# A DEFERRED PROOFS

In this section, we provide the proof for Lemma A.1. Let us first consider the following Lemma:

**Lemma A.1** (Hyper-Box Growth). *Let $y := \sigma(x) = \max(0, x)$ be a ReLU function and consider box inputs with radius $\delta_x$ and centres $\bar{x} \sim \mathcal{D}$. Then the mean radius $\mathbb{E}\delta_y$ of the output boxes will satisfy:*

$$\frac{\partial}{\partial \delta_{x,i}} \mathbb{E}_{\mathcal{D}}[\delta_{y,i}] = \frac{1}{2} P_{\mathcal{D}}[-\delta_{x,i} < \bar{x}_i < \delta_{x,i}] + P_{\mathcal{D}}[\bar{x}_i > \delta_{x,i}] > 0, \tag{9}$$

*and*

$$\frac{\partial}{\partial^2 \delta_{x,i}} \mathbb{E}_{\mathcal{D}}[\delta_{y,i}] = \frac{1}{2}(P_{\mathcal{D}}[\bar{x}_i = -\delta_{x,i}] - P_{\mathcal{D}}[\bar{x}_i = \delta_{x,i}]). \tag{10}$$

*Proof.* Recall that given an input box with centre $\bar{\boldsymbol{x}}$ and radius $\delta_{\boldsymbol{x}}$, the output relaxation of a ReLU layer is defined by:

$$\bar{y}_i = \begin{cases} 0, & \text{if } \bar{x}_i + \delta_{x,i} \leq 0 \\ \frac{\bar{x}_i + \delta_{x,i}}{2}, & \text{elif } \bar{x}_i - \delta_{x,i} \leq 0 \\ \bar{x}_i, & \text{else} \end{cases}, \qquad \delta_{y,i} = \begin{cases} 0, & \text{if } \bar{x}_i + \delta_{x,i} \leq 0 \\ \frac{\bar{x}_i + \delta_{x,i}}{2}, & \text{elif } \bar{x}_i - \delta_{x,i} \leq 0 \\ \delta_{x,i}, & \text{else} \end{cases} \tag{11}$$

We thus obtain the expectation

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}[\delta_{y,i}] &= \int_{-\delta_{x,i}}^{\delta_{x,i}} \frac{\bar{x}_i + \delta_{x,i}}{2} p[\bar{x}_i] d\bar{x}_i + \int_{\delta_{x,i}}^{\infty} \delta_{x,i} p_D(\bar{x}_i) d\bar{x}_i \\ &= \frac{\delta_{x,i}}{2} P_{\mathcal{D}}[-\delta_{x,i} < \bar{x}_i < \delta_{x,i}] + \delta_{x,i} P_{\mathcal{D}}[\bar{x}_i > \delta_{x,i}] + \int_{-\delta_{x,i}}^{\delta_{x,i}} \frac{\bar{x}_i}{2} p[\bar{x}_i] d\bar{x}_i, \end{aligned} \tag{12}$$

its derivative

$$\begin{aligned} \frac{\partial}{\partial \delta_{x,i}} \mathbb{E}_{\mathcal{D}}[\delta_{y,i}] =& \frac{1}{2} P_{\mathcal{D}}[-\delta_{x,i} < \bar{x}_i < \delta_{x,i}] + \frac{\delta_{x,i}}{2}(P_{\mathcal{D}}[\bar{x}_i = -\delta_{x,i}] + P_{\mathcal{D}}[\bar{x}_i = \delta_{x,i}]) \\ &+ P_{\mathcal{D}}[\bar{x}_i > \delta_{x,i}] - \delta_{x,i} P_{\mathcal{D}}[\bar{x}_i = \delta_{x,i}] \\ &+ \frac{\delta_{x,i}}{2}(P_{\mathcal{D}}[\bar{x}_i = -\delta_{x,i}] - P_{\mathcal{D}}[\bar{x}_i = \delta_{x,i}]) \\ =& \frac{1}{2} P_{\mathcal{D}}[-\delta_{x,i} < \bar{x}_i < \delta_{x,i}] + P_{\mathcal{D}}[\bar{x}_i > \delta_{x,i}] > 0, \end{aligned} \tag{13}$$

