A BRANCH AND BOUND FRAMEWORK FOR STRONGER ADVERSARIAL ATTACKS OF RELU NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Strong adversarial attacks are important for evaluating the true robustness of deep neural networks. Most existing attacks find adversarial examples via searching in the input space, e.g., using gradient descent. In this work, we formulate an adversarial attack using a branch-and-bound (BaB) procedure on ReLU neural networks and search adversarial examples in the activation space corresponding to binary variables in a mixed integer programming (MIP) formulation. This attack formulation can be used to tackle hard instances where none of the existing adversarial attacks can succeed. Existing attacks using this formulation rely on generic solvers which cannot exploit the structure of neural networks and also cannot utilize GPU acceleration, so they are mostly limited to small networks and easy problem instances. To improve its scalability and practicability, we propose a top-down beam-search approach to quickly identify the subspace that may contain adversarial examples. The search utilizes the bound propagation based neural network verifiers on GPUs to rapidly evaluate a large number of searching regions, which is not possible in generic MIP solvers. Moreover, we exploit the fact that good candidates of adversarial examples can be easily found via gradient based attacks, and build an *adversarial candidates pool* to further guide the search in activation space via *diving* techniques. Additionally, any candidate adversarial examples found during the process are refined using a bottom-up large neighbourhood search (LNS) guided by the candidates pool. Our adversarial attack framework, BaB-Attack, opens up a new opportunity for designing novel adversarial attacks not limited to searching the input space, and enables us to borrow techniques from integer programming theory and neural network verification to build stronger attacks. In experiments, we can successfully generate adversarial examples for hard input instances where existing strong adversarial attacks fail, and outperform off-the-shelf MIP solver based attacks in both success rates and efficiency. Our results further close the gap between the upper bound of robust accuracy obtained by attacks and the lower bound obtained by verification.

1 INTRODUCTION

Adversarial attacks to deep neural networks aim to find adversarial examples (Szegedy et al., 2013), which are close to benign inputs in certain distance metric yet trigger wrong behavior of neural networks (Carlini & Wagner, 2017; Madry et al., 2018; Athalye et al., 2018; Croce & Hein, 2020b). Adversarial attacks are important tools to gauge the empirical robustness of deep neural networks. Finding an adversarial example can be generally formulated as an optimization problem:

$$x_{\text{adv}} = \arg\min_{x \in \mathcal{C}} f(x) \tag{1}$$

where $\mathcal{C}$ is the set of allowable input domains (e.g., an ℓ_∞ or ℓ_2 norm ball around input x), and $f(x)$ is a criterion to indicate a successful attack: $f(x) < 0$ indicates a successful attack. As $f(x)$ is usually a differentiable function involving a highly non-convex neural network, solving Eq 1 is usually challenging. A straightforward way of solve Eq. 1 is via first-order constrained optimization methods, such as projected gradient descent (Madry et al., 2018). However, because it is generally impossible to find the global minimum of Eq. 1 using gradients, the success of such attacks heavily depends on initialization and whether useful information is provided by gradients. A purely gradient based attack may easily fail due to gradient masking (Papernot et al., 2016; Tramèr et al., 2017). Although some gradient-free based attacks were proposed (Brendel et al., 2018; Cheng et al., 2018;

Alzantot et al., 2019; Andriushchenko et al., 2020), they were mostly based on certain random or heuristic search on the input space and can often miss adversarial examples due to non-convexity.

Our way to tackle the non-convexity is to solve Eq. 1 using an binary encoding of ReLU neurons (Ehlers, 2017; Tjeng et al., 2019a), via binary variables s_i: $s_i = 0$ indicates ReLU's input is negative given a certain input x and $s_i = 1$ otherwise. Each setting of $\{s_1, \cdots, s_N\} \in \{0, 1\}^N$ (called activation space) determines an unique linear region of the input space that may contain adversarial examples. Under this formulation, an MIP solver can exhaustively search the activation space and find the global minimum of Eq. 1, implying that it can always find the adversarial example if one exists given sufficient time. However, using generic MIP solvers is inefficient and often requires that the networks are very small or trained using a certified defense (Wong & Kolter, 2018; Raghunathan et al., 2018a; Xiao et al., 2019).

Our paper aims to build a strong adversarial attack using a branch and bound procedure to systematically search the neuron activation space and is more scalable than generic MIP solvers. We focus on *solving hard instances*, where none of existing adversarial attacks based on searching the input space are successful but no verifier can prove their robustness. We aim to scale up the integer programming based approach for strong adversarial attacks to further close the gap between the upper bound of robust accuracy from adversarial attacks and the lower bound of robust accuracy from verification. The main technical novelty of this work includes:

