Connecting Certified and Adversarial Training

Yuhao Mao [1] **Mark Müller** [1] **Marc Fischer** [1] **Martin Vechev** [1]

Abstract

Training certifiably robust neural networks remains a notoriously hard problem. While adversarial training optimizes *under-approximations* of the worst-case loss, which leads to insufficient regularization for certification, certified training methods, optimize loose *over-approximations*, leading to over-regularization and poor accuracy. In this work, we propose TAPS, a novel certified training method combining IBP and PGD training to optimize more precise, although not necessarily sound, worst-case loss approximations, reducing over-regularization and increasing certified accuracy. Empirically, TAPS achieves a new state-of-the-art in many settings, e.g., reaching a certified accuracy of 22% on TinyImageNet for ℓ_∞-perturbations with radius $\epsilon = 1/255$.

1. Introduction

Adversarial robustness, *i.e.*, a neural network's resilience to small input perturbations (Biggio et al., 2013; Szegedy et al., 2014), has established itself as an important research area. Certification methods can rigorously prove such robustness. *Adversarial training* methods, such as PGD (Madry et al., 2018), aim to improve robustness by training with samples that are perturbed to approximately maximize the training loss. Thus optimizing an *under-approximation* of the worst-case loss, they *empirically* improve robustness significantly, but are insufficient to certify robustness. *Certified training* methods like IBP (Mirman et al., 2018; Gowal et al., 2018), in contrast, optimize over-approximations of the worst-case loss, allowing them to increase certified accuracies at the cost of over-regularization that leads to reduced accuracies. Recent methods (Balunovic & Vechev, 2020; Palma et al., 2022; Müller et al., 2022b), compute *precise* but unsound approximations of the worst-case loss, reducing regularization and achieving higher accuracies.

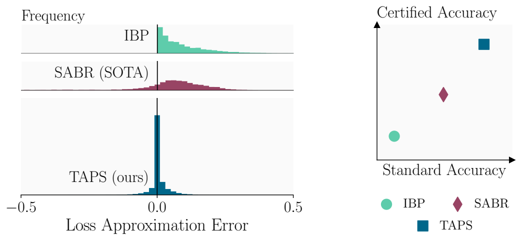


Figure 1: Histograms of the worst-case loss approximation errors over the test set (left) for different training methods show that TAPS (our work) achieves the most precise approximations and highest certified accuracy (right). Results shown here are for a small CNN3.

This Work proposes **T**raining via **A**dversarial **P**ropagation through **S**ubnetworks (TAPS), a novel (unsound) certified training method, based on more precise worst-case loss approximations and increasing both certified and standard accuracies. We showcase this relationship between worst-case loss approximation error and accuracies in Figure 1. Compared to SABR (Müller et al. (2022b), ♦ the current state-of-the-art), TAPS (■) enjoys a 5-fold mean approximation error reduction and significantly reduced variance (Figure 1 left), leading to improved certified and natural accuracies (right). The key technical insight behind TAPS is to combine IBP and PGD training via a gradient connector, a novel mechanism that allows for joint training, such that the over-approximation of IBP and under-approximations of PGD cancel out. We demonstrate, in an extensive empirical study, that TAPS yields exceptionally tight worst-case loss approximations which allow it to improve on state-of-the-art results for MNIST, CIFAR-10, and TinyImageNet.

2. Background

We consider a classifier $F\colon \mathcal{X} \mapsto \mathcal{Y}$ parameterized by weights $\boldsymbol{\theta}$ and predicting a class $y_{\text{pred}} = F(\boldsymbol{x}) = \arg\max_{y\in\mathcal{Y}} f_y(\boldsymbol{x})$ for every input $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ with label $y \in \mathcal{Y} = \{1, \dots, K\}$ where $\boldsymbol{f}\colon \mathcal{X} \mapsto \mathbb{R}^{|\mathcal{Y}|}$ is a neural network, assigning a numerical logit $o_i := f_i(\boldsymbol{x})$ to each class i.

Adversarial Robustness We call a classifier adversarially robust on an ℓ_∞-norm ball $\mathcal{B}(\boldsymbol{x}, \epsilon) := \{\boldsymbol{x}' \mid \|\boldsymbol{x}' - \boldsymbol{x}\|_\infty \leq \epsilon\}$ if it classifies all elements within the ball to the correct class, *i.e.*, $F(\boldsymbol{x}') = y$ for all perturbed inputs $\boldsymbol{x}' \in \mathcal{B}(\boldsymbol{x}, \epsilon)$.

[1]Department of Computer Science, ETH Zürich, Zürich, Switzerland. Correspondence to: Yuhao Mao <yuhmao@ethz.ch>.

Presented at the 2^{nd} Workshop on Formal Verification of Machine Learning, co-located with the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA., 2023. Copyright 2023 by the author(s).

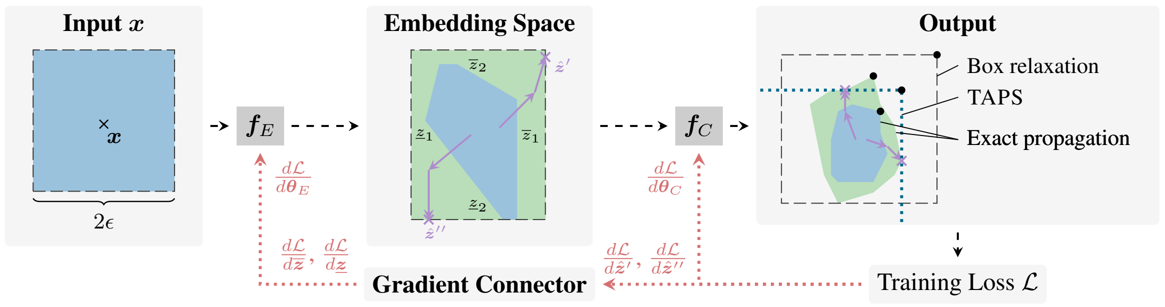

Input x
2ϵ
f_E
Embedding Space
$\overline{z}_2$
$\underline{z}_1$
$\overline{z}_1$
$\underline{z}_2$
$\hat{z}'$
$\hat{z}''$
f_C
Output
Box relaxation
TAPS
Exact propagation
$\frac{d\mathcal{L}}{d\theta_E}$
$\frac{d\mathcal{L}}{d\theta_C}$
$\frac{d\mathcal{L}}{d\overline{z}}, \frac{d\mathcal{L}}{d\underline{z}}$
$\frac{d\mathcal{L}}{d\hat{z}'}, \frac{d\mathcal{L}}{d\hat{z}''}$
Gradient Connector
Training Loss $\mathcal{L}$

Figure 2: Overview of TAPS training. First, forward propagation (-→) of a region $B(\boldsymbol{x}, \epsilon)$ (🟦, left) around an input $\boldsymbol{x}$ (×) through the feature extractor $\boldsymbol{f}_E$ yields the exact reachable set (🟦, middle) and its IBP approximation $[\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}]$ (⬚, middle) in the embedding space. Further IBP propagation through the classifier $\boldsymbol{f}_C$ would yield an imprecise box approximation (⬚, right) of the reachable set (🟦, right). Instead, TAPS conducts an adversarial attack (→) in the embedding space IBP approximation (🟩) yielding an under-approximation (⬚) of its reachable set (🟩, right). We illustrate the points with the worst-case loss in every output region with • and enable back-propagation (·→) through the attack via our gradient connector.

Neural Network Certification can formally *prove* robustness properties of a neural network, *i.e.*, that all inputs in the region $\mathcal{B}(\boldsymbol{x}, \epsilon)$ yield the correct classification. Interval bound propagation (IBP) (Mirman et al., 2018; Gowal et al., 2018) is a particularly simple yet effective certification method. It computes an over-approximation of a network's reachable set by propagating the input region $\mathcal{B}(\boldsymbol{x}, \epsilon)$ layer-by-layer through the network (see Gowal et al. (2018)) and representing the output of each layer as a Box (each dimension is described as an interval) until we obtain upper (and lower) bounds $[\underline{\boldsymbol{o}}^{\Delta}, \overline{\boldsymbol{o}}^{\Delta}]$ on the logit differences $\boldsymbol{o}^{\Delta} := \boldsymbol{o} - o_y \mathbf{1}$. Showing $\overline{o}_i^{\Delta} > 0$ for $i \neq y$ proves robustness.