and its curvature

$$\begin{aligned} \frac{\partial}{\partial^2 \delta_{x,i}} \mathbb{E}_{\mathcal{D}}[\delta_{y,i}] &= \frac{1}{2}(P_{\mathcal{D}}[\bar{x}_i = -\delta_{x,i}] + P_{\mathcal{D}}[\bar{x}_i = \delta_{x,i}]) - P_{\mathcal{D}}[\bar{x}_i = \delta_{x,i}] \\ &= \frac{1}{2}(P_{\mathcal{D}}[\bar{x}_i = -\delta_{x,i}] - P_{\mathcal{D}}[\bar{x}_i = \delta_{x,i}]). \end{aligned} \tag{14}$$

□

Now, we can easily proof Theorem 4.1, restated below for convenience.

**Theorem 4.1** (Hyper-Box Growth). *Let $y := \sigma(x) = \max(0, x)$ be a ReLU function and consider box inputs with radius $\delta_x$ and asymmetrically distributed centres $\bar{x} \sim \mathcal{D}$ such that $P_{\mathcal{D}}(\bar{x} = -z) > P_{\mathcal{D}}(\bar{x} = z),\ \forall z \in \mathbb{R}^{>0}$. Then, the mean output radius $\delta_y$ will grow super-linearly in the input radius $\delta_x$. More formally:*

$$\forall \delta_x, \delta'_x \in \mathbb{R}^{\geq 0}: \quad \delta'_x > \delta_x \implies \mathbb{E}_{\mathcal{D}}[\delta'_y] > \mathbb{E}_{\mathcal{D}}[\delta_y] + (\delta'_x - \delta_x) \frac{\partial}{\partial \delta_x} \mathbb{E}_{\mathcal{D}}[\delta_y]. \tag{8}$$

*Proof.* We apply Lemma A.1 by substituting an asymmetric centre distribution $\mathcal{D}$, satisfying $P_{\mathcal{D}}(\bar{x} = -z) > P_{\mathcal{D}}(\bar{x} = z),\ \forall z \in \mathbb{R}^{>0}$ into Eq. (10) to obtain:

$$\frac{\partial}{\partial^2 \delta_{x,i}} \mathbb{E}_{\mathcal{D}}[\delta_{y,i}] = \frac{1}{2}(P_{\mathcal{D}}[\bar{x}_i = -\delta_{x,i}] - P_{\mathcal{D}}[\bar{x}_i = \delta_{x,i}]) > 0.$$

The theorem follows trivially from the strictly positive curvature. □

**Example for Piecewise Uniform Distribution** Let us assume the centres $\bar{x} \sim D$ are distributed according to:

$$P_{\mathcal{D}}[\bar{x} = z] = \begin{cases} a, & \text{if } -l \leq z < 0 \\ b, & \text{elif } 0 < u \leq l \\ 0, & \text{else} \end{cases}, \quad l = \frac{1}{a+b}, \tag{15}$$

where $a$ and $b$. Then we have by Lemma A.1

$$\mathbb{E}_{\mathcal{D}}[\delta_y] = \frac{\delta_x}{2} P_{\mathcal{D}}[-\delta_x < \bar{x} < \delta_x] + \delta_x P_{\mathcal{D}}[\bar{x} > \delta_x] + \int_{-\delta_x}^{\delta_x} \frac{\bar{x}}{2} p[\bar{x}] d\bar{x} \tag{16}$$

$$= \frac{\delta_x^2}{2}(a+b) + b\delta_x(l - \delta_x) + \frac{\delta_x^2}{4}(b-a) \tag{17}$$

$$= \delta_x^2 \frac{a-b}{4} + \delta_x \frac{b}{a+b}. \tag{18}$$

We observe quadratic growth for $a > b$ and recover the symmetric special case of $\mathbb{E}_{\mathcal{D}}[\delta_y] = 0.5\delta_x$ for $a = b$.