- We apply the GPU-accelerated bound propagation based methods (Wang et al., 2021; Xu et al., 2020; Zhang et al., 2018; Wong & Kolter, 2018), which were originally developed for neural network verification, to the adversarial attack setting. These methods allow us to quickly examine a large number (e.g., thousands) of regions in activation space (which is hardly possible in generic MIP solvers) and rule out the regions that are guaranteed to contain no adversarial examples. They also guide us to the regions that are more likely to contain adversarial examples.
- We employ a beam-search scheme that quickly explores the activation space in a top-down manner and identifies the most promising region. Unlike the best-first search scheme used in many neural network verifiers, our beam-search approach can quickly go deep in the search tree and greatly reduce the number of unstable neurons (i.e., the neurons with s_i that can be set to either 0 or 1). A smaller sub-MIP can be then created on-the-fly to search adversarial examples in the reduced regions only with much less integer variables.
- Searching adversarial examples in the activation space opens up the opportunity to borrow many techniques from the integer programming literature to the adversarial attack setting, which was never demonstrated in the literature. Specifically, we adopt *diving*, a popular technique in integer optimization that heuristically fixes integer variables to reduce the search space. Our diving technique incorporates information from cheaply generated candidate adversarial examples in a *candidates pool*, which is not possible in a generic MIP solver.
- We further apply large neighbourhood search (LNS) techniques on promising adversarial example candidates by freeing the selections of certain neuron activations and explore activation space in a bottom-up manner. The neighbourhood space is suggested by the adversarial candidates pool.
- Our new attack framework, BaB-Attack, is designed to tackle hard instances where existing strong adversarial attacks (such as AutoAttack) cannot succeed. Despite being more expensive than gradient based methods on the input space, BaB-Attack is an order of magnitude faster than MIP-based attacks on hard instances, and can find more adversarial examples that cannot be discovered by any existing attacks, including those based on MIP solvers. On four benchmarking models for MNIST and CIFAR-10 datasets, our attack further close the gap between the upper bound of robust accuracy from adversarial attacks and the lower bound from verification.

2 BACKGROUND

Notations. We define a L layer feed-forward neural network function as $f : \mathbb{R}^{n_0} \to \mathbb{R}$. $f(x) = z^{(L)}(x)$, where pre-activation $z^{(i)}(x) = \mathbf{W}^{(i)}\hat{z}^{(i-1)}(x) + \mathbf{b}^{(i)}$ with i-th layer weight matrix $\mathbf{W}^{(i)}$ and bias $\mathbf{b}^{(i)}$, post-activation $\hat{z}^{(i)}(x) = \text{ReLU}(z^{(i)}(x))$, and input $\hat{z}^{(0)}(x) = x$. For a simpler presentation, we assume $f(x)$ is a binary classifier and benign input x_0 has $f(x_0) > 0$. An attacker seeks to minimize $f(x)$ within a perturbation set $\mathcal{C}$ to make $f(x)$ negative. We can attack a multi-class classifier by considering each pair of target label and ground-truth label where f is the margin between them. We consider attacks to each target label individually similar to (Gowal et al., 2019b).

2.1 THE MIP FORMULATION FOR ADVERSARIAL ATTACK

Due to the high non-convexity of Eq. 1, existing attacks searching on input space can miss adversarial examples even they do exist. A "complete" adversarial adversarial attack, on the other hand, can locate the global minimum to Eq. 1 given sufficient running time and can be used for hard instances where no existing attack can succeed. In other words, it guarantees to find all unsafe testing instances. Tjeng et al. (2019a) formulated the complete adversarial attack and verification of the network into a mixed integer programming (MIP) problem in Eq. 2, and relied on existing MIP solver to solve this problem. In this paper, we refer to it as the "MIP attack". Specifically, the MIP formulation associates each ReLU neuron $\hat{z} = \text{ReLU}(z)$ with an integer indicator variable $s = \mathbb{1}_{z \geq 0}$, indicating the status of a ReLU neuron as (1) *inactive*: when $s = 0$, $\hat{z} = 0$ or (2) *active*: when $s = 1$, $\hat{z} = z$. Adversarial attack can thus be seen as an optimization problem by choosing the s_i for each neuron i that minimizes $f(x)$. Here l, u are pre-computed *intermediate bounds* respectively on each pre-activation variable z such that $l \leq z(x) \leq u$ for any $x \in \mathcal{C}$. We detail how to compute these intermediate bounds in Section 2.3. Notably we say a ReLU neuron is *unstable* if its intermediate bounds $l < 0 < u$, meaning it could be either active or inactive within $x \in \mathcal{C}$.

$$
\begin{aligned}
\min f(x) \ \text{ s.t. } \ & z^{(i)}(x) = \mathbf{W}^{(i)} \hat{z}^{(i-1)}(x) + \mathbf{b}^{(i)}; \quad x \in \mathcal{C}; \\
& \hat{z}^{(i)}(x) \geq z^{(i)}(x); \ \ \hat{z}^{(i)}(x) \leq u^{(i)} s^{(i)}; \ \ \hat{z}^{(i)}(x) \leq z^{(i)}(x) - l^{(i)}(1 - s^{(i)}); \\
& z^{(i)}(x) \in [l^{(i)}, u^{(i)}]; \quad s^{(i)} \in \{0, 1\}
\end{aligned}
\tag{2}
$$

The solving time of MIP attack exponentially increases with the number of unstable ReLU neurons. Therefore, it can hardly scale to large networks, and can take hours to run even on a small network, unless the network is trained using a strong regularization such as in certified dense (Wong & Kolter, 2018; Xiao et al., 2019).

2.2 SEARCHING IN THE ACTIVATION SPACE VIA BRANCH AND BOUND (BAB)

Given the integer formulation in equation 2, we can view a neural network function in the *activation space* s_i rather than in the input space $x \in \mathcal{C}$. Each compatible setting of fixed s_i determines an unique linear region in the input space. For example, propagating an adversarial example x_{adv} through the network determines the s_i for each ReLU neuron, by checking the sign of the input for each ReLU neuron. Each s_i is either fixed at 0 or 1, and the vector $\{s_1, \cdots, s_N\}$ determines the linear region that contains this adversarial example where N is the total number of ReLU neurons.

