Improving Adversarial Robustness via Joint Classification and Multiple Explicit Detection Classes

Sina Baharlouei
USC
baharlou@usc.edu

Fatemeh Sheikholeslami[1]
Amazon Alexa AI
shfateme@amazon.com

Meisam Razaviyayn
USC
razaviya@usc.edu

Zico Kolter
Bosch Center for AI, CMU
zkolter@cs.cmu.edu

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 a provable framework can benefit by extension to networks with multiple explicit abstain classes, where the adversarial examples are adaptively assigned to those. We show that naïvely adding multiple abstain classes can lead to "model degeneracy", then 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 tradeoffs, outperforming state-of-the-art algorithms for various choices of number of abstain classes. Our code is available at `https://github.com/sinaBaharlouei/MultipleAbstainDetection`.

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 has designed various defense mechanisms against such 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 and 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 the 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 the ones developed in (Tjeng et al., 2017; Xiao et al., 2018), are not scalable to large networks. To develop scalable verifiers, the community turn to "inexact" verifiers which can only verify a subset of perturbations to the input data that the network can defend against successfully. These verifiers typically rely on 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 relatively superior performance compared to prior works.

IBP-CROWN (Zhang et al., 2019) combines IBP with novel linear relaxations to have a tighter approximation than standalone IBP. β-Crown (Wang et al., 2021) utilizes a branch-and-bound technique combined with the linear bounds in IBP-CROWN to tighten the relaxation gap further. While β-Crown demonstrates a tremendous performance gain over other verifiers such as Zhang et al. (2019); Fazlyab et al. (2019); Lu and Kumar (2019), it cannot be used as a tool in large-scale training procedures due to its computation-

[1]The work of FS was done when FS was with the Bosch Center for AI.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

ally expensive branch-and-bound search. One can adopt a composition of certified architectures to enhance the performance of the obtained model on both natural and adversarial accuracy (Müller et al., 2021; Horváth et al., 2022).

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 adversarial 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 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 and Gimpel (2016) develops a method based on a simple observation that, for real samples, the output of softmax layer is closer to 0 or 1 compared to out-of-distribution and adversarial examples where the softmax output entries are distributed more uniformly. DeVries and Taylor (2018); Sheikholeslami et al. (2020); Stutz et al. (2020) learn uncertainty regions around actual samples where the network prediction 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., 2021b) 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) as discussed in Tramer (2022). A more resilient approach is to jointly learn the detector and the classifier (Laidlaw and Feizi, 2019; Sheikholeslami et al., 2021; Chen et al., 2021a) by adding an auxiliary *abstain* output class capturing adversarial samples.

Building on these prior works, this paper develops a framework for detecting adversarial examples using multiple abstain classes. We observe that naïvely adding multiple abstain classes (in the existing framework of Sheikholeslami et al. (2021)) 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 novel regularizer and a training procedure to balance 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 on adversarial examples while maintaining the standard accuracy of the classifier.

Challenges and Contribution. We propose a framework for training and verifying robust neural nets with multiple detection classes. The resulting optimization problems for training and verifying such networks is a *constrained min-max* optimization problem over a probability simplex that is more challenging from an optimization perspective than the problems associated with networks with no or single detection classes. We devise an efficient algorithm for this problem. Furthermore, having multiple detectors leads to the "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 mechanism guaranteeing that all detectors contribute to detecting adversarial examples to the extent possible. We propose convergent algorithms for the verification (and training) problems using proximal gradient descent with Bregman divergence. Compared to networks with a single detection class, our experiments show that we enhance the robust verified accuracy by more than 5% and 2% on CIFAR-10 and MNIST datasets, respectively, for various perturbation sizes.

Roadmap. In section 2 we review interval bound propagation (IBP) and β-crown as two existing efficient methods for verifying the performance of multi-layer neural networks against adversarial attacks. We discuss how to train and verify joint classifier and detector networks (with a single abstain class) based on these two approaches. Section 3 is dedicated to the motivation and procedure of joint verification and classification of neural networks with **multiple** abstain classes. In particular, we extend IBP and β-crown verification procedures to networks with multiple detection classes. In section 4, we show how to train neural networks with multiple detection classes via IBP procedure. However, we show that the performance of the trained network cannot be improved by only increasing the number of detection classes due to "model degeneracy" (a phenomenon that happens when multiple detectors behave very similarly and identify the same adversarial examples). To avoid model degeneracy and to automatically/implicitly tune the number of detection classes, we introduce a regularization mechanism such that all detection classes are used in balance.