# B Additional Theoretical Details

## B.1 Cross-Entropy Loss Formulation

Below we derive the formulation of the Cross-Entropy (CE) loss used in equation Eqs. (2) and (5). We let $p_i$ be the label probability of class $i$, $q_i$ the predicted probability of class $i$, $t$ the label for sample $\boldsymbol{x}$ and $\boldsymbol{y}$ the logits predicted by a neural network $\boldsymbol{h}$ for this sample.

$$\begin{aligned}
\mathcal{L}_{\text{CE}} &= -\sum_{i=1}^{n} p_i(y) \log(q_i(\boldsymbol{y})) \\
&= -\sum_{i=1}^{n} \mathbf{1}_{i=y} \log\left(\frac{\exp(y_i)}{\sum_{j=1}^{n}\exp(y_j)}\right) \\
&= -\log\left(\frac{\exp(y_t)}{\sum_{j=1}^{n}\exp(y_j)}\right) \\
&= -\log\left(\frac{\exp(y_t)/\exp(y_t)}{\sum_{j=1}^{n}\exp(y_j)/\exp(y_t)}\right) \\
&= -\log\left(\frac{1}{\sum_{j=1}^{n}\exp(y_j - y_t)}\right) \\
&= \log\left(\sum_{j=1}^{n}\exp(y_j - y_t)\right) - \log(1) \\
&= \log\left(1 + \sum_{\substack{j=1 \\ j\neq y}}^{n}\exp(y_j - y_t)\right)
\end{aligned}$$

# C Additional Experimental Details

In this section, we provide detailed informations on the exact experimental setup.

**Datasets** We conduct experiments on the MNIST (LeCun et al., 2010), CIFAR-10 (Krizhevsky et al., 2009), and TinyImageNet (Le & Yang, 2015) datasets. For TinyImageNet and CIFAR-10 we follow Shi et al. (2021) and use random horizontal flips and random cropping as data augmentation during training and normalize inputs after applying perturbations. Following prior work

(Xu et al., 2020; Shi et al., 2021), we evaluate CIFAR-10 and MNIST on their test sets and TINYIMAGENET on its validation set, as test set labels are unavailable. Following Xu et al. (2020) and in contrast to Shi et al. (2021), we train and evaluate TINYIMAGENET with images cropped to $56 \times 56$.

**Training Hyperparameters** We mostly follow the hyperparameter choices from Shi et al. (2021) including their weight initialization and warm-up regularization[2], and use ADAM (Kingma & Ba, 2015) with an initial learning rate of $5 \times 10^{-4}$, decayed twice with a factor of 0.2. For CIFAR-10 we train 160 an 180 epochs for $\epsilon = 2/255$ and $\epsilon = 8/255$, respectively, decaying the learning rate after 120 and 140 and 140 and 160 epochs. For TINYIMAGENET $\epsilon = 1/255$ we use the same settings as for CIFAR-10 at $\epsilon = 8/255$. For MNIST we train 70 epochs, decaying the learning rate after 50 and 60 epochs. We choose a batch size of 128 for CIFAR-10 and TINYIMAGENET, and 256 for MNIST. We use $\ell_1$ regularization with factors according to Table 5. For all datasets, we perform one epoch of standard training ($\epsilon = 0$) before annealing $\epsilon$ from 0 to its final value over 80 epochs for CIFAR-10 and TINYIMAGENET and for 20 epochs for MNIST. We use an $n = 8$ step PGD attack with an initial step size of $\alpha = 0.5$, decayed with a factor of 0.1 after the $4^{\text{th}}$ and $7^{\text{th}}$ step to select the centre of the propagation region. We use a constant subselection ratio $\lambda$ with values shown in Table 5. For CIFAR-10 $\epsilon = 2/255$ we use shrinking with $c_s = 0.8$ (see below).