Training for Robustness aims to find model parameters $\boldsymbol{\theta}$ that minimize the expected worst-case loss $\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x},y} \left[\max_{\boldsymbol{x}' \in \mathcal{B}(\boldsymbol{x},\epsilon)} \mathcal{L}(\boldsymbol{x}', y)\right]$. As the inner maximization objective can generally not be solved exactly, it is often under- or over-approximated giving rise to adversarial and certified training, respectively. *Adversarial Training* optimizes a lower bound on the inner maximization problem by training the network with concrete samples $\boldsymbol{x}' \in \mathcal{B}(\boldsymbol{x}, \epsilon)$ that (approximately) maximize the loss function. A well-established method for this is *Projected Gradient Descent (PGD)* training (Madry et al., 2018) using the loss $\mathcal{L}_{\text{CE}}(\boldsymbol{x}, y) := \ln\big(1 + \sum_{i \neq y} \exp(f_i(\boldsymbol{x}) - f_y(\boldsymbol{x}))\big)$. Starting from a random initialization point $\hat{\boldsymbol{x}}_0 \in \mathcal{B}(\boldsymbol{x}, \epsilon)$, it performs N update steps $\hat{\boldsymbol{x}}_{n+1} = \Pi_{\mathcal{B}(\boldsymbol{x},\epsilon)} \hat{\boldsymbol{x}}_n + \eta \operatorname{sign}(\nabla_{\hat{\boldsymbol{x}}_n} \mathcal{L}_{\text{CE}}(\hat{\boldsymbol{x}}_n, y))$ with step size η and projection operator Π. Networks trained this way typically exhibit good empirical robustness but remain hard to formally certify. *Certified Training*, in contrast, aims to train *certifiably* robust networks and optimizes a (sound) upper bound of the inner maximization problem, by evaluating the loss $\mathcal{L}_{\text{CE}}$ with upper bounds on the logit differences $\overline{\boldsymbol{o}}^{\Delta}$, e.g. via IBP:

$$\mathcal{L}_{\text{IBP}}(\boldsymbol{x}, y, \epsilon) := \ln\big(1 + \sum_{i \neq y} \exp(\overline{o}_i^{\Delta})\big). \tag{1}$$

3. Precise Worst-Case Loss Approximation

We now introduce TAPS, illustrated in Figure 2. The key insight behind TAPS is that PGD training and IBP training complement each other perfectly: We can combine them sequentially such that the over-approximation errors incurred during IBP are compensated by the under-approximations of PGD. We first partition a neural network $\boldsymbol{f}$ with weights $\boldsymbol{\theta}$ into a *feature extractor* $\boldsymbol{f}_E$ and a *classifier* $\boldsymbol{f}_C$ with parameters $\boldsymbol{\theta}_E$ and $\boldsymbol{\theta}_C$, respectively, such that we have $\boldsymbol{f}_{\boldsymbol{\theta}} = \boldsymbol{f}_C \circ \boldsymbol{f}_E$ and $\boldsymbol{\theta} = \boldsymbol{\theta}_E \cup \boldsymbol{\theta}_C$. We refer to the output space of the feature extractor as the *embedding space*. Given an input sample $\boldsymbol{x}$ (illustrated as × in Figure 2) and a corresponding input region $\mathcal{B}(\boldsymbol{x}, \epsilon)$ (🟦 in the input panel), training proceeds as follows: During the forward pass (black dashed arrows -→), we first use IBP to compute a Box over-approximation $[\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}]$ (dashed box ⬚) of the feature extractor's exact reachable set (blue region 🟦), shown in the middle panel of Figure 2. Then, we conduct separate adversarial attacks (→) within this region in the embedding space (🟩) to bound all output dimensions of the classifier. This yields latent adversarial examples $\hat{\boldsymbol{z}} \in [\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}]$, defining the TAPS bounds $\overline{\boldsymbol{o}}^{\Delta}_{\text{TAPS}}$ (dotted lines ⬚ in the output space) on the network's output. This way, the *under-approximation* of the classifier via PGD, partially compensates the *over-approximation* of the feature extractor via IBP. Full IBP propagation, in contrast, continues to exponentially accumulate approximation errors, yielding the much larger dashed box ⬚. We now compute the TAPS loss $\mathcal{L}_{\text{TAPS}}$ analogously to $\mathcal{L}_{\text{IBP}}$ (Eq. (1)) by plugging the TAPS bound estimate $\overline{\boldsymbol{o}}^{\Delta}_{\text{TAPS}}$ into the Cross-Entropy loss. Comparing the resulting losses (illustrated as • and growing towards the top right), we see that while the TAPS bounds are not necessarily sound, they yield a much better approximation of the true worst-case loss. During the backward

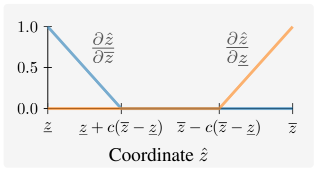


Figure 3: Gradient connector visualization.

pass (orange dotted arrows ⇢ in Figure 2), we compute the gradients w.r.t. the classifier's parameters $\boldsymbol{\theta}_C$ and the latent adversarial examples $\hat{\boldsymbol{z}}$ (classifier input) as usual. However, to compute the gradients w.r.t. the feature extractor's parameters $\boldsymbol{\theta}_F$, we have to compute (pseudo) gradients of the latent adversarial examples $\hat{\boldsymbol{z}}$ w.r.t. the box bounds $\underline{\boldsymbol{z}}$ and $\overline{\boldsymbol{z}}$. As these gradients are not well defined, we introduce the *gradient connector*, discussed next, as an interface between the feature extractor and classifier, imposing such pseudo gradients. This allows us to train $\boldsymbol{f}_E$ and $\boldsymbol{f}_C$ *jointly*, leading to a feature extractor that minimizes approximation errors and a classifier that is resilient to the spurious points included in the approximation. Additionally, TAPS uses an IBP-based regularization (see App. A).

Gradient Connector The key function of the gradient connector is to enable gradient computation through the adversarial example search in the embedding space. Using the chain rule, this only requires us to define the (pseudo) gradients $\frac{\partial \hat{\boldsymbol{z}}}{\partial \underline{\boldsymbol{z}}}$ and $\frac{\partial \hat{\boldsymbol{z}}}{\partial \overline{\boldsymbol{z}}}$ of the latent adversarial examples $\hat{\boldsymbol{z}}$ w.r.t. the box bounds $\underline{\boldsymbol{z}}$ and $\overline{\boldsymbol{z}}$ on the feature extractor's outputs. Below, we will focus on the i^{th} dimension of the lower box bound $\underline{z}_i$ and note that other dimensions and the upper bounds follow analogously.

As the latent adversarial examples can be seen as multivariate functions in the box bounds, we obtain the general form $\frac{d\mathcal{L}}{d\underline{z}_i} = \sum_j \frac{d\mathcal{L}}{d\hat{z}_j}\frac{\partial \hat{z}_j}{\partial \underline{z}_i}$, depending on all dimensions of the latent adversarial example. However, the j^{th} dimension of the latent adversarial example $\hat{z}_j$ is independent of the bounds in the i^{th} dimension $\underline{z}_i$ and $\overline{z}_i$ (for $i \neq j$) as Box approximations do not encode any relational information between dimensions. Therefore, we have $\frac{\partial \hat{z}_j}{\partial \underline{z}_i} = 0$ for $i \neq j$ and obtain $\frac{d\mathcal{L}}{d\underline{z}_i} = \frac{d\mathcal{L}}{d\hat{z}_i}\frac{\partial \hat{z}_i}{\partial \underline{z}_i}$, leaving only $\frac{\partial \hat{z}_i}{\partial \underline{z}_i}$ for us to define.

The most natural gradient connector is the *binary connector*, *i.e.*, set $\frac{\partial \hat{z}_i}{\partial \underline{z}_i} = \mathbb{1}_{\hat{z}_i = \underline{z}_i}$, as a valid sub-gradient for the projection in PGD. However, the latent adversarial input often does not lie on a corner of the Box approximation, leading to sparse gradients and thus a less well-behaved optimization problem. More importantly, the binary connector is very sensitive to the distance between (local) loss extrema and the box boundary and thus inherently ill-suited to gradient-based optimization. For example, a local extremum at $\hat{z}_i$ would induce $\frac{\partial \hat{z}_i}{\partial \underline{z}_i} = 1$ in the box $[\hat{z}_i, 0]$, but $\frac{\partial \hat{z}_i}{\partial \underline{z}_i} = 0$ for $[\hat{z}_i - \epsilon, 0]$, even for arbitrarily small ϵ.

