 dataset with a modification of `diffAI` [MGV18] [MSV19].

Algorithm 1 Pseudo-code for sliced certification.

```
function CERTIFY($x$, $y$, $\epsilon$, $n$)
    $d' \leftarrow \min\{\lfloor \log_2 n \rfloor, d\}$
    $D \leftarrow$ random set of $d'$ distinct values in $[1, n]$
    $B \leftarrow \{B_\epsilon(x)\}$
    for $j \in D$ do
        for $B_i \in B$ do
            Remove $B_i$ from $B$
            $L, R \leftarrow$ split $B_i$ in half on axis $d'$
            $B \leftarrow B \cup \{L, R\}$
        end for
    end for
    $\ell \leftarrow 0$
    for $i$ do
        $\ell \leftarrow \max\{L^*(B_i, y), \ell\}$
    end for
    return $\ell$
end function
```

Robustness Certification. We evaluate the quality of the certification bound as a function of the budget n. Taking a neural network trained by standard means, does a higher budget enable a tighter robustness guarantee?

We take the `smallConv` network from [MGV18] and evaluate our ability to produce an improved robustness guarantee relative to their method for balls of radius $\epsilon \in \{0.01, 0.02\}$. In Figure 1, we plot our bound on the loss over the ball and the runtime as a function of the number of splits our algorithm makes. Note that 0 splits corresponds to their baseline method. We find that increasing the budget for our method monotonically improves the bound on the loss we are able to certify. The runtime of the algorithm remains tractable for small numbers of splits, scaling linearly in the number of balls produced (exponentially in the number of splits).

Robust Training. Additionally, we trained the `smallConv`, `medConv`, and `bigConv` models for 100 epochs, linearly interpolating between the loss L and L^*_ϵ for various values of ϵ. Throughout the training process, we compute the accuracy and $L^*_{8/255}$ provable robustness every 5 epochs. This provable robustness is measured using $k = 4$ splits as described in the Robustness Certification procedure.

We find that each model has a clear tradeoff between robustness and accuracy. At the beginning of training, we see the accuracy quickly reach a peak and slowly decay inversely to robustness. We believe this tradeoff corresponds to an upper-bound on the richness of the hypothesis set defined by a particular network. We plot the `smallConv` resuls in figure Figure 3, and the other larger models display a remarkably similar inverse relationship.

6 The Curse of Dimensionality: Alternative Approaches

The certification (training) procedure outlined in the previous sections becomes unfeasible when the input space dimension or the network scale are sufficiently large. In these cases, the linear budget slicing of $B_\epsilon(x)$ produces subdomains that are too large to be computationally tractable.

While the technique of abstract interpretation preserves soundness, it lacks in completeness for large networks due to the magnitude of the over-approximation. There are many inputs points which are adversarially robust, but for which standard techniques in abstract interpretation are insufficient for verifying. Our technique in section 4 helped slightly to ameliorate this problem without sacrificing soundness but was very limited in its power. Here we present an alternative probabilistic approach which sacrifices soundness to achieve much stronger completeness. With this technique, for sufficient number of samples, the probability that we incorrectly certify a non-robust input becomes negligible, and we are theoretically able to certify more robust inputs than with the standard abstract interpretation.

Tighter Bounds Proof. Fix a network N, a labeled example $(\tilde{x}, \tilde{y})$, and an $\epsilon > 0$. Suppose that we want to carry out the local certification task on $B_\epsilon(\tilde{x})$, the *closed* l_∞ ball of radius ϵ around $\tilde{x} \in \mathbb{R}^d$. That is, we want to find an upper bound $L^*_\epsilon(\tilde{x}, \tilde{y})$ such that

$$L^*_\epsilon(\tilde{x}, \tilde{y}) \geq L_N^{B_\varepsilon}(\tilde{x}, \tilde{y}) = \max_{x \in B_\epsilon(\tilde{x})} L(x, \tilde{y}), \tag{1}$$