Table 5: Hyperparameters for the experiments shown in Table 1.

| Dataset | $\epsilon$ | $\ell_1$ | $\lambda$ |
|---|---|---|---|
| MNIST | 0.1 | $10^{-5}$ | 0.4 |
| | 0.3 | $10^{-6}$ | 0.6 |
| CIFAR-10 | 2/255 | $10^{-6}$ | 0.1 |
| | 8/255 | 0 | 0.7 |
| TINYIMAGENET | 1/255 | $10^{-6}$ | 0.4 |

**ReLU-Transformer with Shrinking** Additionally to standard SABR, outlined in §3, we propose to amplify the BOX growth rate reduction (see §4) affected by smaller propagation regions, by adapting the ReLU transformer as follows:

$$\bar{y}_i = \begin{cases} 0, & \text{if } \bar{x}_i + \delta_{x,i} \leq 0 \\ c_s \frac{\bar{x}_i + \delta_{x,i}}{2}, & \text{elif } \bar{x}_i - \delta_{x,i} \leq 0 \\ \bar{x}_i, & \text{else} \end{cases}, \qquad \delta_{y,i} = \begin{cases} 0, & \text{if } \bar{x}_i + \delta_{x,i} \leq 0 \\ c_s \frac{\bar{x}_i + \delta_{x,i}}{2}, & \text{elif } \bar{x}_i - \delta_{x,i} \leq 0 \\ \delta_{x,i}, & \text{else} \end{cases}. \tag{19}$$

We call $c_s$ the shrinking coefficient, as the output radius of unstable ReLUs is shrunken by multiplying it with this factor. We note that we only use these transformers for the CIFAR-10 $\epsilon = 2/255$ network discussed in Table 1.

**Architectures** Similar to prior work (Shi et al., 2021), we consider a 7-layer convolutional architecture, `CNN7`. The first 5 layers are convolutional layers with filter sizes [64, 64, 128, 128, 128], kernel size 3, strides [1, 1, 2, 1, 1], and padding 1. They are followed by a fully connected layer with 512 hidden units and the final classification. All but the last layers are followed by batch normalization (Ioffe & Szegedy, 2015) and ReLU activations. For the BN layers, we train using the statistics of the unperturbed data similar to Shi et al. (2021). During the PGD attack we use the BN layers in evaluation mode. We further consider narrower version, `CNN7-narrow` which is identical to `CNN7` expect for using the filter sizes [32, 32, 64, 64, 64] and a fully connected layer with 216 hidden units.

**Hardware and Timings** We train and certify all networks using single NVIDIA RTX 2080Ti, 3090, Titan RTX, or A6000. Training takes roughly 3 and 7 hours for MNIST and CIFAR-10, respectively, with TINYIMAGENET taking two and a half days on a single NVIDIA RTX 2080Ti. For more Details see Table 6. Verification with MN-BAB takes around 34h for MNIST, 28h for CIFAR-10 and 2h for TINYIMAGENET on a NVIDIA Titan RTX.

Table 6: SABR training times on a single NVIDIA RTX 2080Ti.

| Dataset | $\epsilon$ | Time |
|---|---|---|
| MNIST | 0.1 | 3h 23 min |
| | 0.3 | 3h 20 min |
| CIFAR-10 | 2/255 | 7h 6 min |
| | 8/255 | 7h 20 min |
| TINYIMAGENET | 1/255 | 57h 24 min |

[2]For the ReLU warm-up regularization, the bounds of the small boxes are considered.