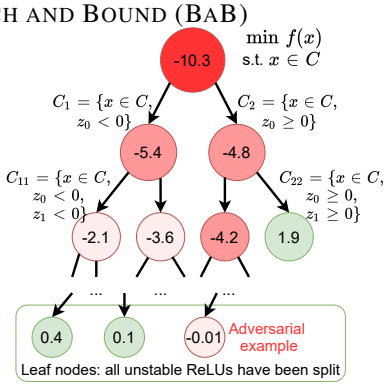


Figure 1: Brand and bound search tree for verification. Each node represents a subdomain and the value is its estimated lower bound (worse lower bounds in darker red; verified subdomains in green). The network performs linearly in leaf node subdomains. The BaB verification process always explores subdomains with worst estimated lower bounds first, tending to perform breath search first. However, we can only detect adversarial examples on leaf nodes for most cases and thus requires depths search first for efficiency.

Instead of directly finding adversarial examples in input space, we can formulate the attack as identifying the linear region (parameterized by activation pattern $\{s_1, \cdots, s_N\}$) that contains an adversarial example. Once $\{s_1, \cdots, s_N\}$ is fixed, an adversarial example can be provable obtained due to the linear property of such region. As a result, we can guarantee to find adversarial examples if we visit all the combinations. Here, unstable ReLU neurons are crucial as it determines the free dimensions to search. When considering the adversarially perturbed inputs inside an entire input domain $\mathcal{C}$, the s_i of unstable ReLUs can take either 0 or 1 while the rest stable ReLUs have fixed s_i. Without any unstable ReLUs, the network becomes a single linear function in $\mathcal{C}$ and we can easily locate adversarial examples if exist.

A similar searching problem in activation space is well explored recently for neural network verification which aims to provide a relaxed sound lower bound for the final objective $\underline{f} \leq \min_{x \in \mathcal{C}} f(x)$. If $\underline{f} \geq 0$ then the network is verified to be safe. Customized Branch and bound (BaB) methods achieve great success in accelerating such a searching process. In specific, BaB for verification

relies on cheap bound propagation methods (Section 2.3) which will relax and estimate the intermediate bounds l, u and a lower bound of final objective $\underline{f}$. We illustrate how to build BaB search tree for verification in Fig. 1. Since each ReLU is piece-wise linear, given a selected unstable ReLU neuron $\hat{z}_j^{(i)} = \text{ReLU}(z_j^{(i)})$, we can then split the original verification domain C into two independent subdomains $\mathcal{C}_1 = \{x \in \mathcal{C}, z_j^{(i)} \geq 0\}$ and $\mathcal{C}_2 = \{x \in \mathcal{C}, z_j^{(i)} < 0\}$. In activation space $\{s_1, \cdots, s_N\}$, this means to reduce the searching space by fixing $s_j^{(i)}$ to be 0 and 1 for subdomains $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. For each subdomain, we can do another round of bound propagations and obtain refined lower bounds for each of them $\underline{f}_{\mathcal{C}_1} \leq \underline{f}$ and $\underline{f}_{\mathcal{C}_2} \leq \underline{f}$, eliminating the nonlinearity introduced by ReLU $z_j^{(i)}$. We can recursively branch on unverified subdomains until all the divided subdomains are verified. At the worst case, all unstable ReLU neurons will be split and the network performs linearly in each subdomain (leaf nodes in Fig. 1). The lower bounds by bound propagation methods will become precise. Negative lower bounds at leaf node indicate adversarial examples while positive ones mean to be safe.

We noticed such BaB searching with ReLU neuron split in verification can also be extended for adversarial attacks with the MIP formulation mentioned in Section 2.1. However, as illustrated in Fig. 1, BaB based verification is a best first search process, i.e., a typical BaB based verifier first picks the subdomains with the worst estimated lower bounds which tend to have lower depth in the search tree first. On the other hand, adversarial example searching process prefers depth first. This is because deeper nodes have less non-linearity introduced by unstable ReLUs in each subdomain leading to tighter lower bound approximations and higher chance in locating adversarial examples. Since BaB searching space is exponentially increasing to the number of unstable ReLUs, BaB based adversarial attack is a very challenging problem. Simply applying the existing BaB process for verification will lead to very poor adversarial attack performance.

2.3 Bound Propagation for Branch and Bound of Neural Networks

As discussed before, BaB based verifiers usually rely on a fast procedure to provides lower bounds for each subdomain. One popular choice is bound propagation method (Zhang et al., 2018; Wong & Kolter, 2018) which can give a sound lower bound of final objective $\underline{f}$ in a fast manner. Essentially, they will relax each nonlinear ReLU neurons into convex domains (Ehlers, 2017) and propagate such convex domains layer by layer through the network while maintaining sound bounds for each intermediate neuron. The whole verification problem relaxed this way can be formalized as a linear programming (LP) problem. It essentially is a relaxed but cheaper version of the original MIP formulation in Eq. 2 where integer indicators s are set free to be continuous floating point values from 0 to 1. CROWN (Zhang et al., 2018; Xu et al., 2021) is a representative for solving this problem fully accelerated by GPUs benefited from automatic differentiable computation graph implementation (Xu et al., 2020). BaB based verifier using α, β-CROWN (Wang et al., 2021), an optimized version of CROWN, achieves the state-of-the-art verification performance.

Interestingly for adversarial attacks, the lower bounds by bound propagation can also serve as a useful guidance towards the leaf node linear space containing adversarial examples. This is because parent subdomains always have smaller lower bounds than children's while the leaf nodes containing adversarial examples have negative lower bounds. We show how to leverage such important guidance for adversarial attacks in Section 3.2.