where $L^*_\epsilon(\tilde{x}, \tilde{y})$ is as close as possible to the exact worst-case adversarial loss $L_N^{B_\epsilon}(\tilde{x}, \tilde{y})$. The upper bound used in [MGV18] is $L_N^{\mathcal{A}_\epsilon}(\tilde{x}, \tilde{y})$: the approximate worst case adversarial loss on $\mathcal{A}_\epsilon(\tilde{x})$, where $\mathcal{A}_\epsilon(\tilde{x})$ is a sound approximation for $B_\epsilon(\tilde{x})$. We assume that the finite computational resources at our disposal allow us to run the abstract interpretation routines on balls of radius at most r, with $r \ll \epsilon$. Consider any point $x \in B_\epsilon(\tilde{x})$ and let $\mathcal{A}_r$ be a sound approximation of $B_r(x)$. Given a loss function L that is pointwise continuous in its first argument, define the function $\phi : B_\epsilon(\tilde{x}) \subset \mathbb{R}^d \to \mathbb{R}$ as

$$\phi(x) = \max_{z \in A_x} \gamma(L^{\#}(z, \tilde{y})), \tag{2}$$

where $A_x = \overline{\mathcal{A}_r(x) \cap \mathcal{A}_\epsilon(\tilde{x})}$ (the overbar denotes the topological closure of a set). Clearly, ϕ is well-defined, since the maximum of continuous functions over closed sets is always attained. Further, the pointwise continuity of L in its first argument implies the pointwise continuity of ϕ on $B_\epsilon(\tilde{x})$. Hence, we know that there exists an $x^* \in B_\epsilon(\tilde{x})$ such that ϕ achieves its maximum at x^*, that is, $\phi(x^*) = \max_{x \in B_\epsilon(\tilde{x})} \phi(x)$. Because the the upper bound given by abstract interpretation becomes tighter as the size of the input set is reduced, we have that

$$\begin{aligned} L_N^{\mathcal{A}_\epsilon}(\tilde{x}, \tilde{y}) &= \max_{z \in \mathcal{A}_\epsilon(\tilde{x})} \gamma(L^{\#}(z, \tilde{y})) \\ &\geq \max_{x \in B_\epsilon(\tilde{x})} \max_{z \in A_x} \gamma(L^{\#}(z, \tilde{y})) \\ &= \max_{x \in B_\epsilon(\tilde{x})} \phi(x) \\ &\geq \max_{x \in B_\epsilon(\tilde{x})} L(x, \tilde{y}) \\ &= L_N^{B_\epsilon}(\tilde{x}, \tilde{y}). \end{aligned} \tag{3}$$

As we see, $\phi(x^*)$ gives a better upper bound than the approximate worst case adversarial loss used in [MGV18], $L_N^{\mathcal{A}_\epsilon}(\tilde{x}, \tilde{y})$.

One way to approximate $\phi(x^*)$ is to use simulated annealing, which is a Markov chain Monte Carlo method for solving global optimization problems in high-dimensions. The algorithm is used to approximate the global minimum of a given function on a bounded subset of $\mathbb{R}^d$. Since in our case we are interested in the global maximum of ϕ on $B_\epsilon(\tilde{x})$, we define the function $h(x) = -\phi(x)$. Then, the global minumum of h will correspond to the global maximum of ϕ. Simulated annealing will produce a sequence of points $x_k \in B_\epsilon(\tilde{x})$ such that $x_k \to x^*$ as $k \to \infty$ (where convergence will occur in some appropriate sense). In the