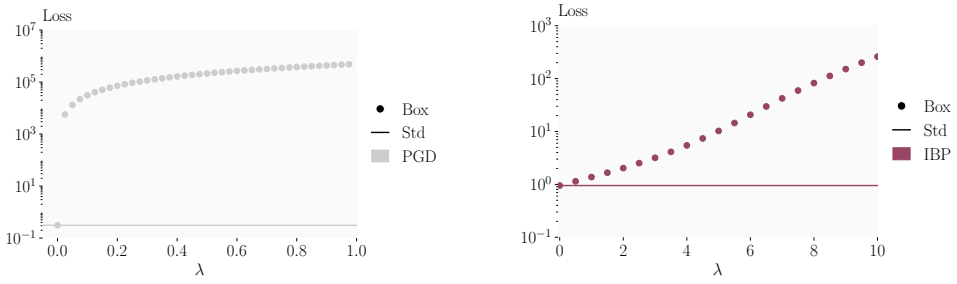



Figure 9: Standard (Std.) and robust cross-entropy loss, computed with BOX (Box) bounds for an adversarially (left) and IBP (right) trained network over subselection ratio $\lambda$. Note the logarithmic y-scale and different axes.

Table 7: Comparison of the standard (Acc.), adversarial (Adv. Acc), and certified (Cert. Acc.) accuracy for different certified training methods on the full CIFAR-10 test set. We use MN-BAB (Ferrari et al., 2022) to compute all certified and adversarial accuracies.

| $\epsilon_\infty$ | Training Method | Source | Acc. [%] | Adv. Acc. [%] | Cert. Acc. [%] |
|---|---|---|---|---|---|
| 2/255 | COLT | Balunovic & Vechev (2020) | 78.42 | **66.17** | 61.02 |
| | CROWN-IBP | Zhang et al. (2020)† | 71.27 | 59.58 | 58.19 |
| | IBP | Shi et al. (2021) | - | - | - |
| | SABR | this work | **79.52** | 65.76 | **62.57** |
| 8/255 | COLT | Balunovic & Vechev (2020) | 51.69 | 31.81 | 27.60 |
| | CROWN-IBP | Zhang et al. (2020)† | 45.41 | 33.33 | 33.18 |
| | IBP | Shi et al. (2021) | 48.94 | 35.43 | **35.30** |
| | SABR | this work | **52.00** | **35.70** | 35.25 |

- No network published.
† Published network does not match reported performance.

## D ADDITIONAL EXPERIMENTAL RESULTS

**Loss Analysis** In Fig. 9, we show the error growth of an adversarially trained (left) and IBP trained model over increasing subselection ratios $\lambda$. We observe that errors grow only slightly super-linear rather than exponential for the adversarially trained network. We trace this back to the large portion of crossing ReLUs (Table 4), especially in later layers, leading to the layer-wise growth being only linear. For the IBP trained model, in contrast, we observe exponential growth across a wide range of propagation region sizes, as the heavy regularization leads to a small portion of active and unstable ReLUs. In Fig. 10, we compare errors for BOX centred around the unperturbed sample (BOX Std) and around a high loss point computed with an adversarial attack (BOX Adex). We observe that while the loss is larger around the adversarial centres, especially for small propagation regions, this effect is small compared to the difference between training or certification methods.

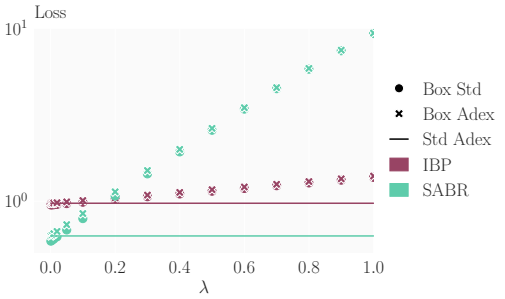



Figure 10: Comparison of the robust cross-entropy losses computed with BOX (Box) centered around unperturbed and adversarial examples for an IBP and SABR trained network over subselection ratio $\lambda$.

### D.1 EFFECT OF VERIFICATION METHOD ON OTHER CERTIFIED DEFENSES

In this section we compare different certified defenses when evaluated using the same, precise verifier MN-BAB (Ferrari et al., 2022). While COLT (Balunovic & Vechev, 2020) and IBP-R (Palma et al., 2022) trained networks were verified using similarly expensive and precise verification methods as MN-BAB (MILP (Tjeng et al., 2019) and $\beta$-CROWN (Wang et al., 2021), respectively), the IBP and CROWN-IBP trained networks were originally verified using much less precise BOX propagation. We compare standard (Acc.), empirical adversarial (Adv. Acc.), and certified (Cert.