2 Background

2.1 Verification of feedforward neural networks

Consider an L-layer feedforward neural network with $\{\mathbf{W}_i, \mathbf{b}_i\}$ denoting the weight and bias parameters associated with layer i, and 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., $\sigma_i(\cdot) = \sigma(\cdot) = \text{ReLU}(\cdot),\ \forall i = 1, \dots, L-1$. Thus, our neural network can be described as

$$\mathbf{z}_i = \sigma(\mathbf{W}_i \mathbf{z}_{i-1} + \mathbf{b}_i)\ \forall i \in [L-1],\ \mathbf{z}_L = \mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L,$$

where $\mathbf{z}_0 = \mathbf{x}$ is the input to the neural network and $\mathbf{z}_i$ is the output of layer i and $[N]$ denotes the set $\{1, \dots, N\}$. Note that the activation function is not applied at the last layer. Further, we use $[\mathbf{z}]_i$ to denote the i-th element of the vector $\mathbf{z}$. 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. $\mathbf{e}_y$) denoting 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 correctly classify all the points inside $\mathcal{C}(\mathbf{x}_0, \epsilon)$. The objective function in Eq. (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 Eq. (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 and 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 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:

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

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 $\mathcal{C}(\mathbf{x}_0, \epsilon) = \{\mathbf{x} \,|\, \|\mathbf{x} - \mathbf{x}_0\|_\infty \leq \epsilon\}$, 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 for all $i \in [L]$:

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

Note that $|\mathbf{W}|$ denotes the element-wise absolute value of matrix $\mathbf{W}$. 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., $\underline{\mathbf{z}}_i \leq \mathbf{z}_i \leq \bar{\mathbf{z}}_i$, 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 $\mathbf{z}_i$. A neuron is called 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 a branch-and-bound (BAB) approach to split the input range of the neuron into two sub-domains $\mathcal{C}_{il} = \{\mathbf{x} \in \mathcal{C}(\mathbf{x}_0, \epsilon) |\, \hat{z}_i \leq 0\}$ and $\mathcal{C}_{iu} = \{\mathbf{x} \in \mathcal{C}(\mathbf{x}_0, \epsilon) |\, \hat{z}_i > 0\}$. The neuron operates linearly within each subdomain. Thus we can verify each subdomain separately. If we have N unstable nodes, the 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 the verifier visits it. 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 $\boldsymbol{\beta}$:

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

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 Training a joint robust classifier and detector

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 network "abstains" from declaring any of the original K classes for the given input. Let a be the abstain class. The classification network performs correctly on an adversarial image if it is

classified correctly as the class of the original (unperturbed) image (similar to robust networks without detectors) or it is classified as the abstain class (detected as an adversarial example). Hence, for input image $(\curvearrowleft_0, y)$ the network is verified against a certain class $k \neq y$ if

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

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.

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:

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

where the term L_{Natural} denotes the natural loss when no adversarial examples are considered. More precisely, $L_{\text{Natural}} = \frac{1}{n} \sum_{i=1}^{n} \ell_{\text{xent}}\big(\mathbf{z}_L(\mathbf{x}_i), y_i\big)$, 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,

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

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

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

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 $L_{\text{robust}}^{\text{abstain}}$ (Sheikholeslami et al., 2021; Gowal et al., 2018).

3 Verification of Neural Networks with Multiple Detection Classes

Motivation: The set of all adversarial images that can be generated within the ϵ-neighborhood of clean images might not be detectable only by a single detection class. Hence, the robust verified accuracy of the joint classifier and detector can be enhanced by introducing multiple abstain classes instead of a single abstain class to detect adversarial examples. This observation is illustrated in a simple example in Appendix F where we theoretically show that 2 detection classes can drastically increase the performance of the detector compared to 1 detection class. Note that a network with multiple detection classes can be equivalently modeled by another network with one more layer and a single abstain class. This added layer, which can be a fully connected layer with a max activation function, can merge all abstain classes and collapse them into 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. The reason is that increasing the number of layers leads to looser bounds in (4) for the last layer. To illustrate this fact, Figure 1 shows that the number of verified points by a $2-$layer neural network is higher than the number of points verified by an equivalent network with 3 layers.

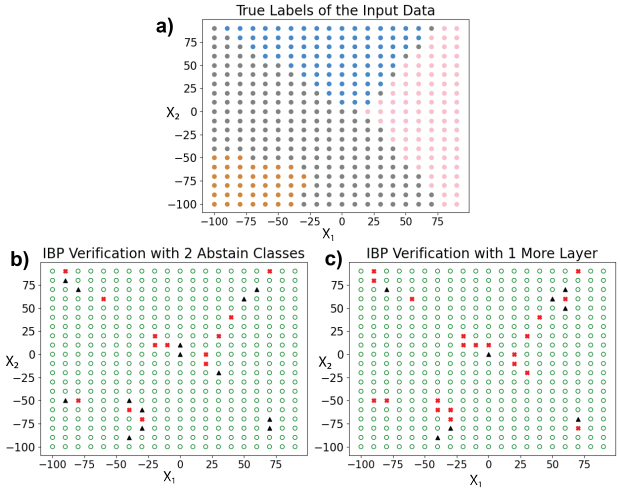

Figure 1: The IBP verification for 400 input data points of 2-layer and 3-layer neural networks. Part **(a)** shows the assigned four labels to data points. Part **(b)** demonstrates that IBP can verify 14 points using one of two abstain classes (black triangles), while it cannot verify 13 data points (red $\times$). **c)** shows that when IBP is applied to a network with one more layer and one detection class, 8 points are verified by the detection class, while it fails to verify 21 points. The description of both networks can be found in Appendix G.