3 Method

3.1 Overview of BaB attack

To systematically search adversarial examples in activation space, we must explore the search tree composed of unstable neurons and enumerate all leaf nodes. Although the worst case search time complexity is exponential in the numbers of unstable ReLU neurons (visiting every leaf node of the tree), practically, if the right search procedure is chosen, only a small fraction of nodes need to be visited to find an adversarial example. In this paper, we propose BaB attack, specializing the BaB searching strategy over the activation space for the purpose of adversarial attacks. The search is well guided by (1) a top-down beam-search thread accelerated on GPUs and (2) a bottom-up search thread on CPUs for local search with sub-MIP solvers. The top-down and bottom-up searches run

in parallel threads and they both benefit from the *adversarial candidates pool*, which contains many examples that have small margins between the ground-truth label and the target label, but are not yet able to change model prediction. The activation patterns of these candidates can be used as extra information to guide the search towards adversarial examples.

Top-Down Search on GPUs. The Top-Down search thread traverses the search tree using a beam search procedure (detailed in Section 3.2), guided by lower bounds computed by a state-of-the-art GPU-based neural network verifier, for example, α,β-CROWN (Wang et al., 2021; Xu et al., 2021; Zhang et al., 2018). The insight is to accelerate the search process by prioritizing suspicious subdomains with small lower bounds, quickly excluding the tree branches that are guaranteed with no adversarial examples. For each subdomain in beam search, we further reduce the searching space using *diving* heuristics with adversarial common patterns (Section 3.3). The adversarial patterns are extracted from a well-maintained adversarial candidates pool continuously updated in beam search. The most promising subdomains in BaB are periodically selected to generate sub-MIP problems with much fewer integer variables than the original problem, which can be solved quickly.

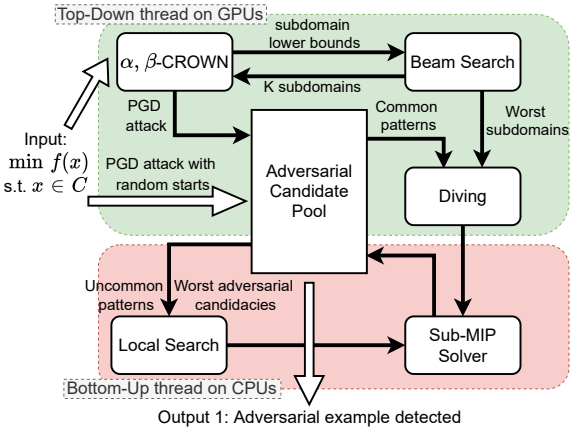


Figure 2: Overview of BaB attack.

Bottom-Up Search on CPUs. Besides the Top-Down beam search with diving, we also have a Bottom-Up thread running in parallel on CPUs, where we use local search techniques (detailed in Section 3.4) to generate a neighborhood around the most promising adversarial candidates. The local search is done by constructing a sub-MIP with most activation pattern s_i fixed to either 0 or 1 as in an adversarial candidate, and a few selected ones are relaxed to $\{0, 1\}$. Although many generic MIP solvers also use a similar technique, our selections of relaxed neurons are based on the uncommon patterns in the adversarial candidates pool, which are not available to generic MIP solvers.

Note that many existing neural network verifiers (Bunel et al., 2020; Wang et al., 2021) also utilize a BaB search tree; however, these verifiers do not aim to find adversarial examples by diving into the leaf nodes of the tree. For verification, the search procedure mostly prefers breadth search (Section 2.2), and beam search is generally unhelpful. Our procedure of building the search tree is thus specialized for finding adversarial examples and different from these works.

3.2 Beam Search Guided by Neural Network Verifiers

The activation space can be quite large (e.g., can be thousands or more unstable neurons) when searching for adversarial examples, and adversarial examples are at the leaf levels of the search tree. Searching directly from the root node and traversing the search tree in an exhaustive manner (such as BFS or DFS) can be quite insufficient, and our beam search based procedure is essential for quickly identifying the subdomains in the search space that may contain adversarial examples. Our specialized solvers with massive parallelization on GPUs brings us great advantages over MIP attacks solved with existing MIP solvers, which conduct branching and bounding on CPUs. In practice, we can visit several orders of magnitude more subdomains with beam search.

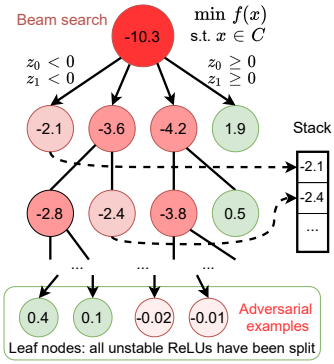


Figure 3: Beam search: We propagate K subdomains in parallel on GPUs, take the softmax of lower bound values, select K_0 worst nodes for next search step.

At the root node in the search tree, our beam search procedure first selects D unstable neurons and expands the tree by D levels, yielding 2^D subdomains. Each subdomain represents a configuration of neuron activations $\{s_1, \cdots, s_D\}$. For each explored subdomain, bound propagation method α,β-CROWN will provide us a sound lower bound indicating how difficult the subdomain is to be verified. To

a certain extent, the lower bounds can also serve as heuristic scores quantifying the probability of adversarial examples (the smaller the lower bound value, the higher chance an adversarial example exists). Instead of directly continuing beam search with the selected K_0 subdomains with the worst lower bounds, we randomly sample K_0 subdomains according to the probabilistic scores. The probabilistic scores are computed by taking the softmax of all the subdomain lower bounds from the current step. This way, the subdomains with worse lower bounds will have higher chances to be explored first, but the remaining ones will still have the possibility to be selected to diversify the selections. The key insight behind such a design is to prevent the beam search from getting stuck at one single local optimum, thus and finding adversarial examples with fewer iterations. The unexplored subdomains will be stored in a stack and will be backtracked to and explored later when all previous subdomains are verified safe or become infeasible.

