Improving adversarial robustness via joint classification and multiple explicit detection classes

Sina Baharlouei [1 2] **Fatemeh Sheikholeslami** [2] **Meisam Razaviyayn** [1] **Zico Kolter** [2 3]

Abstract

This work concerns the development of deep networks that are certifiably robust to adversarial attacks. Joint robust classification-detection was recently introduced as a certified defense mechanism, where adversarial examples are either correctly classified or assigned to the "abstain" class. In this work, we show that such provable framework can be extended to networks with multiple explicit abstain classes, where the adversarial examples are adaptively assigned to those. While naïvely adding multiple abstain classes can lead to "model degeneracy", we propose a regularization approach and a training method to counter this degeneracy by promoting full use of the multiple abstain classes. Our experiments demonstrate that the proposed approach consistently achieves favorable standard vs. robust verified accuracy tradeoff, outperforming state-of-the-art algorithms for various choices of number of abstain classes.

1. Introduction

Deep Neural Networks (DNNs) have revolutionized many machine learning tasks such as image processing (Krizhevsky et al., 2012; Zhu et al., 2021) and speech recognition (Graves et al., 2013; Nassif et al., 2019). However, despite their superior performance, DNNs are highly vulnerable to adversarial attacks and perform poorly on out-of-distributions samples (Goodfellow et al., 2014; Liang et al., 2017; Yuan et al., 2019). To address the vulnerability of DNNs to adversarial attacks, the community have designed various defense mechanisms that are robust against adversarial attacks (Papernot et al., 2016; Jang et al., 2019; Goldblum et al., 2020; Madry et al., 2017; Huang et al., 2021). These mechanisms provide robustness against certain types of attacks such as the Fast Gradient Sign Method (FGSM) (Szegedy et al., 2013; Goodfellow et al., 2014). However, the overwhelming majority of these defense mechanisms are highly ineffective against more complex attacks such as adaptive and brute-force methods (Tramer et al., 2020; Carlini & Wagner, 2017). This ineffectiveness necessitates: 1) the design of rigorous verification approaches that can measure the robustness of a given network; 2) the development of defense mechanisms that are verifiably robust against *any* attack strategy within the class of permissible attack strategies.

To verify robustness of a given network against *any* attack in a reasonable set of permissible attacks (e.g. ℓ_p-norm ball around the given input data), one needs to solve a hard non-convex optimization problem (see, e.g., Problem (1) in this paper). Consequently, exact verifiers, such as (Tjeng et al., 2017; Xiao et al., 2018), are not scalable to large networks. To develop scalable verifiers, the community turn to "inexact" verifiers. Such methods can only verify a subset of perturbations to the input data that the network can defend against successfully. This is typically achieved by finding tractable lower-bounds for the verification optimization problem. Gowal et al. (2018) finds such a lower-bound by *interval bound propagation (IBP)* which is essentially an efficient convex relaxation of the constraint sets in the verification problem. Despite its simplicity, this approach demonstrates a relatively superior performance compared to prior works. IBP-CROWN (Zhang et al., 2019) combines IBP with a novel linear relaxations to have a tighter approximation compared to standalone IBP. β-Crown (Wang et al., 2021) utilizes a branch-and-bound technique combined with the linear bounds proposed by IBP-CROWN to further tighten the relaxation gap. While β-Crown demonstrates a tremendous performance gain over other verifiers such as Zhang et al. (2019); Fazlyab et al. (2019); Lu & Kumar (2019), it cannot be used as a tool in large-scale training procedures due to its computationally expensive branch-and-bound search.

Another line of work for enhancing the performance of certifiably robust neural networks relies on the idea of learning a detector alongside the classifier to capture adversar-

[1]Industrial and Systems Engineering Department, University of Southern California, Los Angeles, CA. [2]Bosch Center for Artificial Intelligence, Pittsburgh, PA. [3]Carnegie Mellon University, Pittsburgh, PA..

1st Workshop on Formal Verification of Machine Learning, Baltimore, Maryland, USA. Colocated with ICML 2022.

ial and out-of-distribution samples. Instead of trying to classify adversarial images correctly, these works design a *detector* to determine whether a given sample is natural/in-distribution or it is a crafted attack/out-of-distribution. Chen et al. (2020) train the detector on both in-distribution and out-of-distribution samples to learn a detector distinguishing these samples. Hendrycks & Gimpel (2016) develops a method based on a simple observation that, for real samples, the output of softmax layer is closer to either 0 or 1 compared to out-of-distribution and adversarial examples where the softmax output entries are distributed more uniformly. DeVries & Taylor (2018); Sheikholeslami et al. (2020) learn uncertainty regions around actual samples where the prediction of the network remains the same. Interestingly, this approach does not require out-of-distribution samples during training. Other approaches such as deep generative models (Ren et al., 2019), self-supervised and ensemble methods (Vyas et al., 2018; Chen et al., 2021) are also used to learn out-of-distribution samples. However, typically these methods are vulnerable to adversarial attacks and can be easily fooled by carefully designed out-of-distribution images (Fort, 2022). A more resilient approach is to jointly learn the detector and the classifier (Laidlaw & Feizi, 2019; Sheikholeslami et al., 2021) by adding an auxiliary *abstain* output class capturing adversarial samples.

Building on these prior works, this paper extends the idea of using a single abstain class to using multiple abstain classes. We observe that naïvely adding multiple abstain classes results in a model degeneracy phenomenon where all adversarial examples are assigned to a small fraction of abstain classes (while other abstain classes are not utilized). To resolve this issue, we propose a regularizer that balances the assignment of adversarial examples to abstain classes. Our experiments demonstrate that utilizing multiple abstain classes in conjunction with the proper regularization enhances the robust verified accuracy of joint detectors/classifiers on adversarial examples while maintaining the standard accuracy of the classifier.

Contributions. We propose a framework for training and verification of robust neural nets with multiple detection classes. We generalize the IBP training and verification procedure, and β-Crown verifier to networks consisting of a classifier and multiple detection classes jointly. We identified a "model degeneracy" phenomenon where not all detection classes are utilized. To prevent model degeneracy and to avoid tuning the number of network detectors, we introduce a regularization approach guaranteeing that all detectors contributing to the detection of adversarial examples. Our experiments show that, compared to the networks with a single detection class, we enhance the robust verified accuracy by more than 5% and 2% on CIFAR-10 and MNIST datasets respectively for various perturbation sizes.

2. Background

2.1. Verification of feedforward neural networks

Consider an L-layer feedforward neural network with $\mathbf{W}_i$ denoting the weight associated with layer i, and $\mathbf{b}_i$ denoting the bias parameter of layer i. Let $\sigma_i(\cdot)$ denote the activation function applied at layer i. Throughout the paper, we assume the activation function is the same for all hidden layers, i.e., $\sigma_i(\cdot) = \text{ReLU}(\cdot),\ \forall i = 1, \dots, L-1$. Thus, our neural network can be described as

$$\begin{aligned} \mathbf{z}_i &= \sigma(\mathbf{W}_i \mathbf{z}_{i-1} + \mathbf{b}_i), \quad i = 1, 2, \dots, L-1, \\ \mathbf{z}_L &= \mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L \end{aligned}$$

where $\mathbf{z}_0 = \mathbf{x}$ is the input to the neural network and $\mathbf{z}_i$ is the output of layer i. Note that the activation function is not applied at the last layer. We consider a supervised classification task where $\mathbf{z}_L$ represents the logits. To explicitly show the dependence of $\mathbf{z}_L$ on the input data, we use the notation $\mathbf{z}_L(\mathbf{x})$ to denote logit values when $\mathbf{x}$ is used as the input data point.

Given an input $\mathbf{x}_0$ with the ground-truth label y, and a perturbation set $\mathcal{C}(\mathbf{x}_0, \epsilon)$ (e.g. $\mathcal{C}(\mathbf{x}_0, \epsilon) = \{\mathbf{x} \,|\, \|\mathbf{x} - \mathbf{x}_0\|_\infty \leq \epsilon\}$), the network is provably robust against adversarial attacks on $\mathbf{x}_0$ if

$$0 \leq \min_{\mathbf{x} \in \mathcal{C}(\mathbf{x}_0, \epsilon)} \mathbf{c}_{yk}^T \mathbf{z}_L(\mathbf{x}), \quad \forall k \neq y, \tag{1}$$

where $\mathbf{c}_{yk} = \mathbf{e}_y - \mathbf{e}_k$ with $\mathbf{e}_k$ (resp. $\mathbf{e}_y$) being the standard unit vector whose k-th row (resp. y-th row) is 1 and the other entries are zero. Condition (1) implies that the logit score of the network for the true label y is always greater than that of any other label k for all $\mathbf{x} \in \mathcal{C}(\mathbf{x}_0, \epsilon)$. Thus, the network will classify all the points inside $\mathcal{C}(\mathbf{x}_0, \epsilon)$ correctly. The objective function in Equation (1) is non-convex when $L \geq 2$. It is customary in many works to move the non-convexity of the problem to the constraint set and reformulate Equation (1) as

$$0 \leq \min_{\mathbf{z} \in \mathcal{Z}(\mathbf{x}_0, \epsilon)} \mathbf{c}_{yk}^T \mathbf{z}, \qquad \forall k \neq y, \tag{2}$$

where $\mathcal{Z}(\mathbf{x}_0, \epsilon) = \{\mathbf{z} \,|\, \mathbf{z} = \mathbf{z}_L(\mathbf{x}) \text{ for some } \mathbf{x} \in \mathcal{C}(\mathbf{x}_0, \epsilon)\}$. This verification problem has a linear objective function and a non-convex constraint set. Since both problems (1) and (2) are non-convex, existing works proposed efficiently computable lower-bounds for the optimal objective value of them. For example, Gowal et al. (2018); Wong & Kolter (2018) utilize convex relaxation, while Tjeng et al. (2017); Wang et al. (2021) rely on mixed integer programming and branch-and-bound to find lower-bounds for the optimal objective value of (2). In what follows, we explain two popular and relatively successful approaches for solving the verification problem (1) (or equivalently (2)) in detail.