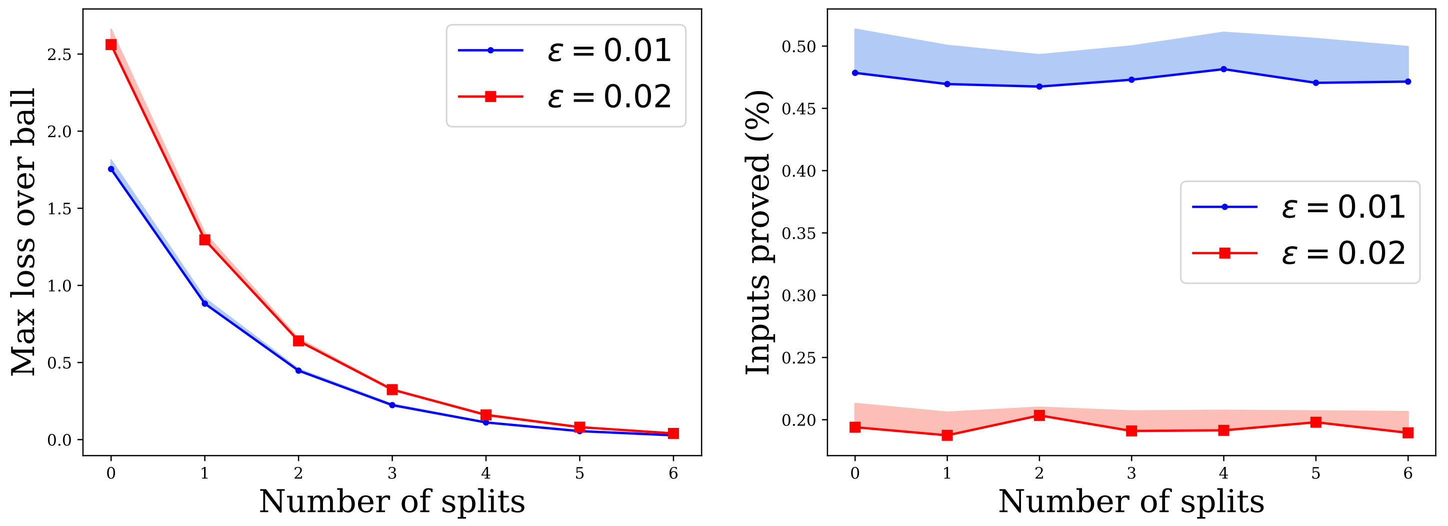

Figure 1: Certification results for $\epsilon \in \{0.01, 0.02\}$ balls. Left: upper bound on max loss over the ∞-ball as a function of the number of times we split the ball during certification. Right: the fraction of the inputs that can be certified by our method as a function of budget. The shaded ranges in both plots show the variance over the random seed used to choose the split axes.

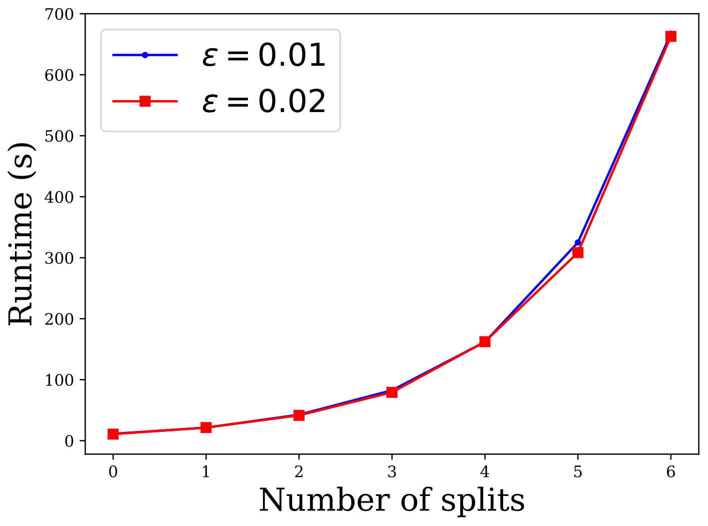

Figure 2: Mean runtime results for $\epsilon \in \{0.01, 0.02\}$. The runtime of our algorithm does not depend on the ball radius ϵ.