Acc.) accuracy for CIFAR-10 at $\epsilon = 2/255$ and $\epsilon = 8/255$ in Table 7. We omit IBP-R, as neither code nor networks are published. For CROWN-IBP (Zhang et al., 2020), we evaluated both the 'best' and last checkpoints of the published networks but observed that the standard accuracy of neither matched the ones reported in the paper. We report the better of the two ('best').

We observe that certified accuracies increase only minimal in most settings, with the exception of CROWN-IBP at $\epsilon = 2/255$, where the certified accuracy rises from $54.0\%$ to $58.2\%$. However, there the adversarial accuracy of $59.6\%$ remains significantly below our certified accuracy of $62.6\%$. At $\epsilon = 8/255$ the IBP trained network achieves $35.3\%$ certified accuracy, matching SABR's performance, however at a $3\%$ lower standard accuracy.

### D.2 Ablation SABR

To assess the different components of SABR, we conduct an ablation study on CIFAR-10 with $\epsilon = 2/255$. Beyond the subselection ratio , discussed in §5 and especially Fig. 7, SABR has two main components: i) the choice of propagation region position and ii) the propagation method.

Table 8: Ablation of SABR's components with respect to standard (Acc.) and certified (Cert. Acc.) accuracy on the first 1000 samples of the CIFAR-10 test set at $\epsilon = 2/255$.

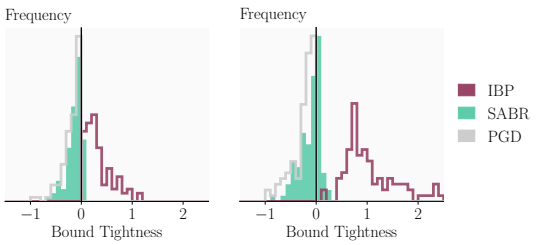
| Training Method | Acc. [%] | Cert. Acc. [%] |
|---|---|---|
| SABR | 80.4 | 61.0 |
| + centred | 82.5 | 27.4 |
| + random | 84.1 | 28.3 |
| + FGSM | 80.8 | 58.2 |
| + no projection | 80.0 | 61.9 |
| + CROWN-IBP | 80.3 | 56.7 |

To analyse the effect of the propagation region's position, we evaluate four methods to compute its center $\boldsymbol{x}'$ for $\lambda = 0.05$: i) always choose the original input as center (centred: $\boldsymbol{x}' = \boldsymbol{x}$), ii) choose the center uniformly at random such that the propagation region lies in the original adversarial region (random: $\boldsymbol{x}' \sim \mathcal{U}(\mathcal{B}^{\epsilon-\tau}(\boldsymbol{x}))$), iii) choose the centre with a weak adversarial attack such as FGSM (Goodfellow et al., 2015) (FGSM: $\boldsymbol{x}' = \boldsymbol{x} + (\epsilon - \tau)\,\mathrm{sign}(\nabla_{\boldsymbol{x}} \mathcal{L}_{\mathrm{CE}}(\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}), t))$), and iv) choose the centre with a strong adversarial attack over the whole input region $\mathcal{B}^{\epsilon}$, without projecting it into $\mathcal{B}^{\epsilon-\tau}$, allowing it to 'stick out' (no projection: $\boldsymbol{x}' = \boldsymbol{x}^*$). We observe that choosing centres either at random or as the original input leads to weaker regularization, increasing standard accuracy slightly ($+3.5\%$ and $+2.1\%$, respectively) but significantly reducing certified accuracy ($-32\%$ and $-33\%$, respectively). Using a weaker adversarial attack has a similar but much less pronounced effect, increasing natural accuracy by $0.4\%$ at the cost of a $3.2\%$ reduction in certified accuracy. Permitting propagation regions to protrude from the original input region increases regularization, leading to a slight increase in certified accuracy ($+0.9\%$) at the cost of decreased standard accuracy ($-0.4\%$).