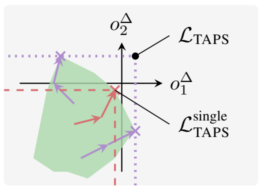

o_2^Δ
$\mathcal{L}_{\text{TAPS}}$
o_1^Δ
$\mathcal{L}_{\text{TAPS}}^{\text{single}}$

Figure 4: Bounds on $o_i^\Delta := o_i - o_t$ obtained via single (- -) and multi-estimator (⋯) PGD and the points maximizing the respective losses: × for $\mathcal{L}_{\text{TAPS}}^{\text{single}}$ and • for $\mathcal{L}_{\text{TAPS}}$.

To alleviate both of these problems, we consider a *linear connector*, *i.e.*, set $\frac{\partial \hat{z}_i}{\partial \underline{z}_i} = \frac{\overline{z}_i - \hat{z}_i}{\overline{z}_i - \underline{z}_i}$. However, even when our latent adversarial example is very close to one bound, the linear connector would induce non-zero gradients w.r.t. to the opposite bound. To remedy this undesirable behavior, we propose the *rectified linear connector*, setting $\frac{\partial \hat{z}_i}{\partial \underline{z}_i} = \max(0, 1 - \frac{\hat{z}_i - \underline{z}_i}{c(\overline{z}_i - \underline{z}_i)})$ where $c \in [0, 1]$ is a constant (visualized in Figure 3 for $c = 0.3$). Observe that it recovers the binary connector for $c = 0$ and the linear connector for $c = 1$. To prevent gradient sparsity ($c \leq 0.5$) while avoiding the above-mentioned counterintuitive gradient connections ($c \geq 0.5$), we set $c = 0.5$ unless indicated otherwise. When the upper and lower bounds agree in the i^{th} dimension, we set both gradients to $\frac{\partial \hat{z}_i}{\partial \underline{z}_i} = \frac{\partial \hat{z}_i}{\partial \overline{z}_i} = 0.5$.

TAPS Loss & Multi-estimator PGD The standard PGD attack, used in adversarial training, henceforth called *single-estimator* PGD, aims to maximize the loss $\mathcal{L}_{\text{CE}}$ of a single input. In the context of TAPS, this results in the overall loss

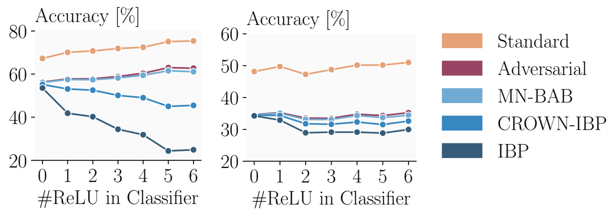
$$\mathcal{L}_{\text{TAPS}}^{\text{single}}(\boldsymbol{x}, y, \epsilon) = \max_{\hat{\boldsymbol{z}} \in [\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}]} \ln \Big(1 + \sum_{i \neq y} \exp(f_C(\hat{\boldsymbol{z}})_i - f_C(\hat{\boldsymbol{z}})_y)\Big),$$

where the embedding space bounding box $[\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}]$ is obtained via IBP. However, this loss is not well aligned with adversarial robustness as we show in Figure 4, where only points in the lower-left quadrant are classified correctly (*i.e.*, $o_i^\Delta := o_i - o_y < 0$). We compute the latent adversarial example $\hat{\boldsymbol{z}}$ by conducting a standard adversarial attack on the Cross-Entropy loss over the reachable set 🟩 (optimally for illustration purposes) and observe that the corresponding output $\boldsymbol{f}(\hat{\boldsymbol{z}})$ (×) is classified correctly. However, if we instead use the logit differences o_1^Δ and o_2^Δ as attack objectives, we obtain two misclassified points (×). Combining their dimension-wise worst-case bounds (⋯), we obtain the point •, which realizes the maximum loss over an optimal box approximation of the reachable set and (when computed exactly) directly corresponds to true robustness. We, thus, propose the *multi-estimator* PGD variant of $\mathcal{L}_{\text{TAPS}}$, which estimates the upper bounds on the logit differences o_i^Δ using separate samples and then computes the loss function using the per-dimension worst-cases as:

$$\mathcal{L}_{\text{TAPS}}(\boldsymbol{x}, y, \epsilon) = \ln \Big(1 + \sum_{i \neq y} \exp\Big(\max_{\hat{\boldsymbol{z}} \in [\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}]} f_C(\hat{\boldsymbol{z}})_i - f_C(\hat{\boldsymbol{z}})_y\Big)\Big).$$

Table 1: Comparison of natural (Nat.) and certified (Cert.) accuracy on multiple datasets.

Dataset	ϵ_∞	Method	Source	Nat. [%]	Cert. [%]
MNIST	0.1	COLT	Balunovic & Vechev (2020)	99.2	97.1
		IBP	Shi et al. (2021)	98.84	97.95
		SortNet	Zhang et al. (2022b)	99.01	98.14
		SABR	Müller et al. (2022b)	**99.23**	98.22
		TAPS	this work	99.19	**98.39**
		STAPS	this work	99.15	98.37
	0.3	COLT	Balunovic & Vechev (2020)	97.3	85.7
		IBP	Shi et al. (2021)	97.67	93.10
		SortNet	Zhang et al. (2022b)	98.46	93.40
		SABR	Müller et al. (2022b)	**98.75**	93.40
		TAPS	this work	97.94	**93.62**
		STAPS	this work	98.53	93.51
CIFAR-10	$\frac{2}{255}$	COLT	Balunovic & Vechev (2020)	78.4	60.5
		IBP	Shi et al. (2021)	66.84	52.85
		SortNet	Zhang et al. (2022b)	67.72	56.94
		IBP-R	Palma et al. (2022)	78.19	61.97
		SABR	Müller et al. (2022b)	79.24	62.84
		TAPS	this work	75.09	61.56
		STAPS	this work	**79.76**	**62.98**
	$\frac{8}{255}$	COLT	Balunovic & Vechev (2020)	51.7	27.5
		IBP	Shi et al. (2021)	48.94	34.97
		SortNet	Zhang et al. (2022b)	**54.84**	**40.39**
		IBP-R	Palma et al. (2022)	51.43	27.87
		SABR	Müller et al. (2022b)	52.38	35.13
		TAPS	this work	49.76	35.10
		STAPS	this work	52.82	34.65
TinyImageNet	$\frac{1}{255}$	IBP	Shi et al. (2021)	25.92	17.87
		SortNet	Zhang et al. (2022b)	25.69	18.18
		SABR	Müller et al. (2022b)	28.85	20.46
		TAPS	this work	28.34	20.82
		STAPS	this work	**28.98**	**22.16**



(a) $\epsilon = 2/255$ (b) $\epsilon = 8/255$

Figure 5: Effect of split location on the standard and robust accuracy of TAPS trained networks, depending on the perturbation magnitude ϵ for different certification methods for CIFAR-10. 0 ReLUs in the classifier recovers IBP training.

4. Experimental Evaluation

We now evaluate TAPS as well as STAPS – a version of TAPS where the IBP parts are replaced with the state-of-the-art method SABR, for details see App. B. We provide details on the experimental setup in App. D.

In Table 1, we compare TAPS to state-of-the-art certified training methods. Most closely related are IBP, recovered by TAPS if the classifier size is zero, and COLT, which also combines bound propagation with adversarial attacks but does not allow for joint training. TAPS dominates IBP, improving on its certified and natural accuracy in all settings and demonstrating the importance of avoiding over-regularization. Compared to COLT, TAPS improves certified accuracies significantly (sometimes at cost of slightly reduced natural accuracy), highlighting the importance of joint optimization. Compared to the recent SABR and IBP-R, TAPS often achieves higher certified accuracies at the cost of slightly reduced natural accuracies. Reducing regularization more uniformly with STAPS achieves higher certified accuracies in almost all settings and better natural accuracies in many, highlighting the orthogonality of TAPS and SABR. Most notably, STAPS increases certified accuracy on TinyImageNet by almost 10% while also improving natural accuracy. SortNet, a generalization of a range of recent architectures (Zhang et al., 2021; 2022a; Anil et al., 2019), introducing novel activation functions tailored to yield networks with high ℓ_∞-robustness, performs well on CIFAR-10 at $\epsilon = 8/255$, but is dominated by STAPS in every other setting.