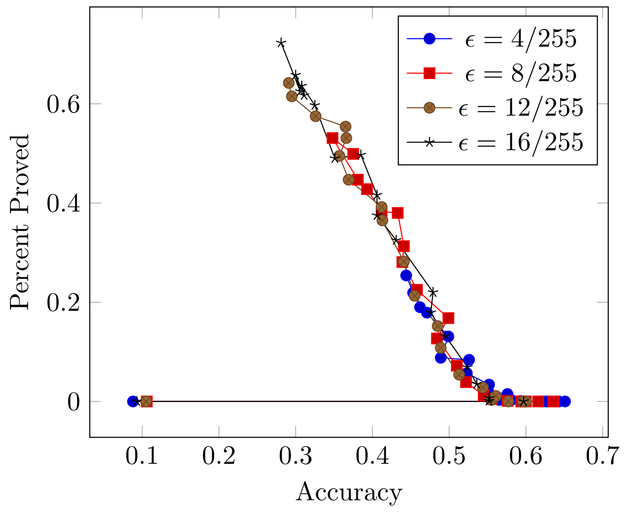

Figure 3: Accuracy vs $L^*_{8/255}$ Percent Proved over 100 epochs of training for `smallConv` according to the scheme discussed in section 5 for training $\epsilon \in \{4/255, 8/255, 12/255, 16/255\}$.

following subsection, we provide some background on the general simulated annealing algorithm.

Simulated Annealing. Suppose that we want to find the global minimum of a given continuous function $h(x)$ defined on a bounded set $B \subset \mathbb{R}^d$. Let x^* denote a point that attains the global minimum of h in B, that is, $x^* = \arg\min_{x\in B} h(x)$. The idea of simulated annealing is to use the Metropolis-Hastings algorithm [Has70] to sample from the distribution with density

$$\rho(x) = \frac{1}{Z} e^{-\beta h(x)}$$

where $\beta > 0$ is an *inverse temperature*, h is the *energy landscape* we want to minimize. x is commonly referred to as 'state', and Z is an unknown normalization constant. One of the key aspects of Metropolis-Hastings sampling algorithm is that it does not require any knowledge of Z.

In the statistical physics literature, h is referred to as 'energy'. We say that $h(x)$ is the energy corresponding to state x. Note that high energy energy states have low probability, while low energy states are the most likely ones. The constant β determines how much energy is preferred by the system. When β is large, the system has a strong preference for low energy states: states x with small energy $h(x)$ will be dominant and most of the probability will be concentrated around the wells of h. When beta is small, the density ρ is 'broad' and no state will dominate the others. We can think of the width (variance) of the peaks of the density ρ as being proportional to the *temperature* $T = 1/\beta$. Simulated annealing requires a *cooling schedule* $\{\beta_k\}_{k=0}^{\infty}$ such that $\beta_k \to \infty$ in probability, as $k \to \infty$. A common choice is some deterministic sequence of numbers $\beta_0 < \beta_1 < \beta_2 < ... < \beta_k < ...$. The cooling schedule is used to sequentially sample the density

$$\rho_k(x) = \frac{1}{Z} e^{-\beta_k h(x)}.$$

Then, the simulated annealing algorithm constructs a sequence of *states* $X_0, X_1, X_2, ...$ and a sequence of *proposal points* $Y_1, Y_2, Y_3, ...$ iteratively, as follows. An initial state X_0 is given. Having generated $X_1, X_2, ..., X_k$, propose $Y_{k+1} \sim N(X_k, r^2 I)$, where r is a fixed parameter. Then we set

$$X_{k+1} = \begin{cases} Y_{k+1} & \text{with probability } a(X_k, Y_{k+1}, \beta_k) \\ \\ X_k & \text{with probability } 1 - a(X_k, Y_{k+1}, \beta_k), \end{cases}$$

where

$$a(x, y, \beta) = \begin{cases} \exp\Big[-\beta\Big(h(y) - h(x)\Big)\Big] & \text{if } h(y) > h(x) \\ \\ 1 & \text{if } h(y) \le h(x). \end{cases} \tag{4}$$

The probability $a = a(x, y, \beta)$ of accepting the proposal point y, given the current state x and the current inverse temperature β, is called *acceptance probability.* Note that the acceptance criterion (1) is an instance of the Metropolis-Hastings criterion. It is proved in [Bél92] that, under mild assumptions, $h(X_k)$ converges in probability to $h(x^*)$, which in turns implies that X_k also converges in probability to x^*. Under the same assumptions, $\min_{1\le k\le n} h(X_k)$ also converges, almost surely, to $h(x^*)$.