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 the experiments on MNIST and CIFAR-10 datasets depicted in Figure 2. 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 network's output 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 $(\mathbf{x}, y)$ is verified if the network either correctly classifies it as class y 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 if

$$0 \leq \min_{\mathbf{z}_L \in \mathcal{Z}(\mathbf{x}_0, \epsilon)} \max\left\{\mathbf{c}_{yk}^T \mathbf{z}_L, \mathbf{c}_{a_1 k}^T \mathbf{z}_L, \ldots, \mathbf{c}_{a_M k}^T \mathbf{z}_L\right\}. \tag{9}$$

Since the set $\mathcal{Z}(\mathbf{x}_0, \epsilon)$ is nonconvex, verifying (9) is computationally expensive. The next subsections present two verifiers for solving (9) based on IBP and β-crown.

3.1 Verification with IBP

Using the IBP mechanism to relax the non-convex set $\mathcal{Z}(\mathbf{x}_0, \epsilon)$ leads to the following result for a given network with M detection classes:

Theorem 1 *Condition (9) is satisfied if for all $k \neq y$:*

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

is greater than or equal to zero where $\mathcal{P} = \{(\eta_0, \ldots, \eta_M) | \sum_{i=0}^{M} \eta_i = 1, \eta_i \geq 0, \forall i = 0, 1, \ldots, M\}$, and $c_k(\boldsymbol{\eta}) = \eta_0 \mathbf{c}_{yk} + \eta_1 \mathbf{c}_{a_1 k} \cdots + \eta_M \mathbf{c}_{a_M k}$. 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

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

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 4 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 and 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 and Pena, 2018, Corollary 2).

Algorithm 1 IBP verification of the network with multiple detection classes against class k

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: $[\mathbf{z}^*_{L-1}]_j = \begin{cases} [\underline{\mathbf{z}}_{L-1}]_j & \text{if } [\mathbf{W}_L^T c_k(\boldsymbol{\eta}^t)]_j \geq 0 \\ [\bar{\mathbf{z}}_{L-1}]_j & \text{otherwise.} \end{cases}$.
5: Set $\eta_m^{t+1} = \frac{\eta_m^t \exp(-2\nu(\mathbf{z}^{*t}_{L-1})^T \mathbf{W}_L^T \mathbf{c}_{a_m k})}{\sum_{j=0}^{M} \eta_j^t \exp(-2\nu(\mathbf{z}^{*t}_{L-1})^T \mathbf{W}_L^T \mathbf{c}_{a_j k})}$, for all $m \in [M]$ where $a_0 = y$.

Remark 2 ***Time Complexity Comparison:*** *Sheikholeslami et al. (2021) finds $J_k(\boldsymbol{\eta})$ using sorting and comparing a finite number of calculated values based on the last two layer weight matrices. While their algorithm finds the **exact** solution of η, it is limited to the one-dimensional case ($M = 1$). In this scenario, their algorithm requires $\mathcal{O}(n_{L-1} \log(n_{L-1}))$ evaluations to find the optimal η where n_{L-1} represents the number of nodes in the one to the last layer. Alternatively, our algorithm gives an ϵ-optimal solution in order of $\mathcal{O}(\frac{1}{\epsilon})$. By choosing $\epsilon = \mathcal{O}(\frac{1}{n_{L-1}})$, Algorithm 1 has almost the same complexity as Algorithm 1 in Sheikholeslami et al. (2021) in the one dimensional case. When $M > 1$, their algorithm cannot be utilized, while our algorithm finds the solution with the same order complexity. Note that the order complexity, in this case, is linear with respect to M.*

3.2 Verification with β-Crown

IBP verification is computationally efficient but less accurate than β–Crown. Hence, we focus on β–Crown verification of networks with multiple abstain classes in this section to obtain a tighter verifier. 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

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

The details of this inequality and the exact definition of the 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 $\boldsymbol{\beta}$ 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 detection classes

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

$[w]_+ = \max(w, 0)$ denotes the projection to the non-negative orthant in Algorithm 2.

4 Training of Neural Networks with Multiple Detection Classes

We follow a similar combination of loss functions to train a neural network consisting of multiple abstain classes as in (7). While the last term (L_{Natural}) can be computed efficiently, the first and second terms cannot be computed efficiently because even evaluating the functions L_{Robust} and

$L^{\text{Abstain}}_{\text{Robust}}$ requires maximizing nonconcave functions. Thus, we will minimize their upper bounds instead of minimizing these two terms. 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 $L^{\text{Abstain}}_{\text{Robust}}$ in (7), let us first start by clarifying its definition in the multi-abstain class scenario:

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

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 $L^{\text{Abstain}}_{\text{Robust}}$. Thus, we minimize an efficiently computable upper bound of this loss function as described in Theorem 3.