Beam search with back-tracking can achieve completeness (Zhou & Hansen, 2005), i.e., given sufficient time, it will systematically visit all leaf nodes, and guarantee to either locate adversarial examples and prove the network unsafe, or prove the network safe if no adversarial example is found on any of the leaf nodes. However in practice, due to a large number of unstable neurons (hundreds to thousands), achieving this completeness is often impractical and we mostly focus on searching adversarial examples as quickly as possible rather than exhaustively visiting every leaf node.

3.3 Diving in Branch and Bound with Adversarial Common Patterns

Diving is a common strategy in traditional MIP solvers, able to quickly uncover feasible solutions (Berthold, 2006). Specifically for adversarial attacks, we noticed that the diving heuristics used for fixing unstable ReLUs can be greatly improved by considering common patterns extracted from adversarial candidates pool. We will first describe our customized diving algorithm and then provide the details on how to extract the common patterns with a proper common pattern threshold.

Diving with adversarial common patterns. As illustrated in Fig. 4, diving with adversarial patterns allows us to quickly dive to a deep subdomain in the BaB search tree that has a higher chance of containing real adversarial examples. Specifically, before each step of beam search for the selected subdomains (e.g., subdomains with the lower bounds -3.6 and -4.2), we will first perform diving with adversarial common patterns to quickly locate suspicious subdomains (dive node with lower bound -1.8) and check the existence of adversarial examples by constructing a much easier sub-MIP problem (adversarial example found at leaf node with output -0.01). If nothing is detected, we will add the leaf nodes into adversarial candidates pool and then backtrack and continue with beam search for the targeted subdomains (subdomains with lower bounds -3.6 and -4.2). Note that the beam search will not be blocked when solving the constructed sub-MIP since it will be added into a sub-MIP pool and handled by the other Bottom-Up threads in parallel on CPUs (Section 3.4).

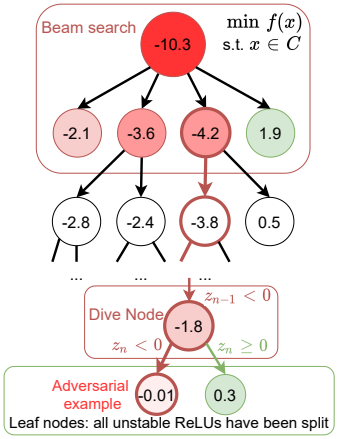


Figure 4: With adversarial common patterns, we can dive to promising subdomains and significantly reduce BaB search space. Here the diving path is highlighted in red and the subtree of dive node will be solved as a sub-MIP in the other Bottom-Up thread in parallel.

Formally, for each unstable ReLU neuron $\hat{z}_t = \text{ReLU}(z_t)$ with the associated integer indicator s_t, it has two neuron patterns $s_t = 1$ and $s_t = 0$ indicating if its pre-ReLU activation z_t is positive or negative. We say one of the two patterns is an adversarial common pattern if the majority of the adversarial candidates agree on this pattern. Recall that the adversarial candidates are the samples producing the lowest final objectives in Eq. 1 found in BaB process. That means the subset of data points with the adversarial common pattern $\mathcal{C}_{\text{adv}} = \{x \in \mathcal{C}, z_t < 0\}$ tend to have lower output values leading to a higher probability of containing real adversarial examples.

adversarial candidates pool dynamically filled with PGD attacks. To obtain useful adversarial common pattern, we will maintain an *adversarial candidates pool* with fixed length. It keeps the N most promising adversarial candidates. A larger value of N will lead to more precise adversarial common patterns with at a greater cost of the matching procedure. The pool is initialized with perturbed samples bounded within $\mathcal{C}$ found by PGD attack with N_0 random starts. For each subdomain explored in beam search with bound propagation method α, β-CROWN, we can also extract a ref-

erence point by taking the worst case solution to the relaxed space. We will perform a PGD attack on this reference point to find an adversarial candidate and add it to the adversarial candidates pool. When the subdomain is a leaf node, i.e., all unstable ReLU neurons are split, the worst case solution to the relaxed space is exactly the desired integer solution, and thus PGD attack is not needed.

Common pattern threshold. A predefined threshold $T \in [0, 1]$ controls the tolerance of pattern agreements. A pattern (either $s_t = 0$ or $s_t = 1$) is a common pattern if over T percent of adversarial candidates have the same pattern on this neuron. All unstable ReLU neurons will be fixed with common patterns if $T = 0.5$. One can adjust this threshold to control the aggressiveness for diving.

3.4 Large neighborhood Search (LNS)

Large neighborhood Search (Walser, 2003; Schrijver, 2003) is a general local search heuristic used in MIP solver: One defines a neighborhood of a reference point (a feasible solution), quickly finds the optimum for the objective function by constructing the sub-MIP problem with neighborhood constraints, and then uses this optimal solution as the next reference point. However, traditional local search in MIP solvers has no guidance regarding suspicious subdomains for adversarial examples and thus has very limited benefits due to the huge search space.

In the design of our BaB attack, we extend the general idea of LNS to a specialized local search for adversarial attacks. Specifically, we select the worst adversarial candidates in the pool as the reference point and then use the adversarial uncommon patterns to unbound certain unstable ReLU neurons to generate sub-MIP problems. The optimal solution to each sub-MIP will be added back to adversarial candidates pool again waiting for another round of local search. Last, we maintain a sub-MIP pool containing all the subdomains generated by local search, and an independent Bottom-Up thread will solve these sub-MIP problems in parallel on CPUs.