Ablation: Split Location TAPS splits a given network into a feature extractor and classifier. To analyze the effect of this, we train multiple `CNN7`s such that we obtain classifiers with between 0 and 6 (all) ReLU layers and illustrate the resulting standard, adversarial, and certified (using different methods) accuracies in Figure 5 for CIFAR-10, Table 10 for MNIST, and Table 13 for TinyImageNet.

For small perturbations ($\epsilon = 2/255$), increasing classifier size and thus decreasing regularization yields increasing natural and adversarial accuracy. While the precise verification methods can translate this to rising certified accuracies up to large classifier sizes, regularization quickly becomes insufficient for the less precise IBP and CROWN-IBP certification. For larger perturbations ($\epsilon = 8/255$), we observe an initial increase of all accuracies with classifier size, followed by a sudden drop and slow recovery. We hypothesize that this effect is due to the IBP regularization starting to dominate optimization combined with increased training difficulty (see App. E for details). For both perturbation magnitudes, gains in certified accuracy can only be realized with the precise MN-BaB certification (Müller et al., 2022b), highlighting the importance of recent developments in neural network verification for certified training.

We further investigate TAPS's worst-case loss approximation precision, the impact of single- vs. multi-estimator PGD, and the effect of our IBP-based regularization in App. E. For a detailed discussion of how TAPS relates to other works, see App. F.

5. Conclusion

We propose TAPS, a novel certified training method that reduces over-regularization by combining IBP and PGD training to obtain a precise worst-case loss. Crucially, TAPS enables joint training over the IBP and PGD approximated components by introducing the gradient connector. Empirically, we confirm that TAPS yields much more precise approximations of the worst-case loss than existing methods and demonstrate that this translates to state-of-the-art performance in certified training in many settings.

References

Anil, C., Lucas, J., and Grosse, R. B. Sorting out lipschitz function approximation. In *Proc. of International Conference on Machine Learning (ICML)*, volume 97, 2019.

Balunovic, M. and Vechev, M. T. Adversarial training and provable defenses: Bridging the gap. In *Proc. of International Conference on Learning Representations (ICLR)*, 2020.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, volume 8190, 2013. doi: 10.1007/978-3-642-40994-3_25.

Brix, C., Müller, M. N., Bak, S., Johnson, T. T., and Liu, C. First three years of the international verification of neural networks competition (VNN-COMP). *CoRR*, abs/2301.05815, 2023. doi: 10.48550/arXiv.2301.05815. URL https://doi.org/10.48550/arXiv.2301.05815.

Bunel, R., Lu, J., Turkaslan, I., Torr, P. H. S., Kohli, P., and Kumar, M. P. Branch and bound for piecewise linear neural network verification. *J. Mach. Learn. Res.*, 21, 2020.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In *Proc. of International Conference on Machine Learning (ICML)*, volume 97, 2019.

Dathathri, S., Dvijotham, K., Kurakin, A., Raghunathan, A., Uesato, J., Bunel, R., Shankar, S., Steinhardt, J., Goodfellow, I. J., Liang, P., and Kohli, P. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Ferrari, C., Müller, M. N., Jovanovic, N., and Vechev, M. T. Complete verification via multi-neuron relaxation guided branch-and-bound. In *ICLR*, 2022.

Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C., and Vechev, M. T. DL2: training and querying neural networks with logic. In *Proc. of International Conference on Machine Learning (ICML)*, volume 97, 2019.

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

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Le, Y. and Yang, X. S. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7), 2015.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Lécuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *Symposium on Security and Privacy (SSP)*, 2019. doi: 10.1109/SP.2019.00044.

Li, B., Chen, C., Wang, W., and Carin, L. Certified adversarial robustness with additive noise. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *Proc. of International Conference on Learning Representations (ICLR)*, 2018.

Mirman, M., Gehr, T., and Vechev, M. T. Differentiable abstract interpretation for provably robust neural networks. In *Proc. of International Conference on Machine Learning (ICML)*, volume 80, 2018.

Müller, M. N., Brix, C., Bak, S., Liu, C., and Johnson, T. T. The third international verification of neural networks competition (VNN-COMP 2022): Summary and results. *CoRR*, abs/2212.10376, 2022a. doi: 10.48550/arXiv.2212.10376. URL https://doi.org/10.48550/arXiv.2212.10376.

Müller, M. N., Eckert, F., Fischer, M., and Vechev, M. T. Certified training: Small boxes are all you need. *CoRR*, abs/2210.04871, 2022b.

Müller, M. N., Makarchuk, G., Singh, G., Püschel, M., and Vechev, M. T. PRIMA: general and precise neural network certification via scalable convex hull approximations. *Proc. ACM Program. Lang.*, 6(POPL), 2022c. doi: 10.1145/3498704.

Palma, A. D., Behl, H. S., Bunel, R. R., Torr, P. H. S., and Kumar, M. P. Scaling the convex barrier with active sets. In *Proc. of International Conference on Learning Representations (ICLR)*, 2021.

Palma, A. D., Bunel, R., Dvijotham, K., Kumar, M. P., and Stanforth, R. IBP regularization for verified adversarial robustness via branch-and-bound. *ArXiv preprint*, abs/2206.14772, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. In *Proc. of International Conference on Learning Representations (ICLR)*, 2018.

Shi, Z., Wang, Y., Zhang, H., Yi, J., and Hsieh, C. Fast certified robust training with short warmup. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. T. Fast and effective robustness certification. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Singh, G., Ganvir, R., Püschel, M., and Vechev, M. T. Beyond the single neuron convex barrier for neural network certification. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15072–15083, 2019a. URL https://proceedings.neurips.cc/paper/2019/hash/0a9fdbb17feb6ccb7ec405cfb85222c4-Abstract.html.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. T. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL), 2019b.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In *Proc. of International Conference on Learning Representations (ICLR)*, 2014.

Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *Proc. of International Conference on Learning Representations (ICLR)*, 2019.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proc. of International Conference on Machine Learning (ICML)*, volume 80, 2018.

Wong, E., Schmidt, F. R., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Zhang, B., Cai, T., Lu, Z., He, D., and Wang, L. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons. In *Proc. of International Conference on Machine Learning (ICML)*, volume 139, 2021.

Zhang, B., Jiang, D., He, D., and Wang, L. Boosting the certified robustness of l-infinity distance nets. In *Proc. of International Conference on Learning Representations (ICLR)*, 2022a.

Zhang, B., Jiang, D., He, D., and Wang, L. Rethinking lipschitz neural networks and certified robustness: A boolean function perspective. *CoRR*, abs/2210.01787, 2022b. doi: 10.48550/arXiv.2210.01787.

Zhang, H., Weng, T., Chen, P., Hsieh, C., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D. S., and Hsieh, C. Towards stable and efficient training of verifiably robust neural networks. In *Proc. of International Conference on Learning Representations (ICLR)*, 2020.

Zhang, H., Wang, S., Xu, K., Li, L., Li, B., Jana, S., Hsieh, C., and Kolter, J. Z. General cutting planes for bound-propagation-based neural network verification. *ArXiv preprint*, abs/2208.05740, 2022c.

A. Training Objective & Regularization

While complete certification methods can decide any robustness property, this requires exponential time. Therefore, networks should not only be regularized to be robust but also certifiable. Thus, we propose to combine the IBP loss for easy-to-learn and certify samples with the TAPS loss for harder samples as follows:

$$\mathcal{L}(\boldsymbol{x}, y, \epsilon) = \mathcal{L}_{\text{TAPS}}(\boldsymbol{x}, y, \epsilon) \cdot \mathcal{L}_{\text{IBP}}(\boldsymbol{x}, y, \epsilon).$$

This expresses that every sample should be either certifiable with TAPS or IBP bounds[1]. Further, as by construction $\mathcal{L}_{\text{TAPS}} \leq \mathcal{L}_{\text{IBP}}$, we add a scaling term α to the loss gradient:

$$\frac{d\mathcal{L}}{d\boldsymbol{\theta}} := 2\alpha \frac{d\mathcal{L}_{\text{TAPS}}}{d\boldsymbol{\theta}} \cdot \mathcal{L}_{\text{IBP}} + (2 - 2\alpha) \frac{d\mathcal{L}_{\text{IBP}}}{d\boldsymbol{\theta}} \cdot \mathcal{L}_{\text{TAPS}}.$$

Here, $\alpha = 0.5$ recovers the standard gradient, obtained via the product rule (both sides weighted with 1), while $\alpha = 0$ and $\alpha = 1$ correspond to using only the (weighted) IBP and TAPS gradients, respectively. Henceforth, we express this as the regularization weight $w_{\text{TAPS}} = \frac{\alpha}{1-\alpha}$, which intuitively expresses the weight put on TAPS, using $w_{\text{TAPS}} = 5$ unless specified otherwise. Lastly, we reduce the variance of $\mathcal{L}$ by averaging $\mathcal{L}_{\text{IBP}}$ and $\mathcal{L}_{\text{TAPS}}$ over a mini batch before multiplying (see App. C).