Theorem 3 *Let $\ell^{Abstain}_{Robust}(\mathbf{x}, y)$ is defined as:*

$$\max_{\|\delta\|\le\epsilon} \min\Big\{\ell_{xent}\big(\mathbf{z}_L(\mathbf{x}+\delta), y\big), \min_{m\in[M]} \ell_{xent}\big(\mathbf{z}_L(\mathbf{x}+\delta), a_m\big)\Big\}$$

Then,

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

where $J(\mathbf{x})$ is a vector whose k-th component equals $J_k(\mathbf{x})$ as defined in (11) and $\ell_{xent\setminus\mathcal{A}_0}(\mathbf{x}_0, y) := -\log\left(\frac{\exp(\mathbf{e}_y^T\mathbf{z}_L(\mathbf{x}_0))}{\sum_{i\in\mathcal{I}\setminus\mathcal{A}_0}\exp(\mathbf{e}_i^T\mathbf{z}_L(\mathbf{x}_0))}\right)$. *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 an incorrect classification. This mechanism enhances the performance of the network in 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 3. Thus our total training loss can be presented as:

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

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

Algorithm 3 Robust Neural Network Training

Input: Batches of data $\mathcal{D}_1,\dots,\mathcal{D}_R$, step-size ν, $\boldsymbol{\theta}(L)$: optimization parameters for loss L.
for $t = 1,\dots,R$ **do**
  Compute $J_o(\mathbf{x})\ \forall\,\mathbf{x}\in\mathcal{D}_t$, $\forall o\in[K]$ by Algorithm 1.
  Compute $\bar{L}_{\text{Robust}}$ on Batch $\mathcal{D}_t$ as Gowal et al. (2018).
  Compute $\bar{L}^{\text{abstain}}_{\text{Robust}}$ on Batch $\mathcal{D}_t$ using Theorem 3.
  $\boldsymbol{\theta}(L) = \boldsymbol{\theta}(L) - \nu\nabla_{\boldsymbol{\theta}}\big(\bar{L}_{\text{Robust}} + \lambda_1\bar{L}^{\text{abstain}}_{\text{Robust}} + \lambda_2 L_{\text{Natural}}\big)$

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 4 (10 abstains, unregularized), several abstain classes collapse together and capture similar adversarial patterns. Such a phenomenon, referred to as "model degeneracy" and illustrated with an example in Appendix F will prevent us from fully utilizing all abstain classes.

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:

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

where $f(\mathbf{z}_{L-1}) = \mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L$. The above regularizer penalizes the objective function if the average value of $\boldsymbol{\eta}$ coefficient corresponding to a given abstain class over all batch samples is smaller than a threshold (which is determined by the hyper-parameter γ). In other words, if an abstain class is not contributing enough to detect 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

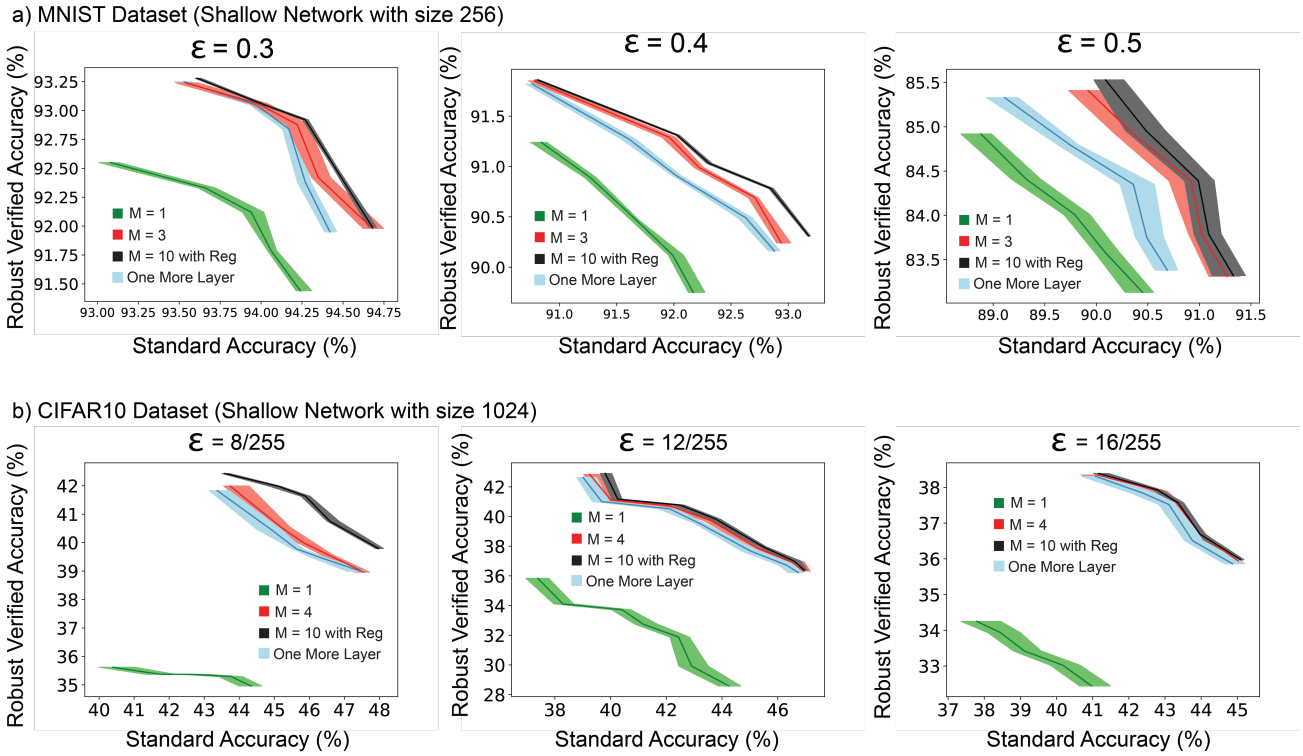

Figure 2: Performance of Multiple-abstain shallow networks on MNIST and CIFAR-10 datasets. We compared multiple abstain neural networks (both regularized and non-regularized versions) with the single abstain networks and networks with one more layer. The above and below rows demonstrate the trade-off between standard and robust verified accuracy on MNIST and CIFAR-10 datasets.