2.2. Verification of neural networks via interval bound propagation (IBP)

Interval Bound Propagation (IBP) of Gowal et al. (2018) tackles problem (2) by convexification of the constraint set $\mathcal{Z}(\mathbf{x}_0, \epsilon)$ to its convex hypercube super-set $[\underline{\mathbf{z}}(\mathbf{x}_0), \bar{\mathbf{z}}(\mathbf{x}_0)]$, i.e., $\mathcal{Z}(\mathbf{x}_0, \epsilon) \subseteq [\underline{\mathbf{z}}(\mathbf{x}_0), \bar{\mathbf{z}}(\mathbf{x}_0)]$. After this relaxation, problem (2) can be lower-bounded by the convex problem:

$$\min_{\underline{\mathbf{z}}(\mathbf{x}_0) \le \mathbf{z} \le \bar{\mathbf{z}}(\mathbf{x}_0)} \mathbf{c}_{yk}^T \mathbf{z} \tag{3}$$

The upper- and lower- bounds $\underline{\mathbf{z}}(\mathbf{x}_0)$ and $\bar{\mathbf{z}}(\mathbf{x}_0)$ are obtained by recursively finding the convex relaxation of the image of the set $\mathcal{C}(\mathbf{x}_0, \epsilon)$ at each layer of the network. In particular, for the adversarial set $\mathcal{C}(\mathbf{x}_0, \epsilon) = \{\mathbf{x} \,|\, \|\mathbf{x} - \mathbf{x}_0\|_\infty \le \epsilon\}$, we start from $\underline{\mathbf{z}}_0(\mathbf{x}_0) = \mathbf{x}_0 - \epsilon\mathbf{1}$ and $\bar{\mathbf{z}}_0(\mathbf{x}_0) = \mathbf{x}_0 + \epsilon\mathbf{1}$. Then, the lower-bound $\underline{\mathbf{z}}_L(\mathbf{x}_0)$ and upper-bound $\bar{\mathbf{z}}_L(\mathbf{x}_0)$ are computed by the recursions:

$$\begin{aligned}
\bar{\mathbf{z}}_i(\mathbf{x}_0) &= \sigma(\mathbf{W}_i^T \frac{\bar{\mathbf{z}}_{i-1} + \underline{\mathbf{z}}_{i-1}}{2} + |\mathbf{W}_i^T| \frac{\bar{\mathbf{z}}_{i-1} - \underline{\mathbf{z}}_{i-1}}{2}), \\
\underline{\mathbf{z}}_i(\mathbf{x}_0) &= \sigma(\mathbf{W}_i^T \frac{\bar{\mathbf{z}}_{i-1} + \underline{\mathbf{z}}_{i-1}}{2} - |\mathbf{W}_i^T| \frac{\bar{\mathbf{z}}_{i-1} - \underline{\mathbf{z}}_{i-1}}{2}), \\
&\forall i = 1, \dots, L.
\end{aligned} \tag{4}$$

One of the main advantages of IBP is its efficient computation: verification of a given input only requires two forward passes for finding the lower and upper bounds, followed by a linear programming.

2.3. Verification of neural networks via β-Crown

Despite its simplicity, IBP-based verification comes with a certain limitation, namely the looseness of its layer-by-layer bounds of the input. To overcome this limitation, tighter verification methods have been proposed in the literature (Singh et al., 2018; Zhang et al., 2019; Dathathri et al., 2020; Wang et al., 2021). Among these, β-crown (Wang et al., 2021) utilizes the branch-and-bound technique to generalize and improve the IBP-CROWN proposed in Zhang et al. (2019). Let $\underline{\mathbf{z}}_i$ and $\bar{\mathbf{z}}_i$ be the estimated element-wise lower-bound and upper-bounds for the pre-activation value of $\mathbf{z}_i$, i.e., $\underline{\mathbf{z}}_i \le \mathbf{z}_i \le \bar{\mathbf{z}}_i$, where these lower and upper bounds are obtained by the method in Zhang et al. (2019). Let $\hat{\mathbf{z}}_i$ be the value we obtain by applying ReLU function to $\mathbf{z}_i$. We say a neuron is unstable if its sign after applying ReLU activation cannot be determined based on only knowing the corresponding lower and upper bounds. That is, a neuron is unstable if $\underline{\mathbf{z}}_i < 0 < \bar{\mathbf{z}}_i$. For **stable** neurons, no relaxation is needed to enforce convexity of $\sigma(\mathbf{z})$ (since the neuron operates in a linear regime). On the other hand, given an unstable neuron, they use branch-and-bound (BAB) approach to split the input range of the neuron into two sub-domains $\mathcal{C}_{il} = \{\mathbf{x} \in \mathcal{C}(\mathbf{x}_0, \epsilon) | \, \hat{z}_i \le 0\}$ and $\mathcal{C}_{iu} = \{\mathbf{x} \in \mathcal{C}(\mathbf{x}_0, \epsilon) | \, \hat{z}_i > 0\}$. Within each subdomain, the neuron operates linearly and hence verification is easy. Thus we can verify for each of these subdomains separately. If we have N unstable nodes, BAB algorithm requires the investigation of 2^N sub-domains in the worst-case. β-Crown proposes a heuristic for traversing all these subdomains: The higher the absolute value of the corresponding lower-bound of a node is, the sooner it is visited by the verifier. For verifying each sub-problem, Wang et al. (2021) proposed a lower-bounded which requires solving a maximization problem over two parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$\begin{aligned}
&\min_{\mathbf{z} \in \mathcal{Z}(\mathbf{x}_0, \epsilon)} \mathbf{c}_{yk}^T \mathbf{z} \ge \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}} g(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&\text{where} \quad g(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = (\mathbf{a} + \mathbf{P}_{\boldsymbol{\alpha}} \boldsymbol{\beta})^T \mathbf{x} + \mathbf{q}_{\boldsymbol{\alpha}}^T \boldsymbol{\beta} + \mathbf{d}_{\boldsymbol{\alpha}}.
\end{aligned} \tag{5}$$

Here, the matrix $\mathbf{P}$ and the vectors $\mathbf{q}, \mathbf{a}$ and $\mathbf{d}$ are functions of $\mathbf{W}_i, \mathbf{b}_i, \underline{\mathbf{z}}_i, \bar{\mathbf{z}}_i, \boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ parameters. See Appendix D for the precise definition of g. Notice that any choice of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ provides a valid lower bound for verification. However, optimizing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in (5) leads to a tighter bound.

2.4. Robust classification of adversarial examples with detection

Sheikholeslami et al. (2021) improves the performance tradeoff on natural and adversarial examples by introducing an auxiliary class for detecting adversarial examples. If this auxiliary class is selected as the output, the networks "abstains" from declaring any of the original K classes for the given input. Let a be the abstain class. The network performs correctly on an adversarial image if it is classified correctly (similar to robust networks without detectors) or it is classified as the abstain class (detected as an adversarial example). Hence, the network is verified against a certain class k if

$$0 \le \min_{\mathbf{z} \in \mathcal{Z}(\mathbf{x}_0, \epsilon)} \max(\mathbf{c}_{yk}^T \mathbf{z}, \mathbf{c}_{ak}^T \mathbf{z}), \tag{6}$$

i.e., if the score of the true label y or the score of the abstain class a is larger than the score of class k.

2.5. Training a joint robust classifier and detector

To train a neural network that can jointly detect and classify a dataset of images, Sheikholeslami et al. (2021) relies on the loss function of the form:

$$L_{\text{Total}} = L_{\text{Robust}} + \lambda_1 L_{\text{Robust}}^{\text{Abstain}} + \lambda_2 L_{\text{Standard}}, \tag{7}$$

where the term L_{Standard} denotes the standard loss when no adversarial examples are considered. More precisely, $L_{\text{Standard}} = \frac{1}{n} \sum_{i=1}^n \ell_{\text{xent}}\big(\mathbf{z}_L(\mathbf{x}_i), y_i\big)$, where ℓ_{xent} is the standard cross-entropy loss. The term L_{Robust} in (7) represents the worst-case adversarial loss used in (Madry et al.,

2017), without considering the abstain class. Precisely,

$$L_{\text{Robust}} = \max_{\boldsymbol{\delta}_1,\ldots,\boldsymbol{\delta}_n} \frac{1}{n}\sum_{i=1}^{n} \ell_{\text{xent}}\big(\mathbf{z}_L(\mathbf{x}_i+\boldsymbol{\delta}_i), y_i\big) \quad \text{s.t. } \|\boldsymbol{\delta}_i\|_\infty \le \epsilon, \ \ \forall i = 1,\ldots,n.$$

Finally, the Robust-Abstain loss $L^{\text{Abstain}}_{\text{Robust}}$ is the minimum of the detector and the classifier losses:

$$L^{\text{Abstain}}_{\text{Robust}} = \max_{\boldsymbol{\delta}_1,\ldots,\boldsymbol{\delta}_n} \frac{1}{n}\sum_{i=1}^{n} \min\Big(\ell_{\text{xent}}\big(\mathbf{z}_L(\mathbf{x}_i+\boldsymbol{\delta}_i), y_i\big), \ell_{\text{xent}}\big(\mathbf{z}_L(\mathbf{x}_i+\boldsymbol{\delta}_i), a\big)\Big) \quad \text{s.t. } \|\boldsymbol{\delta}_i\|_\infty \le \epsilon, \ \forall i \tag{8}$$

In (7), tuning λ_1 and λ_2 controls the trade-off between standard and robust accuracy. Furthermore, to obtain non-trivial results, IBP-relaxation should be incorporated during training for the minimization sub-problems in L_{robust} and $L^{\text{abstain}}_{\text{robust}}$ (Sheikholeslami et al., 2021; Gowal et al., 2018).

3. Verification of neural networks with multiple abstain classes

The robust verified accuracy of a joint classifier and detector can be enhanced by introducing multiple abstain classes instead of a single abstain class for detecting adversarial examples. This is simply because adding more classes would increase the capacity of the network in detecting adversarial examples. This observation is illustrated in a simple example in Appendix F. Note that a network with multiple abstain classes can be equivalently modeled by another network with one more layer and a single abstain class. This added layer can merge all abstain classes and reduce them to a single class. *Thus, any L-layer neural network with multiple abstain classes can be equivalently modeled by an $L+1$-layer neural network with a single abstain class.* However, the performance of verifiers such as IBP reduces as we increase the number of layers. Thus, it is beneficial to train/verify the original L-layer neural network with multiple abstain classes instead of $L+1$-layer network with a single abstain class. This fact will be illustrated further in our experiments. Next, we present how one can verify a network with multiple abstain classes.

Let $a_1, a_2, \ldots, a_M$ be M abstain classes detecting adversarial samples. A sample is considered adversarial if the output of the network is any of the M abstain classes. A neural network with K regular classes and M abstain classes outputs the label of a given sample as $\hat{y}(\mathbf{x}) = \text{argmax}_{i\in\{1,\ldots,k,a_1,\ldots,a_M\}}[\mathbf{z}_L(\mathbf{x})]_i$.

An input is verified if the network either correctly classifies it or assigns it to any of the explicit M abstain classes. More formally and following equation (6), the neural network is verified for input $\mathbf{x}_0$ against a target class k for a given image $(\mathbf{x}, y)$ if

$$0 \le \min_{\mathbf{z}\in\mathcal{Z}(\mathbf{x}_0,\epsilon)} \max\left\{\mathbf{c}_{yk}^T\mathbf{z}_L, \mathbf{c}_{a_1k}^T\mathbf{z}_L, \ldots, \mathbf{c}_{a_Mk}^T\mathbf{z}_L\right\}, \tag{9}$$

Since the set $\mathcal{Z}(\mathbf{x}_0,\epsilon)$ is highly nonconvex, verifying (9) is computationally expensive. In the next two subsections, we present sufficient conditions for (9) based on IBP and β-crown approaches.

3.1. Verification with IBP

Following the IBP approach to relax the nonconvex set $\mathcal{Z}(\mathbf{x}_0,\epsilon)$ leads to the following result:

Theorem 3.1. *Condition (9) is satisfied if*

$$\min_{\boldsymbol{\eta}\in\mathcal{P}} \max_{\underline{\mathbf{z}}_{L-1}\le\mathbf{z}_{L-1}\le\bar{\mathbf{z}}_{L-1}} -c_k(\boldsymbol{\eta})^T(\mathbf{W}_L\mathbf{z}_{L-1}+\mathbf{b}_L), \tag{10}$$

$\forall k \neq y$, is greater than 0. where $\mathcal{P} = \{(\eta_0,\ldots,\eta_M) | \sum_{i=0}^M \eta_i = 1, \eta_i \ge 0, \forall i = 0,1,\ldots,M\}$, and $c_k(\boldsymbol{\eta}) = \eta_0\mathbf{c}_{yk} + \eta_1\mathbf{c}_{a_1k}\cdots+\eta_M\mathbf{c}_{a_Mk}$. Here, the bounds $\underline{\mathbf{z}}_{L-1}$ and $\bar{\mathbf{z}}_{L-1}$ are obtained according to (4).