B. STAPS – Balancing Regularization by Combining TAPS with SABR

Similar to TAPS, SABR (Müller et al., 2022b) reduces the over-regularization of certified training by optimizing a more precise but unsound approximation of the worst-case loss. However, while SABR's approach of propagating a small Box through the whole network significantly reduces regularization in early layers, the exponential growth of Box abstractions still causes a strong regularization of later layers. In contrast, TAPS's approach of propagating the full input region through the first part of the network (the feature extractor) before using PGD for the remainder reduces regularization only in later layers. Thus, we propose STAPS by replacing the IBP component of TAPS with SABR to obtain a more uniform reduction of over-regularization throughout the whole network.

C. Averaging Multipliers Makes Gradients Efficient

Theorem 1. *Let x_i be i.i.d. drawn from the dataset and define $f_i = f_\theta(x_i)$ and $g_i = g_\theta(x_i)$, where f_θ and g_θ are two functions. Further, define $L_1 = (\sum_{i=1}^n \frac{1}{n} f_i) \cdot (\sum_{i=1}^n \frac{1}{n} g_i)$ and $L_2 = \sum_{i=1}^n \frac{1}{n} f_i g_i$. Then, assuming the function value and the gradient are independent, $\mathbb{E}_x\left(\frac{\partial L_1}{\partial \theta}\right) = \mathbb{E}_x\left(\frac{\partial L_2}{\partial \theta}\right)$ and $\mathrm{Var}_x\left(\frac{\partial L_1}{\partial \theta}\right) \leq \mathrm{Var}_x\left(\frac{\partial L_2}{\partial \theta}\right)$.*

Proof. A famous result in stochastic optimization is that stochastic gradients are unbiased. For completeness, we give a short proof of this property: Let $L = \mathbb{E}_x f(x) = \int_{-\infty}^{+\infty} f(x)dP(x)$, thus $\nabla_x L = \nabla_x(\int_{-\infty}^{+\infty} f(x)dP(x)) = \int_{-\infty}^{+\infty} \nabla_x f(x)dP(x) = \mathbb{E}_x(\nabla_x f(x))$. Therefore, $\nabla f(x_i)$ is an unbiased estimator of the true gradient.

Applying that the stochastic gradients are unbiased, we can write $\nabla_\theta f_i = \nabla_\theta f + \eta_i$, where $\nabla_\theta f$ is the expectation of the gradient and η_i is the deviation such that $\mathbb{E}\eta_i = 0$ and $\mathrm{Var}(\eta_i) = \sigma_1^2$. Since x_i is drawn independently, f_i are independent and thus η_i are independent. Similarly, we can write $\nabla_\theta g_i = \nabla_\theta g + \delta_i$, where $\mathbb{E}\delta_i = 0$ and $\mathrm{Var}(\delta_i) = \sigma_2^2$. η_i and δ_i may be dependent.

Define $\bar{f} = \sum_i \frac{1}{n} f_i$ and $\bar{g} = \sum_i \frac{1}{n} g_i$. Explicit computation gives us that $\nabla L_1 = \bar{g}\left(\sum_i \frac{1}{n}\nabla f_i\right) + \bar{f}\left(\sum_i \frac{1}{n}\nabla g_i\right)$, and $\nabla L_2 = \sum_i \frac{1}{n}\left(f_i \nabla g_i + g_i \nabla f_i\right)$. Therefore,

$$\mathbb{E}_x\left(\nabla_\theta L_1 \mid f_i, g_i\right) = \bar{g}\nabla_\theta f + \bar{f}\nabla_\theta g = \mathbb{E}_x\left(\nabla_\theta L_2 \mid f_i, g_i\right).$$

By the law of total probability,

$$\begin{aligned}\mathbb{E}_x\left(\nabla_\theta L_1\right) &= \mathbb{E}_{f_i,g_i}\left(\mathbb{E}_x\left(\nabla_\theta L_1 \mid f_i, g_i\right)\right)\\ &= \mathbb{E}_{f_i,g_i}\left(\mathbb{E}_x\left(\nabla_\theta L_2 \mid f_i, g_i\right)\right)\\ &= \mathbb{E}_x\left(\nabla_\theta L_2\right).\end{aligned}$$

Therefore, we have got the first result: the gradients of L_1 and L_2 have the same expectation.

To prove the variance inequality, we will use variance decomposition formula[2]:

$$\begin{aligned}\mathrm{Var}_x(\nabla_\theta L_k) = \mathbb{E}_{f_i,g_i}(\mathrm{Var}_x(\nabla_\theta L_k \mid f_i, g_i)) +\\ \mathrm{Var}_{f_i,g_i}(\mathbb{E}_x(\nabla_\theta L_k \mid f_i, g_i)),\end{aligned}$$

$k = 1, 2$. We have proved that $\mathbb{E}_x(\nabla_\theta L_1 \mid f_i, g_i) = \mathbb{E}_x(\nabla_\theta L_2 \mid f_i, g_i)$, thus the second term is equal. Next, we prove that $\mathrm{Var}_x(\nabla_\theta L_1 \mid f_i, g_i) \leq \mathrm{Var}_x(\nabla_\theta L_2 \mid f_i, g_i)$, which implies $\mathrm{Var}_x(\nabla_\theta L_1) \leq \mathrm{Var}_x(\nabla_\theta L_2)$.

By explicit computation, we have

$$\begin{aligned}&\mathrm{Var}(\nabla L_1 \mid f_i, g_i)\\ &= (\bar{g})^2 \mathrm{Var}\left(\sum_i \frac{1}{n}\eta_i\right) + (\bar{f})^2 \mathrm{Var}\left(\sum_i \frac{1}{n}\delta_i\right)\\ &= \frac{1}{n}\sigma_1^2 (\bar{g})^2 + \frac{1}{n}\sigma_2^2 (\bar{f})^2, \end{aligned} \tag{2}$$

and

$$\begin{aligned}&\mathrm{Var}(\nabla L_2 \mid f_i, g_i)\\ &= \mathrm{Var}\left(\sum_i \frac{1}{n} f_i \delta_i\right) + \mathrm{Var}\left(\sum_i \frac{1}{n} g_i \eta_i\right)\\ &= \frac{1}{n}\sigma_1^2 \left(\sum_i \frac{1}{n} g_i^2\right) + \frac{1}{n}\sigma_2^2 \left(\sum_i \frac{1}{n} f_i^2\right). \end{aligned} \tag{3}$$

[1] See Fischer et al. (2019) for further discussion.

[2] https://en.wikipedia.org/wiki/Law_of_total_variance

Applying Jensen's formula on the convex function x^2, we have $\left(\sum_i \frac{1}{n} a_i\right)^2 \leq \sum_i \frac{1}{n} a_i^2$ for any a_i, thus $(\bar{f})^2 \leq \sum_i \frac{1}{n} f_i^2$ and $(\bar{g})^2 \leq \sum_i \frac{1}{n} g_i^2$. Combining Eq. (2) and Eq. (3) with these two inequalities gives the desired result. $\square$

D. Experimental Details

We implement TAPS in PyTorch (Paszke et al., 2019) and use MN-BaB (Ferrari et al., 2022) for certification. We conduct experiments on MNIST (LeCun et al., 2010), CIFAR-10 (Krizhevsky et al., 2009), and TinyImageNet (Le & Yang, 2015) using ℓ_∞ perturbations and the `CNN7` architecture (Gowal et al., 2018).

Approximation Precision To evaluate whether TAPS yields more precise approximations of the worst-case loss than other certified training methods, we compute approximations of the maximum margin loss with IBP, PGD (50 steps 3 restarts), SABR ($\lambda = 0.4$), and TAPS on small `CNN3` trained with IBP, SABR, and TAPS for all MNIST test set samples. We report histograms over the difference to the exact worst-case loss computed with a MILP encoding (Tjeng et al., 2019) in Figure 6. Positive values correspond to over-approximations while negative values correspond to under-approximation. We observe that regardless of the training method, the TAPS approximation is by far the most precise, achieving the smallest mean and mean absolute error as well as variance.