optimize $\boldsymbol{\eta}^{ik}$ of different samples and target classes jointly as they are coupled in the regularization term. Since optimizing (16) over the set of all n data points is infeasible for datasets with many samples, we solve the problem by choosing small data batches (≤ 64). We utilize the same Bregman divergence procedure used in Algorithm 1, while the gradient with respect to $\boldsymbol{\eta}^{ik}$ takes the regularization term into account as well.

Hyper-parameter tuning compared to the single-abstain scenario. In contrast to Sheikholeslami et al. (2021), our methodology has the additional hyper-parameter M (the number of abstain classes). Tuning this hyper-parameter can be costly since, in each run, we have to change the architecture (and potentially the stepsizes of the algorithm). This can be viewed as a potential additional computational overhead for training. However, as we observed in our experiments, setting $M = K$ combined with our regularization mechanism always leads to significant performance gains over training with a single abstain class. Thus, tuning the hyper-parameter M is unnecessary when we apply the regularization mechanism. In that case, we have two other hyper-parameters: γ is chosen from the set $\{0.1, 0.2, 0.5, 1, 2, 5, 10\}$ over $K + M$. The experiments show that 1 consistently works the best across different datasets and architectures. Further, μ in (16) is the regularization mechanism hyper-parameter. We determine μ by cross-validation over a wide range of numbers in $[0.1, 10]$. The experiments show that the optimal value of μ is between $[0.8, 1.5]$ depending on the network, dataset, and ϵ.

5 Numerical results

We devise diverse experiments on shallow and deep networks to investigate the effectiveness of joint classifiers and detectors with multiple abstain classes.

5.1 Training Setup

To train the networks on MNIST and CIFAR-10 datasets, we use Algorithm 3 as a part of an optimizer scheduler. Initially, we set $\lambda_1 = \lambda_2 = 0$. Thus, the network is trained without considering any abstain classes. In the second phase, we optimize the objective function (15), where we linearly increase ϵ from 0 to ϵ_{train}. Finally, we further tune the network on the fixed $\epsilon = \epsilon_{\text{train}}$. On both MNIST and CIFAR-10 datasets, we have used an Adam optimizer with a learning rate 5×10^{-4}. The networks are trained with four NVIDIA V100 GPUs. The trade-off between standard accuracy on clean images and robust verified accuracy can be tuned by changing λ_2 from 0 to $+\infty$ where the larger values correspond to more robust networks. For the networks with the regularizer addressing the model degeneracy issue, we choose γ by tuning it in the $[\frac{0.1}{K+M}, \frac{1.5}{K+M}]$. Our observations on both MNIST and CIFAR-10 datasets for different ϵ values show that the optimal value for γ is consistently

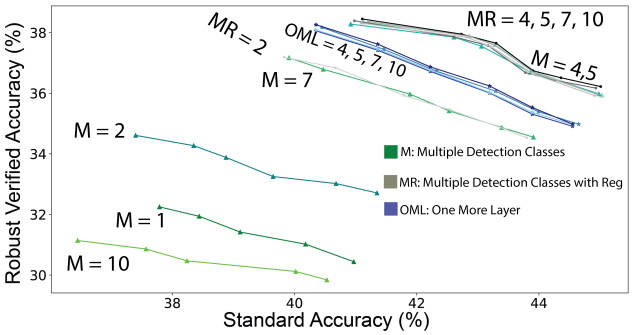

Figure 3: Performance of shallow networks with multiple detection classes without regularization, with regularization, and the network with one more layer on CIFAR-10 dataset. As demonstrated, the networks with regularization work consistently well for $4 \leq M \leq 10$, which is very close to the best performance we get from networks without regularization with 4 detection classes.

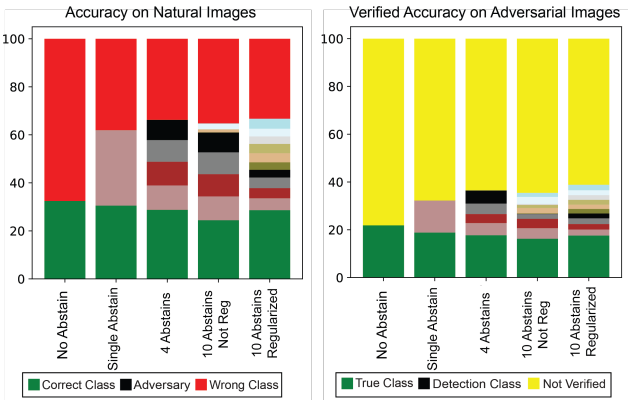

Figure 4: Distribution of Natural and adversarial images over different abstain classes on CIFAR-10 dataset. When there are 10 abstain classes, model degeneracy leads to lower performance than the baseline. Adding the regularization term (rightmost column) will utilize all abstain classes and enhance 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 ($\epsilon = \frac{16}{255}$).