Unlike (9), the condition in (10) is easy to verify computationally. To understand this, let us define

$$J_k(\boldsymbol{\eta}) = \max_{\underline{\mathbf{z}}\le\mathbf{z}_{L-1}\le\bar{\mathbf{z}}} -c_k(\boldsymbol{\eta})^T(W_L\mathbf{z}_{L-1}+\mathbf{b}_L). \tag{11}$$

Then, our aim in (10) is to minimize $J_k(\boldsymbol{\eta})$ over $\mathcal{P}$. First notice that the maximization problem (11) can be solved in closed form as described in Step 1 of Algorithm 1. Consequently, one can rely on Danskin's Theorem (Danskin, 2012) to compute the subgradient of the function $J_k(\cdot)$. Thus, to minimize $J_k(\cdot)$ in (10), we can rely on the Bregman proximal (sub)gradient method (see (Gutman & Pena, 2018) and the references therein). This algorithm is guaranteed to find $\epsilon-$ accurate solution to (10) in $T = O(1/\sqrt{\epsilon})$ iterations–see (Gutman & Pena, 2018, Corollary 2).

Algorithm 1 IBP verification of networks with multiple abstain classes

1: **Parameters**: Stepsize $\nu > 0$, number of iterations T.
2: Initialize $\eta_0 = 1$ and $\eta_1 = \ldots = \eta_M = 0$.
3: **for** $t = 0, 1, \ldots, T$ **do**
4: Set $[\mathbf{z}^{*t}_{L-1}]_j = \begin{cases} [\underline{\mathbf{z}}_{L-1}]_j & \text{if } [\mathbf{W}_L^T c(\boldsymbol{\eta})]_j \ge 0 \\ [\bar{\mathbf{z}}_{L-1}]_j & \text{otherwise.} \end{cases}$, for every j.
5: Set $\eta_m^{t+1} = \frac{\eta_m^t \exp(-2\nu(\mathbf{z}^{*t}_{L-1})^T\mathbf{W}_L^T\mathbf{c}_{a_mk})}{\sum_{j=0}^M \eta_j^t \exp(-2\nu(\mathbf{z}^{*t}_{L-1})^T\mathbf{W}_L^T\mathbf{c}_{a_jk})}$, $\forall m \in \{0,\ldots,M\}$, where a_0 is defined as y.
6: **end for**

3.2. Verification with β-Crown

While IBP verification is computationally efficient, it is less accurate than β–Crown, as discussed earlier. Hence, to obtain a more accurate verification, in this section we focus on β–Crown verification of networks with multiple abstain classes. To this end, we will find a sufficient condition for (9) using the lower-bound technique of (5) in β–Crown. In particular, by switching the minimization and maximization in (9) and using the β–Crown lower bound (5), we can find a lower-bound of the form

$$\min_{\mathbf{z}_L \in \mathcal{Z}(\mathbf{x}_0,\epsilon)} \max\{\mathbf{c}_{yk}^T \mathbf{z}_L, \mathbf{c}_{a_1 k}^T \mathbf{z}_L, \dots, \mathbf{c}_{a_M k}^T \mathbf{z}_L\} \geq \max_{\boldsymbol{\eta} \in \mathcal{P}, \boldsymbol{\alpha}, \boldsymbol{\beta} \geq 0} G(\mathbf{x}_0, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}). \tag{12}$$

The details of this inequality and the exact definition of function $G(\cdot)$ is provided in Appendix E. Note that any feasible solution to the right hand side of (12) is a valid lower-bound to the original verification problem (left-hand-side). Thus, in order for (9) to be satisfied, it suffices to find a feasible $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta})$ such that $G(\mathbf{x}_0, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}) \geq 0$. To optimize the RHS of (12) in Algorithm 2, we utilize AutoLirpa library of (Zhang et al., 2019) for updating $\boldsymbol{\alpha}$, and use Bregman proximal subgradient method to update $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ – See appendix B. We use Euclidean norm Bregman divergence for updating $\boldsymbol{\beta}$, and Shannon entropy Bregman divergence for $\boldsymbol{\eta}$ to obtain closed-form updates.

Algorithm 2 β–Crown verification of networks with multiple abstain classes

1: **Input**: number of iterations T, number of iterations in the inner-loop T_0, Step-size γ.
2: **for** $t = 0, 1, \dots, T$ **do**
3: Update α using AutoLirpa library (Zhang et al., 2019)
4: **for** $k = 0, 1, \dots, T_0$ **do**
5: $\boldsymbol{\beta} = [\boldsymbol{\beta} + \gamma \frac{\partial G(\mathbf{x}_0, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta})}{\partial \boldsymbol{\beta}}]_+$, where $[w]_+ = \max\{0, w\}$ is projection to non-negative orthant
6: $\eta_m^{\text{new}} = \frac{\eta_m^{\text{old}} \exp(2\gamma \frac{\partial G(\mathbf{x}_0, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta})}{\partial \eta_m})}{\sum_{j=0}^{M} \eta_j^{\text{old}} \exp(2\gamma \frac{\partial G(\mathbf{x}_0, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta})}{\partial \eta_j})}$, $\forall m \in \{0, \dots, M\}$
7: **end for**
8: **end for**

4. Training of neural networks with multiple abstain classes

To train a neural network consisting of multiple abstain classes, we follow a similar combination of loss functions as in (7). While the last term (L_{Standard}) can be computed efficiently, the first and second terms cannot be computed efficiently because even evaluating the functions L_{Robust} and $L_{\text{Robust}}^{\text{Abstain}}$ requires maximizing nonconcave functions. Thus, instead of minimizing these two terms, we will minimize their upper-bounds. Particularly, following (Sheikholeslami et al., 2020, Equation (17)), we use $\bar{L}_{\text{Robust}}$ as an upper-bound to L_{Robust}. This upper-bound is obtained by the IBP relaxation procedure described in subsection 2.2. To obtain an upper-bound for the Robust-Abstain loss term $L_{\text{Robust}}^{\text{Abstain}}$ in (7), let us first start by clarifying its definition in the multi-abstain class scenario:

$$L_{\text{Robust}}^{\text{Abstain}} = \max_{\delta_1, \dots, \delta_n} \frac{1}{n} \sum_{i=1}^{n} \min \Big\{ \ell_{\text{xent}}\big(\mathbf{z}_L(\mathbf{x}_i + \delta_i), y_i\big), \min_{m=1,\dots,M} \ell_{\text{xent}}\big(\mathbf{z}_L(\mathbf{x}_i + \delta_i), a_m\big) \Big\}. \tag{13}$$

This definition implies that the classification is considered "correct" for a given input if the predicted label is the ground-truth label or if it is assigned to one of the abstain classes. Since the maximization problem w.r.t. $\{\delta_i\}$ is nonconcave, it is hard to even evaluate $L_{\text{Robust}}^{\text{Abstain}}$. Thus, we minimize an efficiently computable upper-bound of this loss function as described in Theorem 4.1.

Theorem 4.1. *Let* $\ell_{\mathit{Robust}}^{\mathit{Abstain}}(\mathbf{x}, y) = \max_{\|\delta\| \leq \epsilon} \min \Big\{ \ell_{\mathit{xent}}\big(\mathbf{z}_L(\mathbf{x} + \delta), y\big), \min_{m=1,\dots,M} \ell_{\mathit{xent}}\big(\mathbf{z}_L(\mathbf{x}_i + \delta_i), a_m\big) \Big\}$. *Then,*

$$\ell_{\mathit{Robust}}^{\mathit{Abstain}}(\mathbf{x}, y) \leq \bar{\ell}_{\mathit{Robust}}^{\mathit{Abstain}}(\mathbf{x}, y) = \ell_{\mathit{xent}\setminus\mathcal{A}_0}(J(\mathbf{x}), y), \tag{14}$$

where $J(\mathbf{x})$ *is a vector whose* k*-th component equals* $J_k(\mathbf{x})$ *as defined in* (11) *and* $\ell_{\mathit{xent}\setminus\mathcal{A}_0}(\mathbf{x}_0, y) := -\log\left(\frac{\exp(\mathbf{e}_y^T \mathbf{z}_L(\mathbf{x}_0))}{\sum_{i \in \mathcal{I}\setminus\mathcal{A}_0} \exp(\mathbf{e}_i^T \mathbf{z}_L(\mathbf{x}_0))}\right)$. *Here,* $\mathcal{I} = \{1, \dots, K, a_1, \dots, a_M\}$ *is the set of all classes (true labels and abstain classes) and* $\mathcal{A}_0 = \{a_1, \dots, a_M\}$ *is the set of abstain classes.*

Notice that the definition of $\ell_{\text{xent}\setminus\mathcal{A}_0}(\mathbf{x}_0, y)$ removes the terms corresponding to the abstain classes in the denominator. This definition is less restrictive toward abstain classes compared to incorrect classes. Thus, for a given sample, it is more advantageous for the network to classify it as an abstain class instead of incorrect classification. This mechanism enhances the performance of the network on detecting adversarial examples by abstain classes, while it does not have an adverse effect on the performance of the network on natural samples.Note that during the evaluation/test phase, this loss function does not change the final prediction of the network for a given input, since the winner (the entry with the highest score) remains the same.

Overall, we upper-bound the loss in (7) by replacing L_{Robust} with the IBP relaxation approach utilized in Gowal et al.

(2018); Sheikholeslami et al. (2021) and replacing $L^{\text{Abstain}}_{\text{Robust}}$ with $\bar{L}^{\text{Abstain}}_{\text{Robust}} = \frac{1}{n}\sum_{i=1}^{n} \bar{\ell}^{\text{Abstain}}_{\text{Robust}}(\mathbf{x}_i, y_i)$ presented in Theorem 4.1. Thus our total training loss can be presented as:

$$L_{\text{Total}} = \bar{L}_{\text{Robust}} + \lambda_1 \bar{L}^{\text{Abstain}}_{\text{Robust}} + \lambda_2 L_{\text{Standard}} \tag{15}$$

Algorithm 3 describes the procedure of optimizing (15) on a joint classifier and detector with multiple abstain classes.