Note that simulated annealing is not equivalent to gradient descent. In fact, the acceptance probability (4) shows that there is a non-zero probability of accepting worse solutions as the space is explored. However, as the inverse temperature β_k increases, this probability becomes smaller and smaller. Accepting worse solutions allows for more extensive search for the global optimum. In this way, simulated annealing overcomes one of the common pitfalls of gradient descent, which may converge to a local minimum rather than a global one.

7 Conclusion

From our experiments, we demonstrated empirically that there is a connection between margin and generalization for neural networks, and that margin maximization (with careful attention paid to loss in model correctness), is one key to understanding and combating adversarial examples. We were able to enhance existing work in abstract interpretation and apply it to neural network training and evaluation for CIFAR-10. Through slicing we enhanced the completeness of the abstract interpretation techniques used, we evaluated the networks for their capacity to balance correctness and robustness, and we proved an efficient statistical approach using simulated annealing to estimating the robustness of large models where current techniques fail completely. Potential future work in this space includes developing better efficient abstract interpretation domains and techniques, more carefully analyzing various training schemes, and extending this analysis to a larger variety of network types.

References

[AGA21] Kshitiz Aryal, Maanak Gupta, and Mahmoud Abdelsalam. *A Survey on Adversarial Attacks for Malware Analysis.* 2021 (cit. on p. 1).

[Bél92] Claude J. P. Bélisle. *Convergence Theorems for a Class of Simulated Annealing Algorithms on R^d.* 1992 (cit. on p. 5).

[BFT17] Peter Bartlett, Dylan J. Foster, and Matus Telgarsky. *Spectrally-normalized margin bounds for neural networks.* 2017 (cit. on pp. 1, 2).

[CC77] Patrick Cousot and Radhia Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". In: POPL '77. Association for Computing Machinery, 1977 (cit. on pp. 1, 2).

[Fu+22] Shaopeng Fu et al. *Robust Unlearnable Examples: Protecting Data Against Adversarial Learning.* 2022 (cit. on p. 1).

[Gol+22] Shafi Goldwasser et al. *Planting Undetectable Backdoors in Machine Learning Models.* 2022 (cit. on p. 1).

[GSS14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples.* 2014 (cit. on p. 1).

[Has70] W. K. Hastings. *Monte Carlo Sampling Methods Using Markov Chains and Their Applications.* 1970 (cit. on p. 5).

[MGV18] Matthew Mirman, Timon Gehr, and Martin Vechev. "Differentiable Abstract Interpretation for Provably Robust Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning.* Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 3578–3586 (cit. on pp. 1–3).

[MSV19] Matthew Mirman, Gagandeep Singh, and Martin Vechev. *A Provable Defense for Deep Residual Networks.* 2019 (cit. on pp. 1, 2).

[Sha+19] Adi Shamir et al. *A Simple Explanation for the Existence of Adversarial Examples with Small Hamming Distance.* 2019 (cit. on p. 1).

[SMB21] Adi Shamir, Odelia Melamed, and Oriel BenShmuel. *The Dimpled Manifold Model of Adversarial Examples in Machine Learning.* 2021 (cit. on p. 1).

[Yin+21] Bangjie Yin et al. *Adv-Makeup: A New Imperceptible and Transferable Attack on Face Recognition.* 2021 (cit. on p. 1).

[Yua+17] Xiaoyong Yuan et al. *Adversarial Examples: Attacks and Defenses for Deep Learning.* 2017 (cit. on p. 1).

[Zha+22] Jindi Zhang et al. "Evaluating Adversarial Attacks on Driving Safety in Vision-Based Autonomous Vehicles". In: *IEEE Internet of Things Journal* 9.5 (Mar. 2022), pp. 3443–3456 (cit. on p. 1).