close to $\frac{1}{K+M}$. Thus, we suggest choosing hyper-parameter $\gamma = \frac{1}{K+M}$ where K is the number of labels, and M is the number of detection classes. The optimal value for M is 4 for the CIFAR-10 and $M = 3$ for the MNIST dataset. By adding the "model degeneracy" regularizer, the obtained network has nearly the same performance for $M \in [4, 2K]$. Overall, we suggest to choose $M = K$ and $\gamma = \frac{1}{K+M}$ as the default values for hyper-parameters M and γ.

The model's robust verified accuracy will generally be increased by changing λ_2 from 0 to large values. As compensation, the standard accuracy of the model is reduced. Therefore, λ_2 determines the trade-offs between the standard and robust verified accuracy. The tradeoff curves presented in the figures are obtained by changing λ from 0 to 100.

5.2 Robust Verified Accuracy on Shallow Networks

In the first set of experiments depicted in Figure 2, we compare the performance of shallow networks with the fully optimized number of abstain classes to the single abstain network, the network with an additional layer, and the network with $M = K$ regularized by Equation (16). The shallow networks have one convolutional layer with sizes 256 and 1024 for MNIST and CIFAR-10 datasets, respectively. This convolutional layer is connected to the second (last) layer consisting of $K + M$ (20 for both MNIST and CIFAR-10) nodes. The optimal number of abstain classes is obtained by changing the number of them from 1 to 20 on both MNIST and CIFAR-10. The optimal value for the network trained on MNIST is $M = 3$, and $M = 4$ for the 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 ($M = K$) trained via the regularizer described in (16). The other is a network with one extra layer than the shallow network. This network has $K + M$ nodes in one to the last layer and $K + 1$ nodes in the last layer compared to the shallow network. Ideally, the set of models can be supported by such a network is a super-set of the original shallow network. Therefore, it has more capacity to classify images and detect adversarial attacks. However, due to the training procedure of IBP, which is sensitive to a 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. We observe that the unregularized network and the network with one more layer 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 $M = 4$, 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 its performance when M changes from the optimal value to larger values.

Figure 4 shows the percentage of adversarial examples captured by each abstain class ($M = 10$) on CIFAR-10 for both regularized and non-regularized networks. The hyper-parameter γ is set to $\frac{1}{K+M} = 0.05$. Next, we illustrate the performance of networks trained in the first set of experiments by β-crown in Figure 5. The networks verified by Beta-crown have 1% to 2% improvement in robust accuracy compared to the same networks verified by IBP.

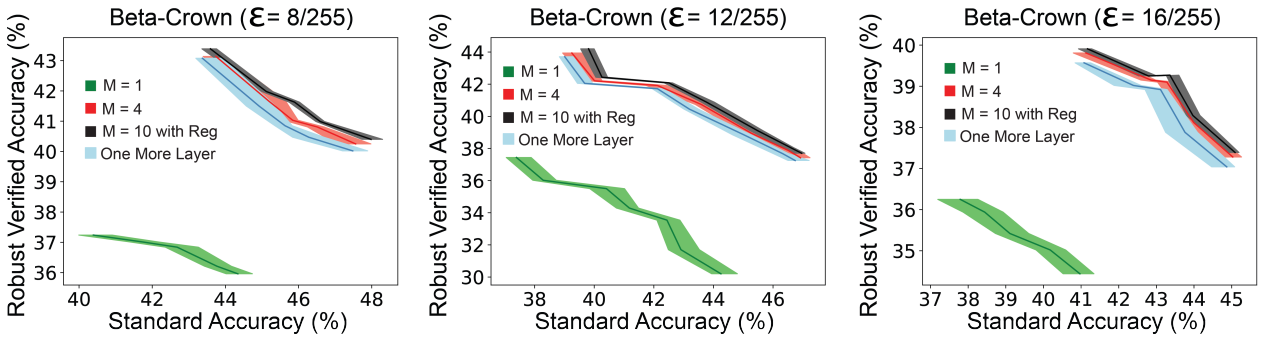

Figure 5: Performance of β-crown on verification of Neural Networks with single abstain, 4 abstain classes, 10 abstain classes with regularized, and networks with one more layer (single abstain) on CIFAR-10 dataset. $M = 1$ coincides with Sheikholeslami et al. (2021).

Method	St Err	PGD Success
(Madry et al., 2017)	2.91%	11.35%
IBP	2.27%	6.68%
IBP Crown	2.17%	12.06%
(Balunovic and Vechev, 2019)	2.77%	15.31%
(Sheikholeslami et al., 2021)	4.74%	4.15%
(Aquino et al., 2022)	4.81%	5.86%
Our Method	4.97%	**3.91%**

Table 1: Standard Error and PGD attack success rate on the MNIST dataset for different state-of-the-art approaches. The chosen ϵ for the PGD attack equals 0.4.