Algorithm 3 Train a robust neural network on a training data

Input: Batches of data $\mathcal{D}_1, \ldots, \mathcal{D}_R$, step-size ν.
for $t = 1, \ldots, R$ **do**
  Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathcal{D}_t$
  Compute $J_o(\mathbf{x}) \quad \forall\, \mathbf{x} \in \mathcal{D}_t, \quad \forall o \in \{1, \ldots, K\}$ by Algorithm 1.
  Compute L_{Robust} as described in Gowal et al. (2018) on Batch $\mathcal{D}_t$.
  Compute $\bar{L}^{\text{abstain}}_{\text{Robust}}$ on Batch $\mathcal{D}_t$ using Theorem 4.1.
  $L = L - \nu\nabla\big(\bar{L}_{\text{Robust}} + \lambda_1 \bar{L}^{\text{abstain}}_{\text{Robust}} + \lambda_2 L_{\text{Standard}}\big)$
end for

4.1. Addressing model degeneracy

Having multiple abstain classes can potentially increase the capacity of our classifier to detect adversarial examples. However, as we will see in Figure 2 (10 abstains, unregularized), several abstain classes collapse together and capture similar adversarial patterns. Such a phenomenon, which we referred to as "model degeneracy" and is illustrated with an example in Appendix F, will prevent us from utilizing all abstain classes fully. To address this issue, we impose a regularization term to the loss function such that the network utilizes all abstain classes in balance. We aim to make sure the $\boldsymbol{\eta}$ values are distributed nearly uniformly and there are no *idle* abstain classes. Let $\boldsymbol{\eta}^{ik}$, $\mathbf{z}_{L-1}(\mathbf{x}_i)$, and y_i be the abstain vector corresponding to the sample $\mathbf{x}_i$ verifying against the target class k, the output of the layer $L-1$, and the assigned label to the data point $\mathbf{x}_i$ respectively. Therefore, the regularized verification problem over n given samples takes the following form:

$$\min_{\boldsymbol{\eta}^1, \ldots, \boldsymbol{\eta}^n \in \mathcal{P}} \sum_{i=1}^{n} \sum_{k \neq y_i} \max_{\underline{\mathbf{z}}(\mathbf{x}_i) \leq \mathbf{z}_{L-1} \leq \bar{\mathbf{z}}(\mathbf{x}_i)} -c_k(\boldsymbol{\eta}^{ik})(\mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L) + \mu \| [\frac{\gamma \mathbf{1}}{M+1} - \frac{1}{n(K-1)} \sum_{j=1}^{n} \sum_{o \neq y_i} \boldsymbol{\eta}^{jo}]_+ \|^2, \tag{16}$$

The above regularizer penalizes the objective function if the average value of $\boldsymbol{\eta}$ coefficient corresponding to a given abstain class over all samples of the batch is smaller than a threshold (the threshold is determined by the hyperparameter γ). In other words, if an abstain class is not contributing enough to the detection of adversarial samples, it will be penalized accordingly. Note that if γ is larger, we penalize an *idle* abstain class more.

Note that in the unregularized case, the optimization of parameters $\boldsymbol{\eta}^{ik}$ are independent of each other. In contrast, by adding the regularizer described in (16) we require to optimize $\boldsymbol{\eta}^{ik}$ parameters of different samples and target classes jointly (they are coupled in the regularization term). Since optimizing (16) over the set of all n samples is infeasible for datasets with large number of samples, we solve the problem over smaller batches of the data to reduce the complexity of problem in each iteration. We utilize the same Bergman divergence procedure used in Algorithm 1, while the gradient with respect to $\boldsymbol{\eta}^{ik}$ takes the regularization term into account as well.

5. Numerical results

We devise a diverse set of experiments on shallow and deep networks to investigate the effectiveness of our proposed joint classifier and detector with multiple abstain classes. To train the neural networks on MNIST and CIFAR-10 datasets, we use Algorithm 3 as a part of an optimizer scheduler. In the first phase, we set $\lambda_1 = \lambda_2 = 0$. Thus, the network is trained without considering any abstain classes initially. In the second phase we optimize the objective function (15), where we linearly increase ϵ from 0 to ϵ_{train}. In the last phase, we further tune the network on the fixed $\epsilon = \epsilon_{\text{train}}$ (see Appendix A for further details).

In the first set of experiments depicted in Figure 1, we compare the performance of the shallow networks with the optimal number of abstain classes to the single abstain network, the network with an additional layer, and the network regularized to have balance between different abstain classes (Equation 16). The shallow networks have one convolutional layer with size 256 and 1024 for training on MNIST and CIFAR-10 datasets respectively. This convolutional layer is connected to the second (last) layer consisting of $K+M$ nodes where K is the number of regular classes (10 for both MNIST and CIFAR-10 datasets) and M is the number of abstain classes. The optimal number of abstain classes is obtained by changing the number of them from $M=1$ to $M=20$ on both CIFAR-10 and MNIST datasets. The optimal value for the network trained on MNIST is $M=3$ and $M=4$ for CIFAR-10 dataset. Moreover, we compare the optimal multi-abstain shallow network to two other baselines: One is the network with the number of abstain classes equal to the number of regular classes ($M=K$) and is trained via the regularizer described in (16). The other baseline is a network with one more layer compared to the shallow network. Instead of the last layer in the shallow network, this network has $K+M$ nodes in the layer one to the last, and $K+1$ nodes in the last layer (the same optimal numbers for M are used for these networks). Ideally the set

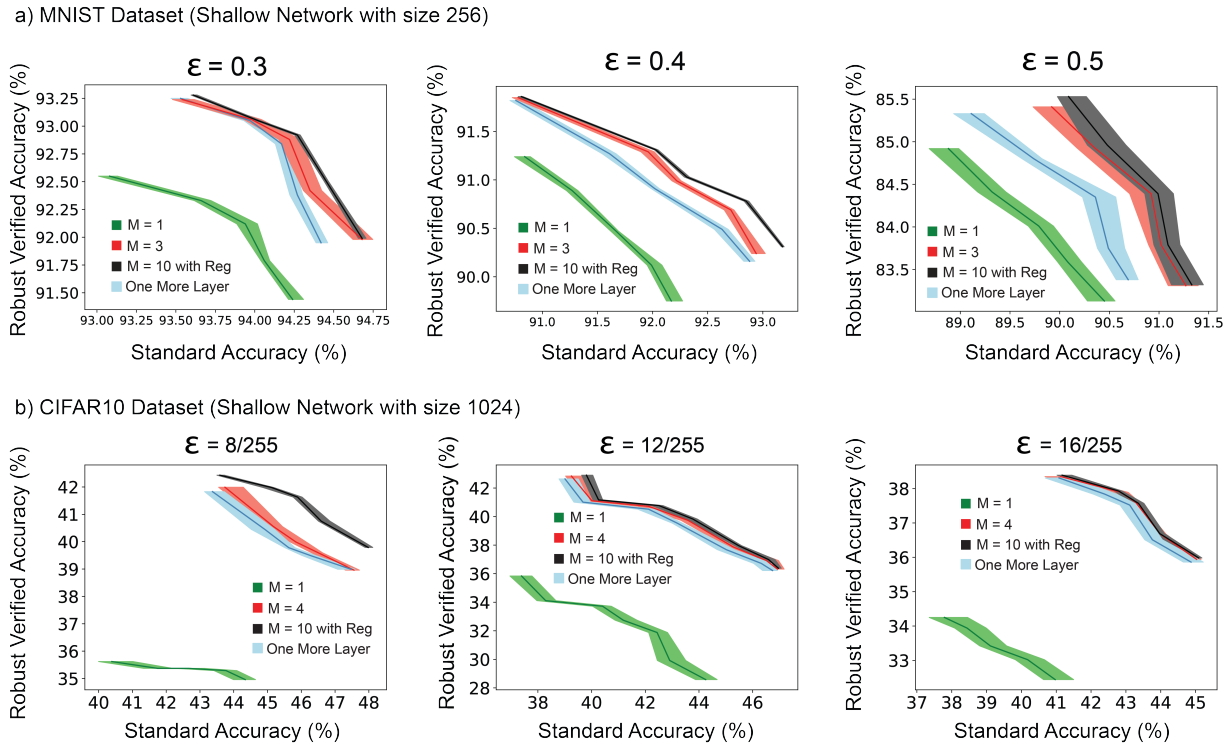

Figure 1. Performance of Multiple-abstain shallow networks on MNIST and CIFAR-10 datasets. We compared multiple abstain neural networks (both regularized and non-regularized version) with the single abstain networks and networks with one more layer. The above and below rows demonstrate the trade-off between standard and robust verified accuracy on MNIST and CIFAR-10 datasets.

of models can be supported by such a network is a super-set of the original shallow network. However, due to the training procedure (IBP) which is sensitive to higher number of layers (the higher the number of layers, the looser the lower and upper bounds), we obtain better results with the original network with multiple abstain classes.

Next, in Figure 3, we investigate the effect of changing the number of abstain classes of the shallow network described above. As we observe, the unregularized networks are much more sensitive to the change of M than the regularized version. This means, we can use the regularized network with the same performance while it does not require to be tuned for the optimal M. In the unregularized version, by increasing the number of abstain classes from $M = 1$ to $M = 5$ we see improvement. However, after this threshold, the network performance drops gradually such that for $M = 10$ where the number of labels and abstain classes are equal ($M = K = 10$) the performance of the network in this case is even worse than the single-abstain network due to the model degeneracy of the multi-abstain network. However, the network trained on the regularized loss maintains the performance when M changes from the optimal value to larger values. Moreover, the regularized networks have superior performance compared to the networks with one more layer.

Figure 2 demonstrates the performance of no abstain, single abstain, and multiple abstain networks with and without regularization on both natural and adversarial images. The above figure shows the distribution of natural images that are classified correctly, captured by the abstain classes, or misclassified. The below chart shows the percentage of adversarial examples classified correctly, or captured by each abstain class ($M = 10$). The results are obtained by training shallow neural networks on CIFAR-10 dataset. The hyper-parameter γ is set to $\frac{1}{K+M} = \frac{1}{20}$ in (16).

Beside the shallow networks, the trade-off between standard and verified robust accuracy is superior on the deep networks with multiple abstain classes compared to the networks with the same structure having single or no abstain classes (See Table 2 in the appendix). The structure of the trained deep network is exactly as the one described in Sheikholeslami et al. (2021).

6. Conclusion

We improved the trade-off between standard accuracy and robust verifiable accuracy by introducing multiple ab-

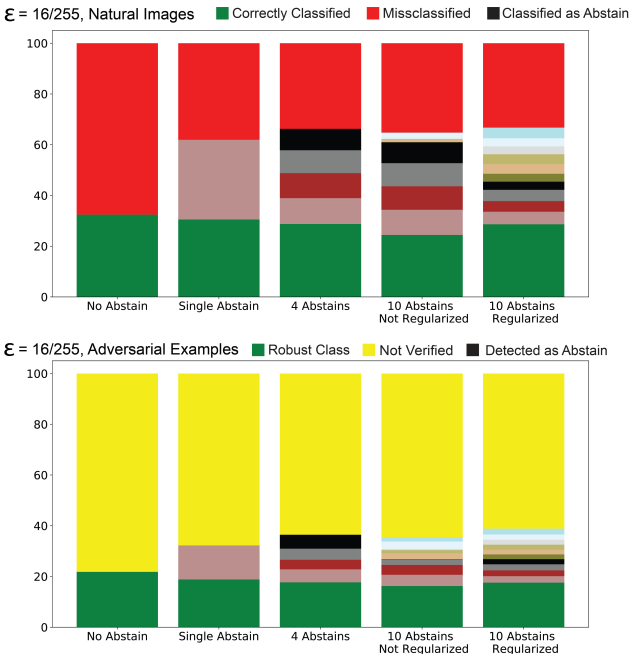

Figure 2. Distribution of natural and adversarial images over different abstain classes on CIFAR-10 dataset. When there are 10 abstain classes, model degeneracy leads to lower performance compared to the baseline. Adding the regularization term (right most column) will utilize all abstain classes and enhance both standard and robust verified accuracy. Standard accuracy is the proportion of correctly classified natural images, while robust verified accuracy is the proportion of images that are robust against all adversarial attacks within the ϵ-neighborhood.