To assess the effect of the propagation method, we compare standard SABR which uses IBP, yielding an easier optimization problem (Jovanovic et al., 2021), to CROWN-IBP, which generally yields tighter bounds. We observe that using the less precise Box propagation indeed yields better standard and certified accuracy.

### D.3 Bound Tightness

To support the intuitions discussed in §3, we compare the tightness of IBP, SABR, and PGD bounds on the worst-case margin loss. To make the computation of the exact worst-case loss ($y^{\Delta}_{\mathrm{MILP}}$) via mixed integer linear programming (MILP) and using the encoding from Tjeng et al. (2019) tractable, we train a small network (2 convolutional layers and 1 linear layer) with IBP and SABR ($\lambda = 0.1$) on MNIST at $\epsilon = 0.1$. We compute bounds on the minimum logit difference $y^{\Delta} := \min_i y_t - y_i$ for the first 100 test set samples using SABR as during training (8-step PGD attack targeting the Cross-Entropy loss), using PGD with a 50-step margin attack and 3 restarts per adversarial label, and using IBP.




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

Figure 11: Tightness comparison of bounds computed with IBP ($\lambda = 1$), SABR ($\lambda = 0.1$), and an adversarial attack (PGD), relative to the exact MILP (Tjeng et al., 2019) solution. A bound tightness greater than 0 indicates that the computed approximate bound is over-approximate (e.g. for certification), one below 0 is under-approximate (e.g. an adversarial attack).

In Fig. 11, we show histograms of the bound tightness ($y^{\Delta}_{\text{MILP}} - y^{\Delta}_{\text{approx}}$), where results greater 0 correspond to the bound (PGD, IBP, or SABR) being *larger* than the actual worst case loss and results smaller 0 correspond to the bound being *smaller*. A sound verification method will thus always yield positive bound tightness and is the more precise the smaller its absolute value. Similarly, adversarial attack based methods will always yield negative bounds, with stronger attacks generally yielding smaller magnitudes.

We observe that, especially for the SABR trained network, IBP bounds are relatively loose. SABR bounds are generally tighter than both PGD and IBP bounds (smaller mean magnitude), although clearly not sound (as discussed in §3). We highlight that the PGD attack used to compute the bound is significantly stronger than the one used for SABR. We conclude that while SABR bounds often do not include the true worst case loss, they represent a better proxy than either IBP or PGD bounds.

### D.4 Box Growth Rates of Trained Networks

In Table 9, we compare the mean and max row-wise $\ell_1$-norm of the effective weight matrices of PGD, SABR and IBP trained networks for CIFAR-10 and $\epsilon = 2/255$, depending on the network layer. Where applicable, we combine batch normalization layers with the preceding affine layers. As we show in §4, this corresponds to the mean and maximum growth rate $\kappa$ for Box of equal side lengths.

Table 9: Mean and maximum row-wise $\ell_1$-norm of effective weight matrices. Where applicable ($^*$), BN layers are merged with the preceding convolutional (Conv) and linear (Lin) layer.

| Layer | PGD | | SABR | | IBP | |
|---|---|---|---|---|---|---|
| | mean | max | mean | max | mean | max |
| Conv 1$^*$ | 4.56 | 12.50 | 0.73 | 2.36 | 0.64 | 2.74 |
| Conv 2$^*$ | 11.88 | 21.44 | 3.21 | 12.93 | 2.54 | 9.00 |
| Conv 3$^*$ | 11.29 | 18.33 | 4.19 | 17.18 | 17.81 | 58.97 |
| Conv 4$^*$ | 20.86 | 31.53 | 4.48 | 22.18 | 6.60 | 25.75 |
| Conv 5$^*$ | 22.47 | 61.31 | 5.62 | 39.71 | 31.77 | 272.52 |
| Lin 1$^*$ | 30.60 | 98.06 | 19.52 | 83.62 | 32.25 | 65.13 |
| Lin 2 | 40.07 | 47.29 | 34.32 | 42.06 | 97.20 | 113.65 |

We observe that growth rates generally increase with network depth. Interestingly training with Box, either in the form of SABR or IBP significantly reduces growth rates in the early layers, but much less in later layers. For IBP trained networks, the maximum growth rate in later layers can even exceed that of the PGD trained network.