Insights of local search for BaB attack. We illustrate the key insights of our local search in BaB attack in Fig. 5. In many cases, there exist adversarial examples nearby the adversarial candidate found in beam search and diving. However, they are missed due to extra unstable neurons fixed by adversarial common patterns in diving ($z_{n-1} < 0$). On the other hand, beam search can still find the adversarial examples but it will take too much time to reach that leaf node, causing a timeout. Our Bottom-Up local search can solve such a problem and accelerate adversarial attacks. It will start from the worst adversarial candidate (leaf node with output 0.01 as the start reference point) and unbound the unstable ReLU neurons according to adversarial uncommon patterns ($z_n < 0$ & $z_{n-1} < 0$). With the relaxed neighborhood (subdomain with lower bound -2.7), we can construct a sub-MIP problem and quickly locate the adversarial example using a MIP solver.

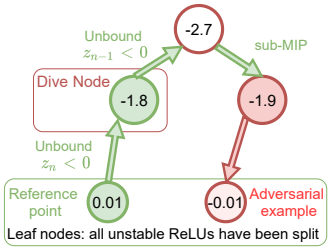
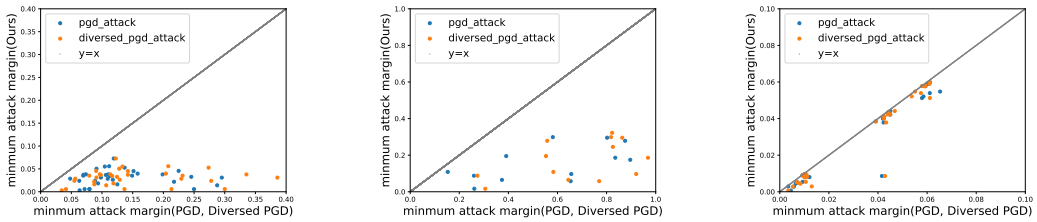


Figure 5: An illustration of local search. We use the worst adversarial candidate as the starting reference point, unbound unstable ReLU neurons based on uncommon patterns, and construct a sub-MIP to solve.

Unbounding ReLUs with adversarial uncommon patterns. The performance of local search heavily depends on the strategies in selecting which ReLU neurons to unbound for constructing the sub-MIP. We noticed this can also be guided by historical adversarial candidates. Formally, among all the ReLU neurons split on reference points, we will find the ReLU neurons where adversarial candidates disagree the most and we define the corresponding patterns as *adversarial uncommon patterns*. Note that the number of unbounded ReLU neurons can also be adjusted by a common pattern threshold. A larger common pattern threshold will lead to more unbounded ReLU neurons from reference points and a higher chance in finding adversarial examples, but it will also introduce larger search space and thus higher cost to solve the sub-MIP.

4 Experiments

Setup For all 10,000 test examples of each model, we first run three commonly used strong adversarial attacks: a multi-targeted PGD attack (Gowal et al., 2019a) with 100 Adam steps and 30 random restarts; a multi-targeted PGD attack with Output Diversified Sampling (ODS) (Tashiro et al., 2020) with 100 Adam steps and 30 random restarts, and AutoAttack (Croce & Hein, 2020b) which is an ensemble of parameter-free attacks. The remaining robust images are then tested with α,β-CROWN

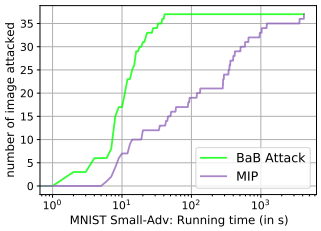


(a) MNIST CNN-A-Adv (b) MNIST Small-Adv (c) CIFAR CNN-B-Adv

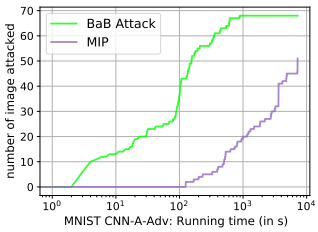
Figure 6: For examples that all attacks failed, we compare the margins between the ground-truth label and the attack target label for the adversarial candidate. A smaller margin is better. Our attack achieves noticeably smaller margins compared to PGD based attacks (autoattack does not return a valid adversarial candidate when it failed).

verifier (Wang et al., 2021; Xu et al., 2021; Zhang et al., 2018) to see if they can be verified robust so no further attack is needed. Any images that failed with the α,β-CROWN verifier are then evaluated in an MIP formulation (Tjeng et al., 2019a) solved using Gurobi. The MIP solver is used as the last resort because it is usually much more expensive than other approaches. We set an one hour timeout for MIP attack and our BaB attack, but our attack usually terminates much faster than a generic MIP solver. Among the five models, we include a relatively small MNIST network trained using adversarial training, with architecture similar to the MNIST model used in (Madry et al., 2018) but scaled down by 4X and maxpool layers removed. We include two challenging adversarially trained models for MNIST and CIFAR from (Dathathri et al., 2020), as well as two baseline CIFAR models (LP_D-CNN-A and LP_D-ResNet) from (Tjeng et al., 2019a). All attacks are ℓ_∞ norm-based with ϵ listed in Table 1.