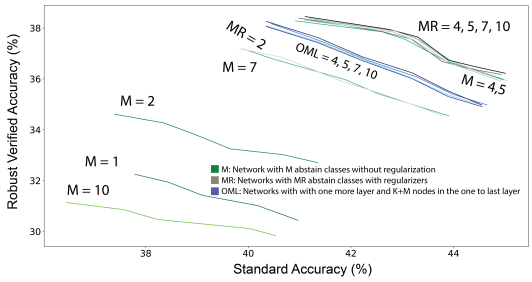

Figure 3. Performance of Multiple-abstain shallow network on CIFAR-10 datasets. Three different types of networks are evaluated: the networks with multiple abstain classes but without the regularization term (M), the networks with multiple abstain classes regularized on with Equation (16) (MR), and the networks with single abstain class but one more layer OML. The results demonstrate the less sensitivity of MR networks compared to M networks. Moreover, the performance of MR networks are better than the OML networks.

stain classes to the training and verification procedures of neural networks. We observed that adding multiple abstain class can result in a "model degeneracy" phenomenon where not all abstain classes are utilized. To avoid model degeneracy when the number of abstain classes is large, we propose a regularizer scheme forcing the network to utilize all abstain classes. Our experiments demonstrate the superiority of the trained shallow and deep networks over state-of-the-art approaches on MNIST and CIFAR-10 datasets. Our code is available at https://anonymous.4open.science/r/MultipleAbstainDetection-E85E.

References

Balunovic, M. and Vechev, M. Adversarial training and provable defenses: Bridging the gap. In International Conference on Learning Representations, 2019.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp), pp. 39–57. IEEE, 2017.

Chen, J., Li, Y., Wu, X., Liang, Y., and Jha, S. Robust out-of-distribution detection for neural networks. arXiv preprint arXiv:2003.09711, 2020.

Chen, J., Zhu, C., and Dai, B. Understanding the role of self-supervised learning in out-of-distribution detection task. arXiv preprint arXiv:2110.13435, 2021.

Danskin, J. M. The theory of max-min and its application to weapons allocation problems, volume 5. Springer Science & Business Media, 2012.

Dathathri, S., Dvijotham, K., Kurakin, A., Raghunathan, A., Uesato, J., Bunel, R., Shankar, S., Steinhardt, J., Goodfellow, I., Liang, P., et al. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. arXiv preprint arXiv:2010.11645, 2020.

DeVries, T. and Taylor, G. W. Learning confidence for out-of-distribution detection in neural networks. arXiv preprint arXiv:1802.04865, 2018.

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. J. Efficient and accurate estimation of lipschitz constants for deep neural networks. arXiv preprint arXiv:1906.04893, 2019.

Fort, S. Adversarial vulnerability of powerful near out-of-distribution detection. arXiv preprint arXiv:2201.07012, 2022.

Goldblum, M., Fowl, L., Feizi, S., and Goldstein, T. Adversarially robust distillation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 3996–4003, 2020.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. arXiv preprint arXiv:1810.12715, 2018.

Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing, pp. 6645–6649. Ieee, 2013.

Gutman, D. H. and Pena, J. F. A unified framework for bregman proximal methods: subgradient, gradient, and accelerated gradient schemes. arXiv preprint arXiv:1812.10198, 2018.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136, 2016.

Huang, T., Halbe, S., Sankar, C., Amini, P., Kottur, S., Geramifard, A., Razaviyayn, M., and Beirami, A. Dair: Data augmented invariant regularization. arXiv preprint arXiv:2110.11205, 2021.

Jang, Y., Zhao, T., Hong, S., and Lee, H. Adversarial defense via learning to generate diverse attacks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2740–2749, 2019.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 2012.

Laidlaw, C. and Feizi, S. Playing it safe: Adversarial robustness with an abstain option. arXiv preprint arXiv:1911.11253, 2019.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. arXiv preprint arXiv:1706.02690, 2017.

Lu, J. and Kumar, M. P. Neural network branching for neural network verification. arXiv preprint arXiv:1912.01329, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083, 2017.

Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., and Shaalan, K. Speech recognition using deep neural networks: A systematic review. IEEE access, 7:19143–19165, 2019.

Nouiehed, M. and Razaviyayn, M. Learning deep models: Critical points and local openness. INFORMS Journal on Optimization, 2021.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In 2016 IEEE symposium on security and privacy (SP), pp. 582–597. IEEE, 2016.

Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., DePristo, M. A., Dillon, J. V., and Lakshminarayanan, B. Likelihood ratios for out-of-distribution detection. arXiv preprint arXiv:1906.02845, 2019.

Sheikholeslami, F., Jain, S., and Giannakis, G. B. Minimum uncertainty based detection of adversaries in deep neural networks. In 2020 Information Theory and Applications Workshop (ITA), pp. 1–16. IEEE, 2020.

Sheikholeslami, F., Rezaabad, A. L., and Kolter, J. Z. Provably robust classification of adversarial examples with detection. International Conference on Learning Representations (ICLR), 2021.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. T. Fast and effective robustness certification. NeurIPS, 1(4):6, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

Tjeng, V., Xiao, K., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. arXiv preprint arXiv:1711.07356, 2017.

Tramer, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. arXiv preprint arXiv:2002.08347, 2020.

Vyas, A., Jammalamadaka, N., Zhu, X., Das, D., Kaul, B., and Willke, T. L. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 550–564, 2018.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. arXiv preprint arXiv:2103.06624, 2021.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In International Conference on Machine Learning, pp. 5286–5295. PMLR, 2018.

Xiao, K. Y., Tjeng, V., Shafiullah, N. M., and Madry, A. Training for faster adversarial robustness verification via inducing relu stability. arXiv preprint arXiv:1809.03008, 2018.

Yuan, X., He, P., Zhu, Q., and Li, X. Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems, 30 (9):2805–2824, 2019.

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. arXiv preprint arXiv:1906.06316, 2019.

Zhu, Z., Huang, G., Deng, J., Ye, Y., Huang, J., Chen, X., Zhu, J., Yang, T., Lu, J., Du, D., et al. Webface260m: A benchmark unveiling the power of million-scale deep face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10492–10502, 2021.

A. Implementation Details

In table A, we demonstrate the structure of the deep networks used in experiments of Table 2. The scheduler used for the experiments is the one utilized by Sheikholeslami et al. (2021). On both MNIST and CIFAR-10 datasets, we have used an Adam optimizer with learning rate $5x10e-4$. κ is scheduled by a linear ramp-down process, starting at 1, which after a warm-up period is ramped down to value $\kappa_{\text{end}} = 0.5$. Value of ϵ during the training is also simultaneously scheduled by a linear ramp-up, starting at 0 and ϵ_{Train} as the final value. The networks are trained with four NVIDIA V100 GPUs.

Network Layers
Conv 64 3×3
Conv 64 3×3
Conv 128 3×3
Conv 128 3×3
Fully Connected 512
Linear 10

Table 1. Standard and Robust Verified error of state-of-the-art approaches on CIFAR-10 dataset.

1. For MNIST, we train on a single Nvidia V100 GPU for 100 epochs with batch sizes of 100. The total number of training steps is 60K. We decay the learning rate by $10\times$ at steps 15K and 25K. We use warm-up and ramp-up duration of 2K and 10K steps, respectively. We do not use any data augmentation techniques and use full 28×28 images without any normalization.

2. CIFAR-10, we train for 3200 epochs with batch sizes of 1600. The total number of training steps is 100K. We decay the learning rate by 10× at steps 60K and 90K. We use warm-up and ramp-up duration of 5K and 50K steps, respectively. During training, we add random translations and flips, and normalize each image channel (using the channel statistics from the train set).

B. Bergman-Divergence Method for Optimizing a Convex Function Over a Probability Simplex

In this section, we show how to optimize a convex optimization problem over a probability simplex by using the Bergman divergence method. Let $\boldsymbol{\eta}$ be a vector of n elements. We aim to minimize the following constrained optimization problem where J is a convex function with respect to $\boldsymbol{\eta}$:

$$\min_{\eta_1,\ldots,\eta_n} J(\eta_1,\ldots,\eta_n) \quad \text{subject to} \quad \sum_{i=1}^{n} \eta_i = 1, \quad \eta_i \geq 0 \quad \forall i = 1,\ldots,n. \tag{17}$$

To solve the above problem, we define the Bergman distance function as:

$$B(\mathbf{x},\mathbf{y}) = \gamma(\mathbf{x}) - \gamma(\mathbf{y}) - \langle \nabla\gamma(\mathbf{x}), \mathbf{x}-\mathbf{y}\rangle$$

where γ is a strictly convex function. For this specific problem where the constrain is over a probability simplex, we choose $\gamma(\mathbf{x}) = \sum_{i=1}^{n} x_i \log(x_i)$. Thus:

$$B(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} x_i \log(\frac{x_i}{y_i})$$

One can rewrite problem 17 as:

$$\min_{\eta_1,\ldots,\eta_n} J(\eta_1,\ldots,\eta_n) + \mathcal{I}_{\mathbb{P}}(\boldsymbol{\eta}) \tag{18}$$

where $\mathbb{P} =$. Applying proximal gradient descent method on the above problem, we have:

$$\boldsymbol{\eta}^{r+1} = \underset{\boldsymbol{\eta}}{\operatorname{argmin}}\, \mathcal{I}_{\mathbb{P}}(\boldsymbol{\eta}) + \langle \nabla J(\boldsymbol{\eta}), \boldsymbol{\eta} - \boldsymbol{\eta}^i \rangle + \frac{1}{2\nu} B(\boldsymbol{\eta}, \boldsymbol{\eta}^i) \tag{19}$$

$$= \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \sum_{i=1}^{n} \frac{\partial J(\boldsymbol{\eta}^r)}{\partial \eta_i}(\eta_i - \eta_i^r) + \frac{1}{2\nu}\Big(\sum_{i=1}^{n} \eta_i \log(\eta_i) - \sum_{i=1}^{n} \frac{\partial \gamma(\boldsymbol{\eta}_i^r)}{\partial \eta_i}(\eta_i - \eta_i^r)\Big) \tag{20}$$

By simplifying the above problem, it turns to:

$$\boldsymbol{\eta}^{r+1} = \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \sum_{i=1}^{n} \eta_i \Big(\frac{\partial J(\boldsymbol{\eta}^r)}{\partial \eta_i} - \frac{1}{2\nu}\log(\eta_i^r) - \frac{1}{2\nu}\Big) + \frac{1}{2\nu}\sum_{i=1}^{n} \eta_i \log(\eta_i) \tag{21}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \eta_i = 1, \quad \eta_i \geq 0 \quad \forall i = 1, \ldots, n. \tag{22}$$

Writing the Lagrangian function of the above problem, we have:

$$\boldsymbol{\eta}^{r+1} = \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \sum_{i=1}^{n} \eta_i \Big(\frac{\partial J(\boldsymbol{\eta}^r)}{\partial \eta_i} - \frac{1}{2\nu}\log(\eta_i^r) - \frac{1}{2\nu}\Big) + \frac{1}{2\nu}\sum_{i=1}^{n} \eta_i \log(\eta_i) + \lambda^*\Big(\sum_{i=1}^{n} \eta_i - 1\Big) \tag{23}$$

$$\text{subject to} \quad \eta_i \geq 0 \quad \forall i = 1, \ldots, n.$$

By taking the derivative with respect to η_i and using the constraint $\sum_{i=1}^{n} \eta_i = 1$, it can be shown that:

$$\eta_i^{r+1} = \frac{\eta_i^r \exp(-2\nu \nabla J(\boldsymbol{\eta})_i)}{\sum_{j=1}^{n} \eta_j^r \exp(-2\nu \nabla J(\boldsymbol{\eta})_j)} \tag{24}$$

We use the update rule (24) in Algorithm 1 and Algorithm 2 to obtain the optimal $\boldsymbol{\eta}$ at each iteration.

C. Proof of Theorems

In this section, we prove Theorem 3.1 and Theorem 4.1.