D.1. TAPS Training Procedure

Algorithm 1 Train Loss Computation

```
Input: data $X_B = \{(\boldsymbol{x}_b, y_b)\}_b$, current $\epsilon$, target $\epsilon^t$, network $\boldsymbol{f}$
Output: A differentiable loss $L$
$\mathcal{L}_{\text{IBP}} = \sum_{b \in \mathcal{B}} \mathcal{L}_{\text{IBP}}(\boldsymbol{x}_b, y_b, \epsilon)/|\mathcal{B}|$.
if $\epsilon < \epsilon^t$ then
    // $\epsilon$ annealing regularization from Shi et al. (2021)
    $\mathcal{L}_{\text{fast}} = \lambda \cdot (\mathcal{L}_{\text{tightness}} + \mathcal{L}_{\text{relu}})$
    return $\mathcal{L}_{\text{IBP}} + \epsilon/\epsilon^t \cdot \mathcal{L}_{\text{fast}}$
end if
$\mathcal{L}_{\text{TAPS}} = \sum_{b \in \mathcal{B}} L_{\text{TAPS}}(\boldsymbol{x}_b, y_b, \epsilon)/|\mathcal{B}|$.
return $\mathcal{L}_{\text{IBP}} \cdot \mathcal{L}_{\text{TAPS}}$
```

To obtain state-of-the-art performance with IBP, various training techniques have been developed. We use two of them: ϵ-annealing (Gowal et al., 2018) and initialization and regularization for stable box sizes (Shi et al., 2021). ϵ-annealing slowly increases the perturbation magnitude ϵ during training to avoid exploding approximation sizes and thus gradients. The initialization of Shi et al. (2021) scales network weights to achieve constant box sizes over network depth. During the ϵ-annealing phase, we combine the IBP loss with the ReLU stability regularization $\mathcal{L}_{\text{fast}}$ (Shi et al., 2021), before switching to the TAPS loss as described in App. A. We formalize this in Algorithm 1.

Table 2: The training epoch and learning rate settings.

Dataset	Batch size	Total epochs	Annealing epochs	Decay-1	Decay-2
MNIST	256	70	20	50	60
CIFAR-10	128	160	80	120	140
TinyImageNet	128	80	20	60	70

D.2. Datasets and Augmentation

We use the MNIST (LeCun et al., 2010), CIFAR-10 (Krizhevsky et al., 2009), and TinyImageNet (Le & Yang, 2015) datasets, all of which are freely available with no license specified.

The data preprocessing mostly follows Müller et al. (2022b). For MNIST, we do not apply any preprocessing. For CIFAR-10 and TinyImageNet, we normalize with the dataset mean and standard deviation (after calculating perturbation size) and augment with random horizontal flips. For CIFAR-10, we apply random cropping to 32×32 after applying a 2 pixel padding at every margin. For TinyImageNet, we apply random cropping to 56×56 during training and center cropping during testing.

D.3. Model Architectures

Unless specified otherwise, we follow Shi et al. (2021); Müller et al. (2022b) and use a `CNN7` with Batch Norm for our main experiments. `CNN7` is a convolutional network with 7 convolutional and linear layers. All but the last linear layer are followed by a Batch Norm and ReLU layer.

D.4. Training Details

We follow the hyperparameter choices of Shi et al. (2021) for ϵ-annealing, learning rate schedules, batch sizes, and gradient clipping (see Table 2). We set the initial learning rate to 0.0005 and decrease it by a factor of 0.2 at Decay-1 and -2. We set the gradient clipping threshold to 10.

We use additional L_1 regularization in some settings where we observe signs of overfitting. We report the L_1 regularization and split position chosen for different settings in Table 3 and Table 5.

We train using single NVIDIA GeForce RTX 3090 for MNIST and CIFAR-10 and single NVIDIA TITAN RTX for TinyImageNet. Training and certification time are reported in Table 4 and Table 6.

Table 3: Hyperparameter for the best classifier splits for TAPS.

Dataset	ϵ	# ReLUs in Classifier	L_1	w
MNIST	0.1	3	1e-6	5
	0.3	1	0	5
CIFAR-10	2/255	5	2e-6	5
	8/255	1	2e-6	5
TinyImageNet	1/255	1	0	5

Table 4: The training and certification time cost for the best classifier splits for TAPS.

Dataset	ϵ	Train Time (s)	Certify Time (s)
MNIST	0.1	42 622	17 117
	0.3	12 417	41 624
CIFAR-10	2/255	141 281	166 474
	8/255	27 017	26 968
TinyImageNet	1/255	306 036	23 497

Table 5: Hyperparameter with the best classifier splits for STAPS.

Dataset	ϵ	# ReLUs in Classifier	L_1	w
MNIST	0.1	1	2e-5	5
	0.3	1	2e-6	5
CIFAR-10	2/255	1	2e-6	2
	8/255	1	2e-6	5
TinyImageNet	1/255	2	1e-6	5

Table 6: The training and certification time cost with the best classifier splits for STAPS.

Dataset	ϵ	Train Time (s)	Certify Time (s)
MNIST	0.1	19 865	12 943
	0.3	23 613	125 768
CIFAR-10	2/255	47 631	398 245
	8/255	48 706	77 793
TinyImageNet	1/255	861 639	35 183

Table 7: Effect of IBP regularization and the TAPS gradient expanding coefficient α for MNIST $\epsilon = 0.3$.

w_{TAPS}	Avg time (s)	Nat (%)	Adv. (%)	Cert. (%)
$\mathcal{L}_{\text{IBP}}$	2.3	97.6	93.37	93.15
0	2.7	97.37	93.32	93.06
1	4.5	97.86	93.80	93.36
5	6.9	98.16	94.18	**93.55**
10	15.7	98.25	94.43	93.02
$15^\dagger$	42.8	98.53	**95.00**	91.55
$20^\dagger$	73.7	**98.75**	94.33	82.67
$\infty^\dagger$	569.7	98.0	94.00	45.00
$\mathcal{L}_{\text{TAPS}}{}^\dagger$	817.1	98.5	94.50	17.50

† Only evaluated on part of the test set within a 2-day time limit.

D.5. Certification Details

We combine IBP (Gowal et al., 2018), CROWN-IBP (Zhang et al., 2020), and MN-BaB (Ferrari et al., 2022) for certification, running the most precise but also computationally costly MN-BaB only on samples not certified by the other methods. We use the same configuration for MN-BaB as Müller et al. (2022b). The certification is run on a single NVIDIA TITAN RTX.

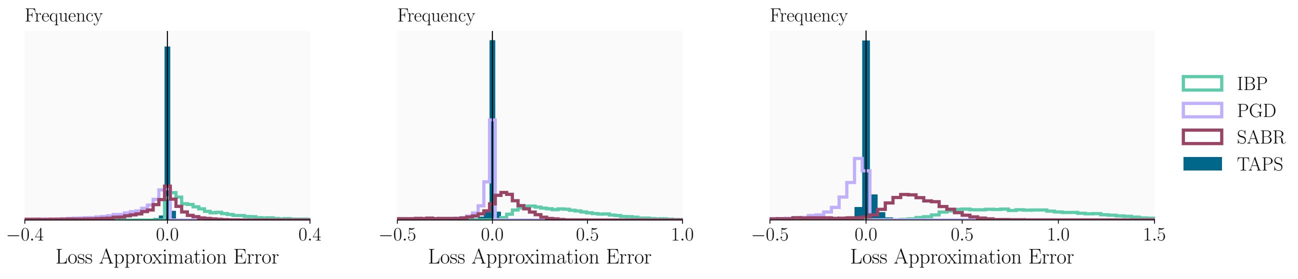
MN-BaB (Ferrari et al., 2022) is a state-of-the-art (Brix et al., 2023; Müller et al., 2022a) neural network verifier, combining the branch-and-bound paradigm (Bunel et al., 2020) with precise multi-neuron constraints (Müller et al., 2022c; Singh et al., 2019a).

We use a mixture of strong adversarial attacks to evaluate adversarial accuracy. First, we run PGD attacks with 5 restarts and 200 iterations each. Then, we run MN-BaB to search for adversarial examples with a timeout of 1000 seconds.