Results Table 1 shows a detailed breakdown of all 10,000 test examples on the 5 models. Among these models, MNIST Small-Adv, CIFAR LP_D-CNN-A, and LP_D-ResNet are relatively easy, and most of datapoints can be either attacked or verified by SOTA tools. The LP_D-CNN-A and LP_D-ResNet models were among the largest ones in (Tjeng et al., 2019a), however they have become quite easy due to the recent progress of neural network verifiers - only 2 and 1 examples must be solved using MIP/BaB. On the relatively small MNIST Small-Adv model, a MIP solver can terminate within a reasonable amount of time, and we can attack the same number of images while being faster. On the CNN-A-Adv model, many instances timeout with the MIP solver and we can attack more images under less time (see Fig. 7 for time vs. number of images attacked). The CNN-B model is much harder, and we managed to solve two more images compared to MIP. For all images that none of the existing attacks work, we plot the margins between the ground-truth label and target label in Figure 6, and our method achieves a consistently smaller margin, indicating that it is a stronger attack.



(a) MNIST-SMALL-ADV



(b) MNIST-CNN-A-ADV

Figure 7: Running time vs. number of attacked images compared to MIP attack.

5 Related Work

Adversarial examples and attacks. Adversarial examples are first investigated in (Szegedy et al., 2013; Biggio et al., 2013) and they can be easily constructed by single-/multi-step gradient descent to fool regularly trained neural networks (Kurakin et al., 2016; Goodfellow et al., 2015). However, purely gradient-based methods will fail due to gradient masking and obfuscated gradients (Tramèr et al., 2017; Papernot et al., 2016; Athalye et al., 2018). A large body of white-box adversarial attacks have been proposed to solve this challenging task but have marginal improvements (Zheng et al., 2019; Gowal et al., 2019b; Wang et al., 2019). Even popular attacks like PGD (Madry et al., 2018) or CW (Carlini & Wagner, 2017) have been shown to fail leading to overestimation of robustness (Mosbach et al., 2018; Croce et al., 2020) despite their empirically good performance

	MNIST		CIFAR		
Name	Small-Adv	CNN-A-Adv	LPd-CNN-A	LPd-RES	CNN-B-Adv
Training Method	Adv. training		Certified Training		Adv. Training
Model Source	(Madry et al., 2018)	(Dathathri et al., 2020)	(Wong et al., 2018; Tjeng et al., 2019b)		(Dathathri et al., 2020)
Clean Accuracy	97.94	96.33	60.86	27.07	78.25
ℓ_∞ perturbation ϵ	0.3	0.3	2/255	8/255	2/255
Verified by α,β-CROWN (Wang et al., 2021)	8072	6699	4956	2243	4534
Verified by MIP	+104	+43	+63	+0	+2
Total Verified	8176	6742	5019	2243	4536
Attacked by Multi-targeted PGD-100 Gowal et al. (2019b)	1347	2126	1065	461	1675
Attacked by Diversified PGD (Tashiro et al., 2020)	1330	2102	1065	461	1676
Attacked by AutoAttack (Croce & Hein, 2020b)	1563	2200	1065	463	1671
Total Attacked + Verified	9735	8996	6084	2706	6212
Remaining Examples	59	637	2	1	1613
Avg. Unstable Neurons (integer variables)	619	1895	325	1586	2288
No. images attacked by MIP attack	**37**	51	2	0	1
MIP Avg. Time (s.) (excluding timed out)	439	2546	13.8	-	24
No. images attacked by our BaB attack	**37**	**66**	2	0	**3**
Our Avg. Time (s.) (excluding timed out)	**16**	**124**	**9.9**	-	**19**
Combined Attacked	37	67	2	0	3
Remaining Unsolved	22	570	0	1	1548

Table 1: Comparison between MIP attack (using Gurobi) and our BaB Attack. Both attacks focus on hard instances where none of the existing tools can attack or verify, so we filter out points that already verified or attacked using gradient based attacks. We are faster than MIP and can also find more adversarial examples.

and efficiency. Due to non-convexity of the adversarial attack objective, gradient-free methods and black-box attacks are also widely explored but mostly result in similar or worse performance to gradient-based ones (Papernot et al., 2017; Chen et al., 2017; Ilyas et al., 2018a;b; Xiao et al., 2018; Tashiro et al., 2020; Andriushchenko et al., 2020). Recently, stronger attacks (Croce & Hein, 2020a;b) are further proposed but none of them can guarantee that the attack is strong enough for detecting all adversarial examples. In this paper, we are the first to search adversarial examples in activation space instead of input space enabled by mature techniques of branch and bound used for verification, aiming to end the arms race in adversarial attacks.

Neural network verification. Early neural network verifiers solve the verification problem with linear solvers as a whole and can only scale to very small networks (Katz et al., 2017; Huang et al., 2017; Ehlers, 2017; Dutta et al., 2018; Tjeng et al., 2019a). Efficient bound propagation methods with various relaxations are then proposed for larger network verification but without completeness guarantee (Wong & Kolter, 2018; Raghunathan et al., 2018a;b; Singh et al., 2018b;a; Zhang et al., 2018; Weng et al., 2018). Recent BaB based verifiers relying on bound propagations can better exploit the structure of neural networks with ReLU neuron splits and refinements to accelerate and achieve completeness (Bunel et al., 2018; Wang et al., 2018a; Lu & Kumar, 2020; Botoeva et al., 2020). BaB based methods with input domain split and refinements are also investigated but their performance is greatly limited by input dimensions (Wang et al., 2018b; Royo et al., 2019; Anderson et al., 2019). Recent verifiers propose to use more efficient solvers to accelerate verification on GPUs and achieve several orders of magnitude speedup (Bunel et al., 2020; De Palma et al., 2021; Anderson et al., 2020; Tjandraatmadja et al., 2020; Xu et al., 2021; Müller et al., 2021; 2020; Wang et al., 2021). We customize our BaB attack based on α, β-CROWN (Xu et al., 2021; Wang et al., 2021) which is one of the most efficient GPU-based complete verifiers that won the 2nd neural network verification competition (VNN-COMP 2021) (Bak et al., 2021).