Proof of Theorem 3.1: Starting from Equation 9, we can equivalently formulate it as:

$$\min_{\mathbf{z} \in \mathcal{Z}(\mathbf{x}_0, \epsilon)} \max(\mathbf{c}_{yk}^T \mathbf{z}, \mathbf{c}_{a_1 k}^T \mathbf{z}, \ldots, \mathbf{c}_{a_M k}^T \mathbf{z}) = \min_{\mathbf{z} \in \mathcal{Z}(\mathbf{x}_0, \epsilon)} \max_{\{\eta_0, \ldots, \eta_M\} \in \mathcal{P}} \quad c_k(\boldsymbol{\eta})^T \mathbf{z}. \tag{25}$$

Note that the maximum element of the left hand side can be obtained by setting its corresponding η coefficient to 1 on the right hand side. Conversely, any optimal solution to the right hand is exactly equal to the maximum element of the left hand side. According to the min-max equality (duality), when the minimum and the maximum problems are interchanged, the following inequality holds:

$$\begin{aligned} &\min_{\mathbf{z} \in \mathcal{Z}(\mathbf{x}_0, \epsilon)} \max_{\{\eta_0, \ldots, \eta_M\} \in \mathcal{P}} \quad \eta_0 \mathbf{c}_{yk}^T \mathbf{z} + \eta_1 \mathbf{c}_{a_1 k}^T \mathbf{z} + \cdots + \eta_M \mathbf{c}_{a_M k}^T \mathbf{z} \geq \\ &\max_{\{\eta_0, \ldots, \eta_M\} \in \mathcal{P}} \min_{\mathbf{z} \in \mathcal{Z}(\mathbf{x}_0, \epsilon)} \quad \eta_0 \mathbf{c}_{yk}^T \mathbf{z} + \eta_1 \mathbf{c}_{a_1 k}^T \mathbf{z} + \cdots + \eta_M \mathbf{c}_{a_M k}^T \mathbf{z}. \end{aligned} \tag{26}$$

Moreover, by the definition of upper-bounds and lower-bounds presented in (Gowal et al., 2018), $\mathcal{Z}(\mathbf{x}_0, \epsilon)$ is a subset of $\underline{\mathbf{z}}_L \leq \mathbf{z} \leq \bar{\mathbf{z}}_L$. Thus:

$$\max_{\{\eta_0,\ldots,\eta_M\}\in\mathcal{P}} \min_{\mathbf{z}\in\mathcal{Z}(\mathbf{x}_0,\epsilon)} \quad \eta_0\mathbf{c}_{yk}^T\mathbf{z} + \eta_1\mathbf{c}_{a_1k}^T\mathbf{z} + \cdots + \eta_M\mathbf{c}_{a_Mk}^T\mathbf{z} \geq$$
$$\max_{\{\eta_0,\ldots,\eta_M\}\in\mathcal{P}} \min_{\underline{\mathbf{z}}_L\leq\mathbf{z}\leq\bar{\mathbf{z}}_L} \quad \eta_0\mathbf{c}_{yk}^T\mathbf{z} + \eta_1\mathbf{c}_{a_1k}^T\mathbf{z} + \cdots + \eta_M\mathbf{c}_{a_Mk}^T\mathbf{z}. \tag{27}$$

Combining Equality (25) with (26) and (27), we have:

$$\min_{\mathbf{z}\in\mathcal{Z}(\mathbf{x}_0,\epsilon)} \max(\mathbf{c}_{yk}^T\mathbf{z}, \mathbf{c}_{a_1k}^T\mathbf{z}, \ldots, \mathbf{c}_{a_Mk}^T\mathbf{z}) \geq \max_{\{\eta_0,\ldots,\eta_M\}\in\mathcal{P}} \min_{\underline{\mathbf{z}}_L\leq\mathbf{z}\leq\bar{\mathbf{z}}_L} \quad c_k(\boldsymbol{\eta})^T\mathbf{z}. \tag{28}$$

Since $\mathbf{z}_L = \mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L$, the right-hand-side of the above inequality can be rewritten as:

$$\min_{\mathbf{z}\in\mathcal{Z}(\mathbf{x}_0,\epsilon)} \max(\mathbf{c}_{yk}^T\mathbf{z}, \mathbf{c}_{a_1k}^T\mathbf{z}, \ldots, \mathbf{c}_{a_Mk}^T\mathbf{z}) \geq \max_{\boldsymbol{\eta}\in\mathcal{P}} \min_{\underline{\mathbf{z}}_{L-1}\leq\mathbf{z}\leq\bar{\mathbf{z}}_{L-1}} \quad c(\boldsymbol{\eta})^T(\mathbf{W}_L\mathbf{z} + \mathbf{b}_L),$$

which is exactly the claim of Theorem 3.1.

Proof of Theorem 4.1: For the simplicity of the presentation, assume that $a_0 = y$. Partition the set of possible values of $\mathbf{z}_L$ in the following sets:

$$\hat{\mathcal{Z}}_{a_i} = \{\mathbf{z}_L | [\mathbf{z}_L]_{a_i} \geq [\mathbf{z}_L]_{a_j}\ \forall j \neq i\}$$

If $\mathbf{z}_L \in \hat{\mathcal{Z}}_{a_i}$, then:

$$\begin{aligned}[\mathbf{z}_L]_{a_i} - [\mathbf{z}_L]_k &\geq [\mathbf{z}_L]_{a_j} - [\mathbf{z}_L]_k \quad \forall j \neq i \Rightarrow [\mathbf{z}_L]_{a_i} - [\mathbf{z}_L]_k \\ &= \max_{i=0,\ldots,M}\{[\mathbf{z}_L]_{a_i} - [\mathbf{z}_L]_k\} = \max_{i\in\{0,\ldots,M\}}\{\mathbf{c}_{a_i,k}^T\mathbf{z}_L\}\end{aligned}$$

Thus:

$$\begin{aligned}[\mathbf{z}_L]_{a_i} - [\mathbf{z}_L]_k = \max_{i=0,\ldots,M}\{\mathbf{c}_{a_i,k}^T\mathbf{z}_L\} &\geq \min_{\mathbf{z}_L\in\mathcal{Z}(\mathbf{x}_0,\epsilon)} \max_{i=0,\ldots,M}\{\mathbf{c}_{a_i,k}^T\mathbf{z}_L\} \\ &= \min_{\mathbf{z}_{L-1}\in\mathcal{Z}_{L-1}(\mathbf{x}_0,\epsilon)} \max_{i=0,\ldots,M}\{\mathbf{c}_{a_i,k}^T(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L)\} \\ &\geq \min_{\underline{\mathbf{z}}\leq\mathbf{z}_{L-1}\leq\bar{\mathbf{z}}} \max_{i=0,\ldots,M}\{\mathbf{c}_{a_i,k}^T(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L)\} \\ &= \min_{\underline{\mathbf{z}}\leq\mathbf{z}_{L-1}\leq\bar{\mathbf{z}}} \max_{\boldsymbol{\eta}\in\mathbb{P}} c(\boldsymbol{\eta})^T(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L)\end{aligned} \tag{29}$$

Note that the second inequality holds since the minimum is taken over a larger set in the right hand side of the inequality. Using the min-max inequality:

$$\min_{\underline{\mathbf{z}}\leq\mathbf{z}_{L-1}\leq\bar{\mathbf{z}}} \max_{\boldsymbol{\eta}\in\mathbb{P}} c(\boldsymbol{\eta})^T(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L) \geq \max_{\boldsymbol{\eta}\in\mathbb{P}} \min_{\underline{\mathbf{z}}\leq\mathbf{z}_{L-1}\leq\bar{\mathbf{z}}} c(\boldsymbol{\eta})^T(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L) = -J_k(\boldsymbol{\eta}) \tag{30}$$

Combining (29) and (30), and multiplying both sides by -1, we obtain:

$$[\mathbf{z}_L]_k - [\mathbf{z}_L]_{a_i} \leq J_k(\boldsymbol{\eta}) \tag{31}$$

On the other hand:

$$\begin{aligned}&\max_{\|\boldsymbol{\delta}\|_\infty\leq\epsilon} \min_{m=0,\ldots,M} \ell_{\text{xent}\setminus\mathcal{A}_m}\Big(\mathbf{z}_L(\mathbf{x}+\delta), a_m\Big) \\ &\leq \max_{\|\boldsymbol{\delta}\|_\infty\leq\epsilon} \ell_{\text{xent}\setminus\mathcal{A}_i}\Big(\mathbf{z}_L(\mathbf{x}+\delta), a_i\Big) \\ &\leq \max_{\underline{\mathbf{z}}_{L-1}\leq\mathbf{z}\leq\bar{\mathbf{z}}_{L-1}} \ell_{\text{xent}\setminus\mathcal{A}_i}(\mathbf{z}_L) \quad \text{s.t.} \quad \mathbf{z}_L = \mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L.\end{aligned} \tag{32}$$

Moreover, by the property of the cross-entropy loss, we have:

$$\ell_{\text{xent}\setminus\mathcal{A}_i}(\mathbf{z}_L) = \ell_{\text{xent}\setminus\mathcal{A}_i}(\mathbf{z}_L - [\mathbf{z}_L]_{a_i}\mathbf{1}) \tag{33}$$

Combining (31), (32) and (33), we have:

$$\begin{aligned}
&\max_{\|\boldsymbol{\delta}\|_\infty \le \epsilon} \min_{m=0,\dots,M} \ell_{\text{xent}\setminus\mathcal{A}_m}\Big(\mathbf{z}_L(\mathbf{x}+\delta), a_m\Big) \\
&\le \max_{\underline{\mathbf{z}}_{L-1} \le \mathbf{z}_{L-1} \le \bar{\mathbf{z}}_{L-1}} \ell_{\text{xent}\setminus\mathcal{A}_i}(\mathbf{z}_L) \quad \text{s.t.} \quad \mathbf{z}_L = \mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L. \\
&= \max_{\underline{\mathbf{z}}_{L-1} \le \mathbf{z}_{L-1} \le \bar{\mathbf{z}}_{L-1}} \ell_{\text{xent}\setminus\mathcal{A}_i}(\mathbf{z}_L - [\mathbf{z}_L]_{a_i}\mathbf{1}) \quad \text{s.t.} \quad \mathbf{z}_L = \mathbf{W}_L\mathbf{z}_{L-1} \\
&\le \max_{\underline{\mathbf{z}}_{L-1} \le \mathbf{z}_{L-1} \le \bar{\mathbf{z}}_{L-1}} \ell_{\text{xent}\setminus\mathcal{A}_i}(J_k(\boldsymbol{\eta}), a_i) \\
&= \max_{\underline{\mathbf{z}}_{L-1} \le \mathbf{z}_{L-1} \le \bar{\mathbf{z}}_{L-1}} \ell_{\text{xent}\setminus\mathcal{A}_0}(J_k(\boldsymbol{\eta}), a_0)
\end{aligned}$$

Summing up over all data points, the desired result is proven.

D. Details of β-Crown

In this section, we show how β-crown sub-problems can be obtained for neural networks without abstain classes and with multiple abstain classes respectively. Before proceeding, let us have a few definitions and lemmas.