### D.5 Training Algorithm

**Algorithm 1** get_propagation_region

**Input:** Neural network $\boldsymbol{h}$, input $\boldsymbol{x}$, label $t$, perturbation radius $\epsilon$, subselection ratio $\lambda$, step size $\alpha$, step number $n$

**Output:** Center $\boldsymbol{x}'$ and radius $\tau$ of propagation region $\mathcal{B}^{\tau}(\boldsymbol{x}')$

$(\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}) \leftarrow \text{clamp}((\boldsymbol{x} - \epsilon, \boldsymbol{x} + \epsilon), 0, 1)$ // Get bounds of input region

$\boldsymbol{\tau} \leftarrow \lambda/2 \cdot (\overline{\boldsymbol{x}} - \underline{\boldsymbol{x}})$ // Compute propagation region size $\tau$

$\boldsymbol{x}^*_0 \leftarrow \text{Uniform}(\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}})$ // Sample PGD initialization

**for** $i = 0 \dots n-1$ **do** // Do $n$ PGD steps

  $\boldsymbol{x}^*_{i+1} \leftarrow \boldsymbol{x}^*_i + \alpha \cdot \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}^*_i} \mathcal{L}_{\text{CE}}(h(\boldsymbol{x}^*_i), t))$

  $\boldsymbol{x}^*_{i+1} \leftarrow \text{clamp}(\boldsymbol{x}^*_{i+1}, \underline{\boldsymbol{x}}, \overline{\boldsymbol{x}})$

**end for**

$\boldsymbol{x}' \leftarrow \text{clamp}(\boldsymbol{x}^*_n, \underline{\boldsymbol{x}} + \tau, \overline{\boldsymbol{x}} - \tau)$ // Ensure that $\mathcal{B}^{\tau}(\boldsymbol{x}')$ will lie fully in $\mathcal{B}^{\epsilon}(\boldsymbol{x})$

**return** $\boldsymbol{x}', \tau$

**Algorithm 2** SABR Training Epoch

**Input:** Neural network $\boldsymbol{h}_\theta$, training set $(\boldsymbol{X}, \boldsymbol{T})$, perturbation radius $\epsilon$, subselection ratio $\lambda$, learning rate $\eta$, $\ell_1$ regularization weight $\ell_1$

**for** $(\boldsymbol{x}, t) = (\boldsymbol{x}_0, t_0) \dots (\boldsymbol{x}_b, t_b)$ **do** // Sample batches $\sim (\boldsymbol{X}, \boldsymbol{T})$

  $(\boldsymbol{x}', \tau) \leftarrow$ get_propagation_region // Refer to Algorithm 1

  $\mathcal{B}^\tau(\boldsymbol{x}') \leftarrow \textsc{Box}(\boldsymbol{x}', \tau)$ // Get box with midpoint $\boldsymbol{x}'$ and radius $\tau$

  $\boldsymbol{u}_{y^\Delta} \leftarrow$ get_upper_bound$(\boldsymbol{h}_\theta, \mathcal{B}^\tau(\boldsymbol{x}'))$ // Get upper bound $\boldsymbol{u}_{y^\Delta}$ on logit differences // based on IBP

  loss $\leftarrow \mathcal{L}_{\text{CE}}(\boldsymbol{u}_{y^\Delta}, t)$

  $\text{loss}_{\ell_1} \leftarrow \ell_1 \cdot$ get_$\ell_1$_norm$(h_\theta)$

  $\text{loss}_{tot} \leftarrow \text{loss} + \text{loss}_{\ell_1}$

  $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \text{loss}_{tot}$ // Update model parameters $\theta$

**end for**