6 Conclusion

In this paper, we proposed BaB attack, a strong adversarial attack using branch and bound to systematically search adversarial examples in activation space. We propose a top-down beam search utilizing bound propagation methods from neural network verifiers on GPUs to quickly locate subdomains containing adversarial examples. We further exploit the common activation patterns of adversarial candidates found during beam search to guide the searching process with diving techniques. Moreover, a bottom-up large neighborhood search will look locally around promising adversarial candidates in parallel to refine the findings. Experimental results show that our BaB attack can find adversarial examples for hard input instances where SOTA adversarial attacks fail. We achieve higher attack success rates within much less time compared to generic MIP solver based attacks.

Limitations of this study. One limitation of BaB Attack is speed - it usually takes a few minutes to explore a branch and bound tree using beam search, while gradient based adversarial attacks are often very fast, in a few seconds. Practically, BaB attack is mostly useful for hard instances, and we can use other simpler attacks as a filter to reduce the number of inputs for BaB attack. Additionally, BaB attack currently relies on neural network verifiers to give a lower bound for each subdomain to guide the beam search, so its scalability is limited by SOTA verification tools. Yet, we have demonstrated that it is faster and also more effective than attacks using off-the-shelf MIP solver.

REFERENCES

Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Mani B Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1111–1119, 2019.

Greg Anderson, Shankara Pailoor, Isil Dillig, and Swarat Chaudhuri. Optimization and abstraction: A synergistic approach for analyzing neural network robustness. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2019.

Ross Anderson, Joey Huchette, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong convex relaxations and mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 2020.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020.

Anish Athalye, Nicolas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Stanley Bak, Changliu Liu, and Taylor Johnson. The second international verification of neural networks competition (vnn-comp 2021): Summary and results. *arXiv preprint arXiv:2109.00498*, 2021.

Timo Berthold. Primal heuristics for mixed integer programs. 2006.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 2013.

Elena Botoeva, Panagiotis Kouvaros, Jan Kronqvist, Alessio Lomuscio, and Ruth Misener. Efficient verification of relu-based neural networks via dependency analysis. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018.

Rudy Bunel, Alessandro De Palma, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip H. S. Torr, and M. Pawan Kumar. Lagrangian decomposition for neural network verification. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020.

Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 39–57. IEEE, 2017.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26. ACM, 2017.

Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning (ICML)*, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020b.

Francesco Croce, Jonas Rauber, and Matthias Hein. Scaling up the randomized gradient-free adversarial attack reveals overestimation of robustness using established attacks. *International Journal of Computer Vision*, 2020.

Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, et al. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Alessandro De Palma, Rudy Bunel, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip HS Torr, and M Pawan Kumar. Improved branch and bound for neural network verification via lagrangian decomposition. *arXiv preprint arXiv:2104.06718*, 2021.

Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, 2018.

Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019a.

Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. An alternative surrogate loss for pgd-based adversarial testing. *arXiv preprint arXiv:1910.09338*, 2019b.

Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification (CAV)*, 2017.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning (ICML)*, 2018a.

Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018b.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification (CAV)*, 2017.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Jingyue Lu and M Pawan Kumar. Neural network branching for neural network verification. *International Conference on Learning Representation (ICLR)*, 2020.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018.

Christoph Müller, Gagandeep Singh, Markus Püschel, and Martin Vechev. Neural network robustness verification on gpus. *arXiv preprint arXiv:2007.10868*, 2020.

Mark Niklas Müller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel, and Martin Vechev. Precise multi-neuron abstractions for neural network certification. *arXiv preprint arXiv:2103.03638*, 2021.

Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. ACM, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *International Conference on Learning Representations (ICLR)*, 2018a.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018b.

Vicenc Rubies Royo, Roberto Calandra, Dusan M Stipanovic, and Claire Tomlin. Fast neural network verification via shadow prices. *arXiv preprint arXiv:1902.07247*, 2019.

Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018a.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. Boosting robustness certification of neural networks. In *International Conference on Learning Representations*, 2018b.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2013.

Yusuke Tashiro, Yang Song, and Stefano Ermon. Diversity can be transferred: Output diversification for white-and black-box attacks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Christian Tjandraatmadja, Ross Anderson, Joey Huchette, Will Ma, Krunal Patel, and Juan Pablo Vielma. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *International Conference on Learning Representations (ICLR)*, 2019a.

Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL `https://openreview.net/forum?id=HyGIdiRqtm`.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Joachim P Walser. *Integer optimization by local search: A domain-independent approach*. Springer, 2003.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018a.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *USENIX Security Symposium*, 2018b.

Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Enhancing gradient-based attacks with symbolic intervals. *arXiv preprint arXiv:1906.02282*, 2019.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*, 2021.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *ICLR*, 2018.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, 2018.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Kai Y Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. In *ICLR*, 2019.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *International Conference on Learning Representations (ICLR)*, 2021.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Tianhang Zheng, Changyou Chen, and Kui Ren. Distributionally adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Rong Zhou and Eric A Hansen. Beam-stack search: Integrating backtracking with beam search. In *ICAPS*, pp. 90–98, 2005.