Lemma D.1. ***(Zhang et al., 2019, Theorem 15)*** *Given two vectors* $\mathbf{u}$ *and* $\mathbf{v}$*, the following inequality holds:*

$$\mathbf{v}^\top \mathit{ReLU}(\mathbf{u}) \ge \mathbf{v}^\top \mathbf{D}_{\boldsymbol{\alpha}}\mathbf{u} + \mathbf{b}',$$

where $\mathbf{b}'$ *is a constant vector and* $\mathbf{D}_{\boldsymbol{\alpha}}$ *is a diagonal matrix containing* $\boldsymbol{\alpha}_j$*'s as free parameters:*

$$\mathbf{D}_{j,j}(\boldsymbol{\alpha}) = \begin{cases} 1, & \textit{if } \underline{\mathbf{z}}_j \ge 0 \\ 0, & \textit{if } \bar{\mathbf{z}}_j \le 0 \\ \boldsymbol{\alpha}_j, & \textit{if } \bar{\mathbf{z}}_j > 0 > \underline{\mathbf{z}}_j \textit{ and } \mathbf{v}_j \ge 0 \\ \frac{\bar{\mathbf{z}}_j}{\bar{\mathbf{z}}_j - \underline{\mathbf{z}}_j}, & \textit{if } \bar{\mathbf{z}}_j > 0 > \underline{\mathbf{z}}_j \textit{ and } \mathbf{v}_j < 0, \end{cases} \tag{34}$$

Definition D.2. The recursive function $\Omega(i,j)$ is defined as follows (Wang et al., 2021):

$$\Omega(i,i) = \boldsymbol{I}, \quad \Omega(i,j) = \boldsymbol{W}_i\boldsymbol{D}_{i-1}(\boldsymbol{\alpha}_{i-1})\Omega(i-1,j)$$

β-crown defines a matrix $\mathbf{S}$ for handling splits through the branch-and-bound process. The multiplier(s) $\boldsymbol{\beta}$ determines the branching rule.

$$\mathbf{S}_i[j][j] = \begin{cases} -1, & \text{if split } \mathbf{z}_i[j] \ge 0 \\ 1, & \text{if split } \mathbf{z}_i[j] < 0 \\ 0, & \text{if no split } \bar{\mathbf{z}}_j, \end{cases} \tag{35}$$

Thus, the verification problem of β-crown is formulated as:

$$\min_{\mathbf{z} in \mathcal{Z}} \mathbf{c}^T\big(\mathbf{W}^L\text{ReLU}(\mathbf{z}_{L-1}) + \mathbf{b}_{L-1}\big) \ge \min_{\mathbf{z} in \mathcal{Z}} \max_{\boldsymbol{\beta}_{L-1}} \mathbf{c}^T\big(\mathbf{W}^L\mathbf{D}_{L-1}\mathbf{z}_{L-1} + \mathbf{b}_{L-1}\big) + \boldsymbol{\beta}_{L-1}^\top\mathbf{S}_{L-1} \tag{36}$$

Having these definitions, we can write $\mathbf{P}, \mathbf{q}, \mathbf{a}$, and $\mathbf{d}$ explicitly as functions of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. $\mathbf{P} \in \mathbb{R}^{d_0 \times (\sum_{i=1}^{L-1} d_i)}$ is a block matrix $\mathbf{P} := \begin{bmatrix}\mathbf{P}_1^\top & \mathbf{P}_2^\top & \cdots & \mathbf{P}_{L-1}^\top\end{bmatrix}$, $\mathbf{q} \in \mathbb{R}^{\sum_{i=1}^{L-1} d_i}$ is a vector $\mathbf{q} := \begin{bmatrix}\mathbf{q}_1^\top & \cdots & \mathbf{q}_{L-1}^\top\end{bmatrix}^\top$. Moreover:

$$\mathbf{a} = [\Omega(L,1)\mathbf{W}_1]^\top \in \mathbb{R}^{d_0 \times 1},$$

$$\mathbf{P}_i = \mathbf{S}_i \Omega(i,1)\mathbf{W}_1 \in \mathbb{R}^{d_i \times d_0}, \quad \forall\, 1 \le i \le L-1$$

$$\mathbf{q}_i = \sum_{k=1}^{i} \mathbf{S}_i \Omega(i,k)\mathbf{b}_k + \sum_{k=2}^{i} \mathbf{S}_i \Omega(i,k)\mathbf{W}_k \underline{\mathbf{b}}_{k-1} \in \mathbb{R}^{d_i}, \quad \forall\, 1 \le i \le L-1$$

$$\mathbf{d} = \sum_{i=1}^{L} \Omega(L,i)\mathbf{b}_i + \sum_{i=2}^{L} \Omega(L,i)\mathbf{W}_i \underline{b}_{i-1}$$

$$\underline{b}_i = \begin{cases} 1, & \text{if } \underline{\mathbf{z}}_j \ge 0 \\ 0, & \text{if } \bar{\mathbf{z}}_j \le 0 \\ \boldsymbol{\alpha}_j, & \text{if } \bar{\mathbf{z}}_j > 0 > \underline{\mathbf{z}}_j \text{ and } \mathbf{v}_j \ge 0 \\ \frac{\bar{\mathbf{z}}_j}{\bar{\mathbf{z}}_j - \underline{\mathbf{z}}_j}, & \text{if } \bar{\mathbf{z}}_j > 0 > \underline{\mathbf{z}}_j \text{ and } \mathbf{v}_j < 0, \end{cases}$$

Now we extend the definition of g for the network consisting of multiple abstain classes. Let $\bar{\mathbf{z}}$ be the pre-activation value of vector z before applying ReLU function. We aim to solve the following verification problem:

$$\min_{\mathbf{z}_{L-1} \in \mathcal{Z}_{L-1}(\mathbf{x}_0, \epsilon)} \quad \max_{\boldsymbol{\eta} \in \mathcal{P}} c_k(\boldsymbol{\eta})^T (\mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L).$$

Applying Lemma D.1 to the above problem, we have:

$$\begin{aligned} &\min_{\mathbf{z}_{L-1} \in \mathcal{Z}_{L-1}(\mathbf{x}_0, \epsilon)} \quad \max_{\boldsymbol{\eta} \in \mathcal{P}} c_k(\boldsymbol{\eta})^T \Big(\mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L \Big) \\ \le &\min_{\mathbf{z}_{L-1} \in \mathcal{Z}_{L-1}(\mathbf{x}_0, \epsilon)} \quad \max_{\boldsymbol{\eta} \in \mathcal{P}} c_k(\boldsymbol{\eta})^T \Big(\mathbf{W}_L \mathbf{D}_{L-1}(\boldsymbol{\alpha}_{L-1}) \hat{\mathbf{z}}_{L-1} + \mathbf{b}_L \Big) \end{aligned}$$

Adding the β-crown Lagrangian multiplier to the above problem, it turns to:

$$\begin{aligned} &\min_{\mathbf{z}_{L-1} \in \mathcal{Z}_{L-1}(\mathbf{x}_0, \epsilon)} \quad \max_{\boldsymbol{\eta} \in \mathcal{P}} \quad c_k(\boldsymbol{\eta})^T \Big(\mathbf{W}_L \mathbf{D}_{L-1}(\boldsymbol{\alpha}_{L-1}) \hat{\mathbf{z}}_{L-1} + \mathbf{b}_L \Big) \le \\ &\min_{\mathbf{z}_{L-1} \in \mathcal{Z}_{L-1}(\mathbf{x}_0, \epsilon)} \quad \max_{\boldsymbol{\eta} \in \mathcal{P}, \boldsymbol{\alpha}_{L-1}, \boldsymbol{\beta}_{L-1}} \quad c_k(\boldsymbol{\eta})^T \Big(\mathbf{W}_L \mathbf{D}_{L-1}(\boldsymbol{\alpha}_{L-1}) \mathbf{z}_{L-1} + \mathbf{b}_L \Big) + \boldsymbol{\beta}_{L-1}^\top \mathbf{S}_{L-1} \mathbf{z}_{L-1} \\ &\le \max_{\boldsymbol{\alpha}_{L-1}, \boldsymbol{\beta}_{L-1}} \min_{\mathbf{z}_{L-1} \in \mathcal{Z}_{L-1}(\mathbf{x}_0, \epsilon)} \quad \max_{\boldsymbol{\eta} \in \mathcal{P}} \Big(c_k(\boldsymbol{\eta})^T \mathbf{W}_L \mathbf{D}_{L-1}(\boldsymbol{\alpha}_{L-1}) + \boldsymbol{\beta}_{L-1}^\top \mathbf{S}_{L-1} \Big) \hat{\mathbf{z}}_{L-1} \\ &+ c_k(\boldsymbol{\eta})^T \mathbf{b}_L = \max_{\boldsymbol{\alpha}_{L-1}, \boldsymbol{\beta}_{L-1}} \min_{\mathbf{z}_{L-1} \in \mathcal{Z}_{L-1}(\mathbf{x}_0, \epsilon)} \quad \max_{\boldsymbol{\eta} \in \mathcal{P}} \Big(c_k(\boldsymbol{\eta})^T \mathbf{W}_L \mathbf{D}_{L-1}(\boldsymbol{\alpha}_{L-1}) \\ &+ \boldsymbol{\beta}_{L-1}^\top \mathbf{S}_{L-1} \Big) \Big(\mathbf{W}_{L-1} \mathbf{z}_{L-2} + \mathbf{b}_{L-1} \Big) + c_k(\boldsymbol{\eta})^T \mathbf{b}_L \end{aligned}$$

Replace the definition of $\mathbf{A}^{(i)}$ in (Wang et al., 2021, Theorem 3.1) with the following matrix and repeat the proof.

$$\mathbf{A}^{(i)} = \begin{cases} c_k(\boldsymbol{\eta})^T \mathbf{W}_L, & \text{if} \quad i = L-1 \\ \Big(\mathbf{A}^{(i+1)} \mathbf{D}_{i+1}(\boldsymbol{\alpha}_{i+1}) + \boldsymbol{\beta}_{i+1}^\top \mathbf{S}_{i+1} \Big) \mathbf{W}_{i+1}, & \text{if} \quad 0 \le i \le L-2 \end{cases} \tag{37}$$

Note that the definition of $\mathbf{d}$ will be changed in the following way:

$$\mathbf{d} = c_k(\boldsymbol{\eta})^T \mathbf{b}_L + \sum_{i=1}^{L} \Omega(L,i)\mathbf{b}_i + \sum_{i=2}^{L} \Omega(L,i)\mathbf{W}_i \underline{b}_{i-1}$$

Moreover, $\Omega(L,j) = c_k(\boldsymbol{\eta})^T \boldsymbol{W}_L \boldsymbol{D}_{L-1}(\boldsymbol{\alpha}_{L-1})\Omega(L-1,j)$. The rest of the definitions remain the same.

D.1. Performance of β-crown on networks with multiple detection classes

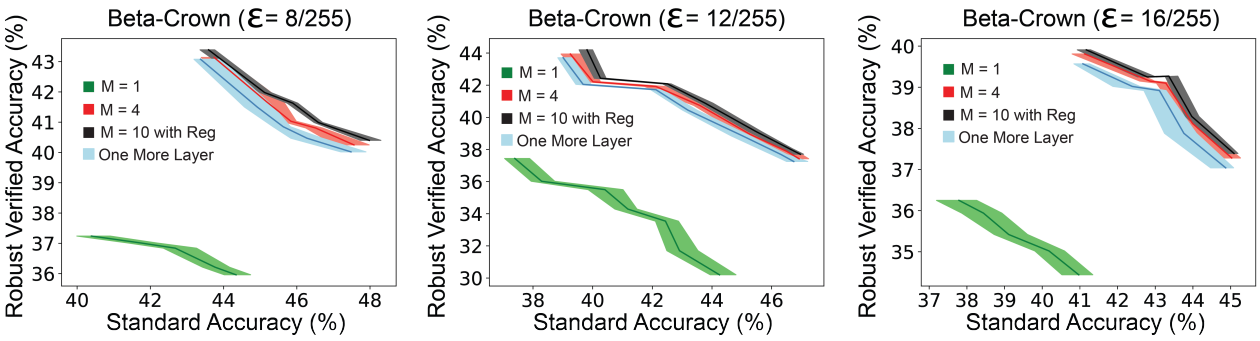

Figure 4. Performance of β-crown on verification of Neural Networks with single abstain, 4 abstain classes, 10 abstain classes with regularized, and a network with one more layer (single abstain) on CIFAR-10 dataset.