Method	St Err	PGD Success
(Madry et al., 2017)	49.78%	68.48%
IBP	50.51%	65.23%
IBP Crown	54.02%	65.42%
(Balunovic and Vechev, 2019)	48.3%	69.81%
(Sheikholeslami et al., 2021)	55.60%	63.63%
(Aquino et al., 2022)	50.25%	64.94%
Our Method	56.44%	**60.29%**

Table 2: Standard Error and PGD attack success rate on the CIFAR-10 dataset for different state-of-the-art approaches. The chosen ϵ for the PGD attack equals $8/255$.

5.3 Performance on Deep Neural Networks

Networks with multiple abstain classes achieve a superior trade-off between standard and verified robust accuracy on the deep networks as well. To demonstrate the performance of multiple abstain classes compared to state-of-the-art approaches, we trained deep neural networks on MNIST and CIFAR-10 datasets with different ϵ values. The results are reported in Table 4. The structure of the trained deep network is the same as the one described in Sheikholeslami et al. (2021) (see Appendix A).

While the verified robust accuracy guarantees the robustness of the networks against **all** attacks within the ϵ-neighborhood of each given test data point, one can argue that in many practical situations, being robust against certain adversarial attacks such as PGD attack (Madry et al., 2017) is sufficient. Table 1 and Table 2 demonstrate the performance of several state-of-the-art approaches for training robust neural networks against adversarial attacks on the MNIST and CIFAR-10 datasets, respectively. The performances are evaluated by the standard accuracy and robustness against the PGD attack on the test samples. The chosen ϵ is 0.4 for MNIST and $8/255$ for CIFAR-10, and the attacks are applied to each test sample using 100 iterations of the projected gradient descent (PGD). Our method achieves the best robustness against PGD attacks on both MNIST and CIFAR-10 datasets.

6 Conclusion

We improved the trade-off between standard accuracy and robust verifiable accuracy of the shallow and deep neural networks by introducing a training mechanism for networks with multiple abstain classes. We observed that increasing the number of abstain classes results in the "model degeneracy" phenomenon where not all abstain classes are utilized, and regular training can lead to solutions with poor performance in terms of standard and robust verified accuracy. To avoid the model degeneracy when the number of abstain classes is large, we propose a regularizer scheme penalizing the network if it does not utilize all abstain classes in balance. Our experiments demonstrate the superiority of the trained shallow and deep networks over state-of-the-art approaches on MNIST and CIFAR-10 datasets. We have used multiple detection classes to improve the performance of the verifiable neural networks. However, the application of multiple detection classes can be beyond such networks for detecting out-of-distribution samples or specific types of adversarial attacks.

7 Acknowledgment

The work of SB and MR was supported in part by the AFOSR Young Investigator Program award, by the NSF CAREER award #2144985, a gift from the USC-Meta Center for Research and Education in AI and Learning and by a gift from 3M.

References

B. Aquino, A. Rahnama, P. Seiler, L. Lin, and V. Gupta. Robustness against adversarial attacks in neural networks using incremental dissipativity. IEEE Control Systems Letters, 6:2341–2346, 2022.

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

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

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

J. Chen, J. Raghuram, J. Choi, X. Wu, Y. Liang, and S. Jha. Revisiting adversarial robustness of classifiers with a reject option. In The AAAI-22 Workshop on Adversarial Machine Learning and Beyond, 2021a.

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

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

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

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

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

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

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

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

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

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

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

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

M. Z. Horváth, M. N. Müller, M. Fischer, and M. Vechev. Robust and accurate–compositional architectures for randomized smoothing. arXiv preprint arXiv:2204.00487, 2022.

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

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

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

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

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

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

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

M. N. Müller, M. Balunović, and M. Vechev. Certify or predict: Boosting certified robustness with compositional architectures. In International Conference on Learning Representations (ICLR 2021). OpenReview, 2021.

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

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

N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against

deep neural networks. In 2016 IEEE symposium on security and privacy (SP), pages 582–597. IEEE, 2016.

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

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

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

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

D. Stutz, M. Hein, and B. Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In International Conference on Machine Learning, pages 9155–9166. PMLR, 2020.

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

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

F. Tramer. Detecting adversarial examples is (nearly) as hard as classifying them. In International Conference on Machine Learning, pages 21692–21702. PMLR, 2022.

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

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

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

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

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

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

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

Z. Zhu, G. Huang, J. Deng, Y. Ye, J. Huang, X. Chen, J. Zhu, T. Yang, J. Lu, D. Du, 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, pages 10492–10502, 2021.

A Implementation Details

In table A, we demonstrate the structure of the deep networks used in experiments of Table 4.

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

Table 3: 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 Bregman-Divergence Method for Optimizing a Convex Function Over a Probability Simplex

In this section, we use the Bregman divergence method to optimize a convex optimization problem over a probability simplex. 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 Bregman 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 constraint 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 the proximal gradient descent method to the above problem, we have:

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

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

By simplifying the above problem, it turns to:

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

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

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

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

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

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 1 and Theorem 3.

Proof of Theorem 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:

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

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 1.