E. Extended Evaluation

In this section, we further investigate TAPS's worst-case loss approximation precision, the impact of single- vs. multi-estimator PGD, and the effect of our IBP-based regularization.

IBP Regularization To analyze the effectiveness of the multiplicative IBP regularization discussed in App. A, we train with IBP in isolation ($\mathcal{L}_{\text{IBP}}$), IBP with TAPS weighted gradients ($w_{\text{TAPS}} = 0$), varying levels of gradient scaling for the TAPS component ($w_{\text{TAPS}} \in [1, 20]$), TAPS with IBP weighting ($w_{\text{TAPS}} = \infty$), and TAPS loss in isolation, reporting results in Table 7. We observe that IBP in isolation yields comparatively low standard but moderate certified accuracies with fast certification times. Increasing the weight w_{TAPS} of the TAPS gradients reduces regularization, leading to longer certification times. Initially, these



(a) IBP-trained (b) SABR-trained (c) TAPS-trained

Figure 6: Distribution of the worst-case loss approximation errors over test set samples, depending on the training and bounding method. Positive values correspond to over-approximations and negative values to under-approximations. We use an exact MILP encoding (Tjeng et al., 2019) as reference.

also translate to higher adversarial and certified accuracies, peaking at $w_{\text{TAPS}} = 15$ and $w_{\text{TAPS}} = 5$, respectively, before especially certified accuracy decreases as regularization becomes insufficient for certification. We confirm these trends for TinyImageNet in Table 12 in App. E.

Table 8: Comparison of single- and multi-estimator PGD, depending on the split position for MNIST at $\epsilon = 0.3$.

# ReLU in Classifier	Single		Multi	
	Certified	Natural	Certified	Natural
1	-[†]	31.47[†]	**93.62**	97.94
3	92.91	98.56	93.03	98.63
6	92.41	**98.88**	92.70	**98.88**

[†] Training encounters mode collapse. Last epoch performance reported.

Single-Estimator vs Multi-Estimator PGD To evaluate the importance of our multi-estimator PGD variant, we compare it to single-estimator PGD across a range of split positions, reporting results in Table 8. We observe that across all split positions, multi-estimator PGD achieves better certified and better or equal natural accuracy. Further, training collapses reproducibly for single-estimator PGD for small classifiers, indicating that multi-estimator PGD additionally improves training stability.

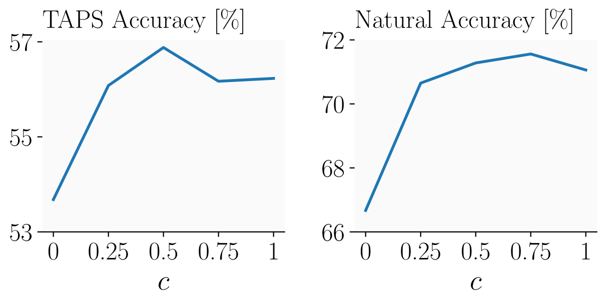


Figure 7: Effect of the gradient connector on TAPS (left) and natural (right) accuracy.

Table 9: Comparison of TAPS accuracy with certified and adversarial accuracy.

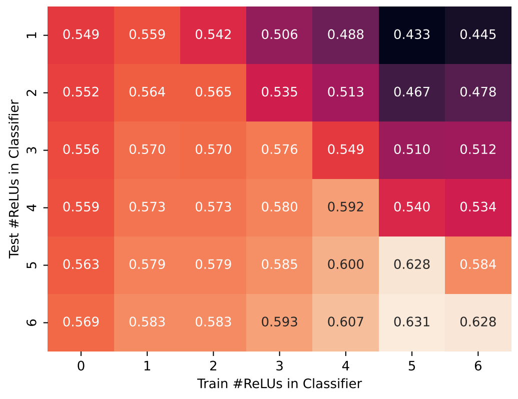
Dataset	cor(TAPS, cert.)	cor(TAPS, adv.)	TAPS − cert.	TAPS − adv.
MNIST	0.9139	0.9633	0.0122 ± 0.0141	0.0033 ± 0.0079
CIFAR-10	0.9973	0.9989	0.0028 ± 0.0095	-0.0040 ± 0.0077

Gradient Connector In Figure 7, we illustrate the effect of our gradient connector's parameterization c (see Section 3). We report TAPS accuracy (the portion of samples where all latent adversarial examples are classified correctly) as a proxy for the goodness of fit. Recall that $c = 0$ corresponds to the binary connector and $c = 1$ to the linear connector. We observe that the binary connector achieves poor TAPS and natural accuracy, indicating a less well-behaved optimization problem. TAPS accuracy peaks at $c = 0.5$, indicating high goodness-of-fit and thus a well-behaved optimization problem. This agrees well with our theoretical considerations aiming to avoid sparsity ($c < 0.5$) and contradicting gradients ($c > 0.5$).

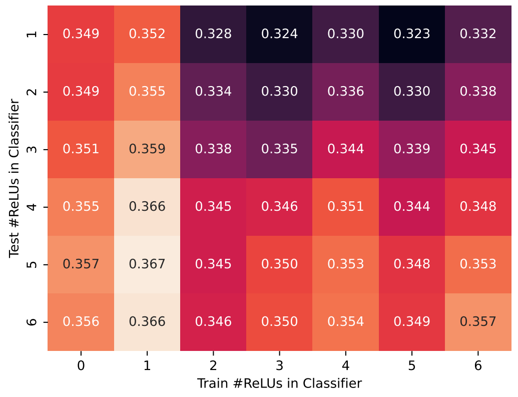
TAPS Accuracy as GoF In practice, we want to avoid certifying every model with expensive certification methods, especially during hyperparameter tuning and applying early stopping. Therefore, we need a criterion to select models. In this section, we aim to show that TAPS accuracy is a good proxy for goodness of fit (GoF).

We compare the TAPS accuracy to adversarial and certified accuracy with all models we get on MNIST and CIFAR-10. The result is shown in Table 9. From Table 9, we can see that the correlations between TAPS accuracy and both the adversarial and the certified accuracy are close to 1. In addition, the differences are small and centered at zero, with a small standard deviation. Therefore, we conclude that TAPS accuracy is a good estimate of the true robustness, thus a good measurement of GoF. In all the experiments, we perform model selection based on the TAPS accuracy.

Training Difficulty Since TAPS is merely a training technique, we can test TAPS-trained models trained with a new classifier split. By design, if the training is successful, then under a given classifier split for testing, the model trained



$\epsilon = \frac{2}{255}$

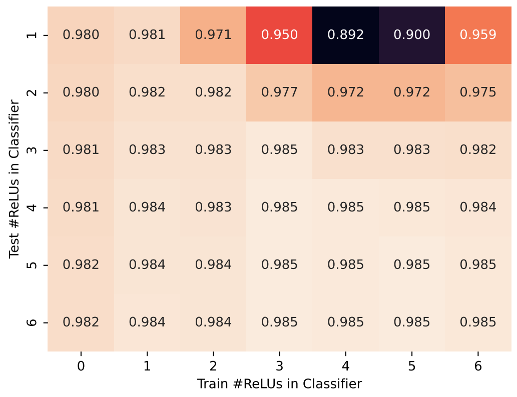


$\epsilon = \frac{8}{255}$

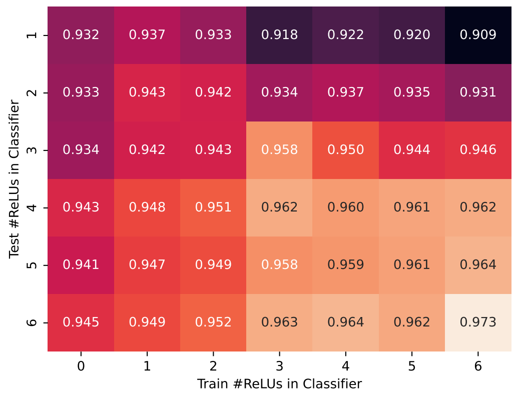
Figure 8: TAPS accuracy of models trained by different classifier sizes for CIFAR-10.

with the same split should have the best TAPS accuracy. Although this is often true, we find that in some cases, a smaller classifier split results in higher TAPS accuracy, indicating the difficulty of training.