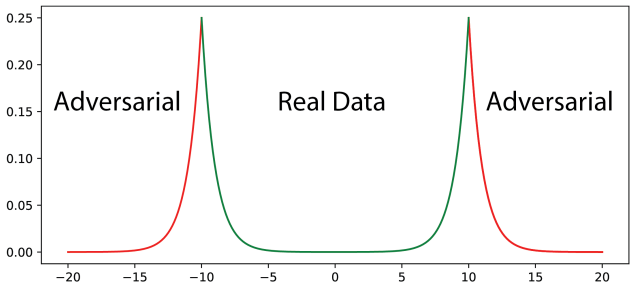

Figure 5. Distribution of adversarial and real data described in the example. While one linear classifier cannot separate the adversarial (red section) and real (green section) data points, two detection classes are capable of detecting adversarial examples.

E. Derivation of equation (12)

In this section, we show how to derive Equation E.

$$
\begin{aligned}
&\min_{\mathbf{z}_L \in \mathcal{Z}(\mathbf{x},\epsilon)} \quad \max\{\mathbf{c}_{yk}^T \mathbf{z}_L, \mathbf{c}_{a_1 k}^T \mathbf{z}_L, \ldots, \mathbf{c}_{a_M k}^T \mathbf{z}_L\} \\
&= \min_{\mathbf{z}_L \in \mathcal{Z}(\mathbf{x},\epsilon)} \quad \max_{\boldsymbol{\eta} \in \mathcal{P}} \sum_{i=0}^{M} \eta_i c_{a_i k}^T \mathbf{z}_L \\
&\geq \max_{\boldsymbol{\eta} \in \mathcal{P}} \quad \min_{\mathbf{z}_L \in \mathcal{Z}(\mathbf{x},\epsilon)} \sum_{i=0}^{M} \eta_i c_{a_i k}^T \mathbf{z}_L \\
&\geq \max_{\boldsymbol{\eta} \in \mathcal{P}} \quad \max_{\boldsymbol{\alpha},\boldsymbol{\beta} \geq 0} \eta_i c_{a_i k}^T \mathbf{z}_L \\
&= \max_{\boldsymbol{\alpha},\boldsymbol{\beta} \geq 0, \boldsymbol{\eta} \in \mathcal{P}} \Big(\sum_{i=0}^{M} \eta_i g_i(\mathbf{x}_0, \boldsymbol{\alpha}, \boldsymbol{\beta}) \triangleq G(\mathbf{x}_0, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}) \Big)
\end{aligned}
$$

F. A simple example on the benefits and pitfalls of having multiple abstain classes

In this example, we provide a simple toy example illustrating:

1. How adding multiple abstain classes can improve the detection of adversarial examples.
2. How detection with multiple abstain classes may suffer from a “model degeneracy" phenomenon.

Example: Consider a simple one dimensional data distributed where the read data is coming from the Laplacian distribution with probability density function $P_r(X = x) = \frac{1}{2}\exp(-|x|)$. Assume that the adversary samples are distributed according to the probability density function $P_a(X = x) = \frac{1}{4}(\exp(-|x - 10|) + \exp(-|x + 10|)$. Assume that $\frac{1}{3}$ data is real, and $\frac{2}{3}$ is coming from adversary. The adversary and the real data is illustrated in Fig 5.

Consider a binary neural network classifier with no hidden layer for detecting adversaries. More specifically, the neural network has two weight vectors w^r and w^a, and the bias values b^r and b^a. The network classifies a sample $\mathbf{x}$ as "real" if $w^r x + b^r > w^a x + b^a$; otherwise, it classifies the sample as out-of-distribution/abstain. The misclassification rate of this classifier is given by:

$$
\begin{aligned}
P(\text{error}) &= \frac{1}{3} P_{x \sim P_r}(w^a x + b^a > w^r x + b^r) + \frac{2}{3} P_{x \sim P_a}(w^a x + b^a < w^r x + b^r) \\
&= \frac{1}{3} P_{x \sim P_r}(x > \frac{b^r - b^a}{w^a - w^r}) + \frac{2}{3} P_{x \sim P_a}(x < \frac{b^r - b^a}{w^a - w^r}),
\end{aligned}
$$

where due to symmetry and scaling invariant, without loss of generality we assumed that $w^a - w^r > 0$. Let $t = \frac{b^r - b^a}{w^a - w^r}$. Therefore,

$$P(\text{error}) = \frac{1}{3}\int_t^{+\infty} \frac{1}{2}\exp(-|x|)dx + \frac{2}{3}\int_{-\infty}^{t} \frac{1}{4}(\exp(-|x-10|) + \exp(-|x+10|)dx \tag{38}$$

Thus, to find the optimal classifier, we require to determine the optimal t minimizing the above equation. One can numerically verify that the optimal t is given by $t^* = 5$ leading to the minimum misclassification rate of ≈ 0.34. This value is the optimal misclassification rate that can be achieved by our single abstain class neural network.

Now consider a neural network with two abstain classes. Assume that the weights and biases corresponding to the abstain classes are $w_1^a, w_2^a, b_1^a, b_2^a$, and the weight and bias for the real class is given by w_r and b_r. A sample x is classified as a real example if and only if both of the following conditions hold:

$$w^r x + b_r > w_1^a x + b_1^a \tag{39}$$
$$w^r x + b_r > w_2^a x + b_2^a, \tag{40}$$

otherwise, it is classified as an adversarial (out of distribution) sample. The misclassification rate of such classifier is given by:

$$P(\text{error}) = \frac{1}{3}P_{x\sim P_c}(\text{Conditions (39) hold}) + \frac{2}{3}P_{x\sim P_a}(\text{Conditions (39) do not hold}) \tag{41}$$

Claim 1: The point $w_1^a = -1, w_2^a = 1, b_1^a = b_2^a = 0, b^r = 5, w^r = 0$ is a global minimum of (41) with the optimum misclassification rate less than 0.1.

Proof: Define $t_1 = -\frac{b_1^a - b_r}{w_1^a - w^r}, t_2 = -\frac{b_2^a - b_r}{w_2^a - w^r}$. Considering all possible sign cases, it is not hard to see that at the optimal point, $w_1^a - w^r$ and $w_2^a - w^r$ have different signs. Without loss of generality, assume that $w_1^a - w^r < 0$ and $w_2^a - w^r > 0$. Then:

$$P(\text{error}) = \frac{1}{3}P_{x\sim P_c}(x \le t_1 \vee x \ge t_2) + \frac{2}{3}P_{x\sim P_a}(x \ge t_1 \wedge x \le t_2) \tag{42}$$

It is not hard to see that the optimal solution is given by $t_1^* = -5, t_2^* = 5$. Plugging these values in above equation, we can check that the optimal loss is less than 0.1. ■

Claim 1 shows that by adding an abstain class, the misclassification rate of the classifier goes down from 0.34 to below 0.1. This simple example illustrates the benefit of having multiple abstain classes. Next, we show that by having multiple abstain classes, we are prone to the "model degeneracy" phenomenon.

Claim 2: Let $\bar{w}_1^a = \bar{w}_2^a = 1, \bar{b}_1^a = \bar{b}_2^a = 0, \bar{w}^r = 0, \bar{b}^r = 5$. Then, there exists a point $(\tilde{w}, \tilde{b}) = (\tilde{w}_1^a, \tilde{w}_2^a, \tilde{b}_1^a, \tilde{b}_2^a, \tilde{w}^r, \tilde{b}^r)$ such that $(\tilde{w}, \tilde{b})$ is a local minimum of the loss function in (41) and $\|(\tilde{w}, \tilde{b}) - (\bar{w}, \bar{b})\|_2 \le 0.1$.

Proof: Let $t_1 = -\frac{b_1^a - b_r}{w_1^a - w^r}, t_2 = -\frac{b_2^a - b_r}{w_2^a - w^r}$. Notice that in a neighborhood of point $(\bar{w}, \bar{b})$, we have $w_1^a - w^r > 0$ and $w_2^a - w^r > 0$. Thus, after the loss function in (41) can be written as:

$$\begin{aligned}
\ell(t_1, t_2) &= \frac{1}{3}P_{x\sim P_c}(x \le t_1 \vee x \ge t_2) + \frac{2}{3}P_{x\sim P_a}(x \ge t_1 \wedge x \le t_2) \\
&= \frac{1}{3}P_{x\sim P_r}(x \ge \min(t_1, t_2)) + \frac{2}{3}P_{x\sim P_r}(x \le \min(t_1, t_2)) \\
&= \frac{1}{3}P_{x\sim P_r}(x \ge z) + \frac{2}{3}P_{x\sim P_r}(x \le z),
\end{aligned}$$

where $z = \min_{t_1, t_2}$. It suffices to show that the above function has a local minimum close to the point $\bar{z} = 5$ (see (Nouiehed & Razaviyayn, 2021)). Simplifying $\ell(t_1, t_2)$ as a function of z, we have:

$$\ell(t_1, t_2) = h(z) = \frac{1}{6}\exp(-z) + \frac{1}{3} - \frac{1}{6}\exp(-z - 10) + \frac{1}{6}\exp(z - 10)$$

By plotting $h(z)$, we can observe that it has a local minimum close to $\bar{z} = 5$. ■

This claim shows that by optimizing the loss, we may converge to the local optimum $(\tilde{w}, \tilde{b})$ where both abstain classes become essentially the same and we do not utilize the two abstain classes fully.

ϵ	Method	Standard Error (%)	Robust Verified Error (%)
$\epsilon_{\text{train}} = 8.8/255$ $\epsilon_{\text{test}} = 8/255$	Interval Bound Propagation (Gowal et al., 2018)	50.51	68.44
	IBP-CROWN (Zhang et al., 2019)	54.02	66.94
	(Balunovic & Vechev, 2019)	48.3	72.5
	Single Abstain (Sheikholeslami et al., 2021)	55.60	63.63
	Multiple Abstain Classes (Current Work)	56.72	61.45
	Multiple Abstain Classes (Verified by Beta-crown)	56.72	57.55
$\epsilon_{\text{train}} = 17.8/255$ $\epsilon_{\text{train}} = 16/255$	Interval Bound Propagation (Gowal et al., 2018)	68.97	78.12
	IBP-CROWN (Zhang et al., 2019)	66.06	76.80
	Single Abstain (Sheikholeslami et al., 2021)	66.37	67.92
	Multiple Abstain Classes (verified by IBP)	66.25	64.57
	Multiple Abstain Classes (Verified by Beta-crown)	66.25	62.81

Table 2. Standard and Robust Verified error of state-of-the-art approaches on CIFAR-10 dataset.

G. Deep neural networks with multiple abstain classes

In Table 2, we compare the performance of several state-of-the-art approaches to networks with multiple abstain classes where the network is deep as the one described in Sheikholeslami et al. (2021).