Proof of Theorem 3: 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:

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

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

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

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

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

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

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

Definition 5 *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.

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

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

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

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 $\mathbf{P} := \big[\mathbf{P}_1^\top \ \mathbf{P}_2^\top \ \cdots \ \mathbf{P}_{L-1}^\top\big]$, $\mathbf{q} \in \mathbb{R}^{\sum_{i=1}^{L-1} d_i}$ is a vector $\mathbf{q} := \big[\mathbf{q}_1^\top \ \cdots \ \mathbf{q}_{L-1}^\top\big]^\top$. 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 the 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 4 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.

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 are illustrated in Fig 6.

Consider a binary neural network classifier with no hidden layer for detecting adversaries. 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.

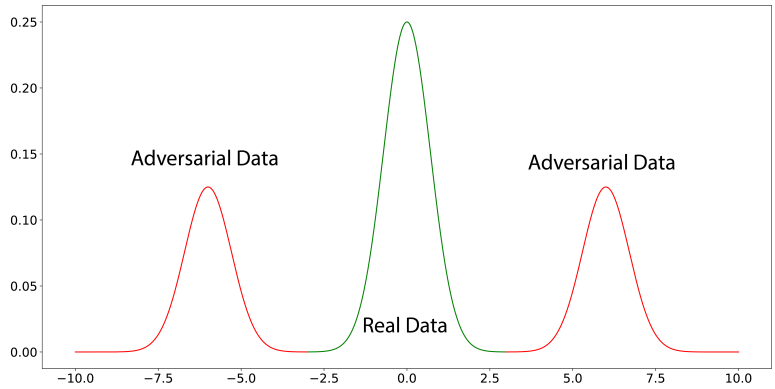

Figure 6: Distribution of adversarial and real data described in the example. While one linear classifier cannot separate the adversarial (red section) and real (green section) data points, two detection classes can detect adversarial examples.

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 the 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 the 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 and Razaviyayn, 2021)). Simplifying $\ell(t_1, t_2)$ as a function of z, we have:

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

Plotting $h(z)$ shows 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.

G Structure of Neural Networks in Section 3

In Section 3 we introduced a toy example in the Motivation subsection to show how loser IBP bounds can become when we go from a 2-layer network to an equivalent 3-layer network. The structure of the 2-layer neural networks is as follows:

$$\mathbf{z}_2(\mathbf{x}) = \mathbf{W}_2 \text{ReLU}(\mathbf{W}_1 \mathbf{x}),$$

where $\mathbf{x}$ is the 2-dimensional input, $\mathbf{W}_1 = \begin{pmatrix} 0.557 & -0.296 & -0.449 \\ -0.474 & -0.504 & 0.894 \\ -0.0208 & 0.0679 & 0.901 \end{pmatrix}$, and $\mathbf{W}_2 = \begin{pmatrix} 0.817 & -0.376 & 0.36 \\ 0.524 & 0.530 & 0.0557 \\ 0.0753 & 0.191 & 0.744 \\ -0.547 & 0.660 & -0.718 \end{pmatrix}$.

Note that the input data is 2 dimensional, but we add an extra one for incorporating bias into $\mathbf{W}_1$ and $\mathbf{W}_2$. The chosen ϵ for each data point equals 1.

H Experiments on Deep Neural Networks

To compare the performance of our proposed approach to other state-of-the-art methods on deep neural networks, we run the methods on the network with the structure described in Appendix A. The results are reported in Table 4.

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 and 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
$\epsilon_{\text{train}} = 17.8/255$ $\epsilon_{\text{test}} = 16/255$	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 4: Standard and Robust Verified error of state-of-the-art approaches on CIFAR-10 dataset.

I Limitations

The proposed framework for training and verifying joint detector and classifier networks defines the uncertainty set on each sample as an L_∞ norm ball. The results can be extended to other L_p norm balls (L_1 or L_2) by changing the Interval Bound Propagation (Gowal et al., 2018) procedure to other constraint sets defined by L_p balls. However, the experiments in the paper are performed on the L_∞ constraint sets. Furthermore, the networks in the numerical section are trained on MNIST and CIFAR-10 datasets due to the expensive and complex training procedure of **verifiable** neural networks. The training procedure of the verifiably robust neural networks is yet limited to these datasets, even for the fastest methods such as IBP. Besides, on large-scale datasets with millions of samples and many different classes, the optimal M might be much larger compared to its optimal value of $M = 3$ and $M = 4$ on MNIST and CIFAR-10 datasets, respectively. Therefore, it is crucial to devise techniques for training large verifiable neural networks in general and networks with multiple detection classes in particular.

J Societal Impacts

Given the susceptibility of presently trained neural networks to adversarial examples and out-of-distribution samples, the deployment of such models in critical applications like self-driving cars has been the subject of debate. Ensuring the reliability and safety of neural networks in unpredictable and adversarial environments requires the development of mechanisms that guarantee the models' robustness. Our current study proposes a systematic approach for training and validating neural networks against adversarial attacks. From a broader perspective, establishing verifiable assurances for the performance of artificial intelligence (AI) models alleviates ethical and safety concerns associated with AI systems.