Figure 8 shows the tested TAPS accuracy for models trained with IBP and different splits for CIFAR-10. The result on MNIST is provided in Figure 9. From these figures, we can see that for CIFAR-10 $\epsilon = 2/255$ and MNIST, the models trained with the same test split has the highest TAPS accuracy, as expected. However, for CIFAR-10 $\epsilon = 8/255$, the model trained with classifier size 4 is consistently better for all test splits. Furthermore, as we show in Section 4, this model has the best adversarial and certified accuracy as well. This means that in this setting, the training of larger splits is too difficult, such that TAPS is not able to find a good model to minimize the given loss. However, in other settings, TAPS is easy enough to train.



$\epsilon = 0.1$



$\epsilon = 0.3$

Figure 9: TAPS accuracy of models trained by different classifier sizes for MNIST.

Table 10: Effect of split position into the classifier and feature extractor (overall model size remains unchanged). All numbers are in percentages. All results for MNIST.

ϵ	# ReLUs in Classifier	Nat.	Adv.	Cert.	
				MN-BaB	IBP
0.1	0	98.87	98.16	98.13	97.83
	1	99.06	98.37	98.31	96.27
	2	99.16	98.35	98.25	87.82
	3	99.19	**98.51**	**98.39**	62.83
	4	**99.28**	98.47	98.03	4.75
	5	99.22	**98.51**	98.17	9.76
	6	99.09	98.45	98.27	81.89
0.3	0	97.60	93.37	93.15	93.08
	1	97.94	94.01	**93.62**	92.76
	2	98.16	94.18	93.55	91.85
	3	98.63	94.48	93.03	89.40
	4	98.7	94.85	93.44	89.52
	5	98.63	94.64	93.26	89.15
	6	**98.88**	**95.11**	92.70	85.03

Split Position We include the full tables of the experiment in Section 4 in Table 10 and Table 11.

Repeatability Due to the large computational cost of up to 10 GPU-days for some experiments (see Tables 4 and 6), we could not repeat all experiments multiple times to report full statistics. However, we repeated the experiments for MNIST at $\epsilon = 0.1$ and $\epsilon = 0.3$ (see Table 14), and find very small standard deviations, indicating good repeatability of our results.

F. Related Work

Verification Methods In this work, we only consider deterministic verification methods, which analyze a given network as is. While *complete* (or *exact*) methods (Tjeng et al., 2019; Palma et al., 2021; Wang et al., 2021; Müller et al., 2022c; Zhang et al., 2022c; Ferrari et al., 2022) can decide any robustness property given enough time, *incomplete* methods (Singh et al., 2018; Raghunathan et al., 2018; Zhang et al., 2018; Dathathri et al., 2020) sacrifice some precision for better scalability. However, recent complete methods can be used with a timeout to obtain effective incomplete methods.

Certified Training Most certified training methods compute and minimize sound over-approximations of the worst-case loss using different approximation methods: DIFFAI (Mirman et al., 2018) and IBP (Gowal et al., 2018) use BOX approximations, Wong et al. (2018) use DEEPZ relaxations (Singh et al., 2018), Wong & Kolter (2018) back-substitute linear bounds using fixed relaxations, Zhang et al. (2020) use dynamic relaxations (Zhang et al., 2018; Singh et al., 2019b) and compute intermediate bounds using BOX relaxations. Shi et al. (2021) significantly shorten training

Table 11: Effect of split position into the classifier and feature extractor (overall model size remains unchanged). All numbers are in percentages. All results for CIFAR-10.

ϵ	#ReLUs	Nat.	Adv.	Cert.	
				MN-BaB	IBP
$\frac{2}{255}$	0	67.27	56.32	56.14	53.54
	1	70.10	57.78	57.48	41.86
	2	70.74	57.83	57.39	40.24
	3	71.88	58.89	58.23	34.41
	4	72.45	60.38	59.47	31.88
	5	75.09	**63.00**	**61.56**	24.36
	6	**75.40**	62.73	61.11	24.90
$\frac{8}{255}$	0	48.15	34.63	34.60	34.26
	1	49.76	**35.29**	**35.10**	32.92
	2	47.28	33.54	33.12	28.94
	3	48.76	33.50	33.12	29.14
	4	50.19	34.78	34.35	29.14
	5	50.2	34.33	33.72	28.83
	6	**51.03**	35.25	34.44	29.97

Table 12: Effect of IBP regularization and the TAPS gradient expanding coefficient α for TINYIMAGENET $\epsilon = \frac{1}{255}$.

w_{TAPS}	Avg time (s)	Nat. (%)	Adv. (%)	Cert. (%)
$\mathcal{L}_{\text{IBP}}$	0.28	25.00	19.72	19.72
1	1.17	25.83	20.24	20.22
5	2.34	28.34	20.94	20.82
10	4.12	28.23	**21.05**	**20.89**
20	5.94	**28.44**	20.68	20.44

Table 13: Performance of TAPS and STAPS with different number of ReLU layers included in the classifier on ImageNet.

ReLU	TAPS					STAPS				
	Nat. (%)	Adv. (%)	Cert. (%)	Train (s)	Certify (s)	Nat. (%)	Adv. (%)	Cert. (%)	Train (s)	Certify (s)
1	28.34	20.94	20.82	306 036	23 497	28.75	22.25	22.04	350 924	35 894
2	27.02	20.94	20.84	944 520	32 407	28.98	22.40	22.16	861 639	35 183

Table 14: Result statistics for MNIST. Estimated with 3 independent runs.

ϵ	Nat. (%)	Adv. (%)	Cert. (%)
0.1	99.22 ± 0.03	98.45 ± 0.06	98.28 ± 0.10
0.3	97.96 ± 0.04	93.96 ± 0.04	93.57 ± 0.02

schedules by combining IBP training with a special initialization. Some more recent methods instead compute and optimize more precise, but not necessarily sound, worst-case loss approximations: SABR (Müller et al., 2022b) reduce the regularization of IBP training by propagating only small but carefully selected subregions. IBP-R (Palma et al., 2022) combines adversarial training at large perturbation radii with an IBP-based regularization. COLT (Balunovic & Vechev, 2020) is conceptually most similar to TAPS and thus compared to in more detail below.

COLT (Balunovic & Vechev, 2020), similar to TAPS, splits the network into a feature extractor and classifier, computing bounds on the feature extractor's output (using the ZONOTOPE (Singh et al., 2019b) instead of BOX domain) before conducting adversarial training over the resulting region. Crucially, however, COLT lacks a gradient connector and, thus, does not enable gradient flow between the latent adversarial examples and the bounds on the feature extractor's output. Therefore, gradients can only be computed for the weights of the classifier but not the feature extractor, preventing the two components from being trained jointly. Instead, a stagewise training process is used, where the split between feature extractor and classifier gradually moves through the network starting with the whole network being treated as the classifier. This has several repercussions: not only is the training very slow and limited to relatively small networks (a four-layer network takes almost 2 days to train) but more importantly, the feature extractor (and thus the whole network) is never trained to allow precise bound propagation. Instead, only the classifier is trained to become robust to the incurred imprecisions. As this makes bound propagation methods ineffective for certification, Balunovic & Vechev (2020) employ precise but very expensive mixed integer linear programming (MILP (Tjeng et al., 2019)), further limiting the scalability of COLT.

In our experimental evaluation (Section 4), we compare TAPS in detail to the above methods.

Robustness by Construction Li et al. (2019), Lécuyer et al. (2019), and Cohen et al. (2019) construct probabilistic classifiers by introducing randomness into the inference process of a base classifier. This allows them to derive robustness guarantees with high probability at the cost of significant (100x) runtime penalties. Zhang et al. (2021; 2022a) introduce ℓ_∞-distance neurons, generalized to SORTNET by Zhang et al. (2022b) which inherently exhibits ℓ_∞-Lipschitzness properties, yielding good robustness for large perturbation radii, but poor performance for smaller ones.

G. Limitations

TAPS and all other certified training methods can only be applied to mathematically well-defined perturbations of the input such as ℓ_p-balls, while real-world robustness may require significantly more complex perturbation models. Further and similarly to other unsound certified training methods, TAPS introduces a new hyperparameter, the split position, that can be tuned to improve performance further beyond the default choice of 1 ReLU layer in the classifier. Finally, while training with TAPS is similarly computationally expensive as with other recent methods, it is notably more computationally expensive than simple certified training methods such as IBP.

H. Reproducibility

We publish our code, trained models, and detailed instructions on how to reproduce our results at ANONYMIZED. Additionally, we provide detailed descriptions of all hyperparameter choices, data sets, and preprocessing steps in App. D.