Appendix to Wasserstein distributional robustness of neural networks

Anonymous Author(s)
Affiliation
Address
`email`

A Bounds on Adversarial Accuracy

Recall that Proposition 5.2 states the following tower-like property

$$V(\delta) = \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}] = o(\delta),$$

where

$$C(\delta) = \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[J_\theta(x,y)|S] \quad \text{and} \quad W(\delta) = \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[J_\theta(x,y)|S^c].$$

Proof of Proposition 5.2. One direction follows directly from the usual tower property of conditional expectation:

$$\begin{aligned} V(\delta) = \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[J(x,y)] &= \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[\mathbf{E}_Q[J(x,y)|\sigma(S)]] \\ &\leqslant \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}]. \end{aligned}$$

For the other direction, note that

$$Q(E|S) = Q(E \cap S)/Q(S) \quad \text{and} \quad Q(E|S^c) = Q(E \cap S^c)/Q(S^c),$$

are well defined for any Borel E by Assumption 5.1. Take an arbitrary $\varepsilon > 0$ and $Q_c, Q_w \in B_\delta(P)$ such that

$$\mathbf{E}_{Q_c}[J(x,y)|S] \geqslant C(\delta) - \varepsilon \quad \text{and} \quad \mathbf{E}_{Q_w}[J(x,y)|S^c] \geqslant W(\delta) - \varepsilon.$$

We further take $Q_\star \in B_\delta(P)$ such that

$$\mathbf{E}_{Q_\star}[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}] \geqslant \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}] - \varepsilon,$$

and write distribution $\widetilde{Q}$ given by

$$\widetilde{Q}(E) = Q_c(E|S)Q_\star(S) + Q_w(E|S^c)Q_\star(S^c).$$

These give us

$$\begin{aligned} \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}] \leqslant& \mathbf{E}_{Q_\star}[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}] + \varepsilon \\ \leqslant& \mathbf{E}_{Q_\star}\big[\mathbf{E}_{Q_c}[J_\theta(x,y)|S]\mathbb{1}_S + \mathbf{E}_{Q_w}[J_\theta(x,y)|S^c]\mathbb{1}_{S^c}\big] + 3\varepsilon \\ =& \mathbf{E}_{\widetilde{Q}}[J_\theta(x,y)] + 3\varepsilon. \end{aligned}$$

Recall Assumption 5.1 (ii) gives for any $Q \in B_\delta(P)$

$$\mathcal{W}_p(Q(\cdot|S), P(\cdot|S)) + \mathcal{W}_p(Q(\cdot|S^c), P(\cdot|S^c)) = o(\delta).$$

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Now we take $\pi_c \in \Pi(Q_\star(\cdot|S), Q_c(\cdot|S))$ such that $\mathbf{E}_{\pi_c}[d(X,Y)^p] = \mathcal{W}_p(Q_\star(\cdot|S), Q_c(\cdot|S))^p$, and similarly $\pi_w \in \Pi(Q_\star(\cdot|S^c), Q_w(\cdot|S^c))$. Then by definition of $\widetilde{Q}$, we have $\pi = Q_\star(S)\pi_c + Q_\star(S^c)\pi_w \in \Pi(Q_\star, \widetilde{Q})$. Moreover, we derive

$$\begin{aligned}\mathcal{W}_p(Q_\star, \widetilde{Q})^p \leqslant \mathbf{E}_\pi[d(X,Y)^p] &= Q_\star(S)\mathbf{E}_{\pi_c}[d(X,Y)^p] + Q_\star(S^c)\mathbf{E}_{\pi_w}[d(X,Y)^p] \\ &= Q_\star(S)\mathcal{W}_p(Q_\star(\cdot|S), Q_c(\cdot|S))^p + Q_\star(S^c)\mathcal{W}_p(Q_\star(\cdot|S^c), Q_w(\cdot|S^c))^p \\ &= o(\delta^p),\end{aligned}$$

which implies $\mathcal{W}_p(P, \widetilde{Q}) = \delta + o(\delta)$ by triangle inequality. Hence, by L-Lipschitzness of J_θ and (Bartl et al., 2021, Appendix Corollary 7.5) we obtain

$$\sup_{Q \in B_\delta(P)} \mathbf{E}_Q[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}] \leqslant V(\delta + o(\delta)) + 3\varepsilon \leqslant V(\delta) + o(\delta) + 3\varepsilon.$$

Finally, by taking $\varepsilon \to 0$, we deduce

$$V(\delta) = \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}] + o(\delta)$$

which concludes the proof of Proposition 5.2. □

Now, we present the proof of the lower bound estimate in Theorem 5.1,

$$\mathcal{R}_\delta = \frac{A_\delta}{A} \geqslant \frac{W(0) - V(0)}{W(0) - V(0)}.$$

Proof of Theorem 5.1. By adding and substracting $W(\delta)\mathbb{1}_S$, Proposition 5.2 now gives

$$\begin{aligned}V(\delta) &= \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[C(\delta)\mathbb{1}_S + W(\delta)\mathbb{1}_{S^c}] + o(\delta) \\ &= \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[W(\delta) - (W_\delta - C(\delta))\mathbb{1}_S] + o(\delta) \\ &= W(\delta) - (W(\delta) - C(\delta))A_\delta + o(\delta).\end{aligned}$$

Naturally, we can rewrite loss $V(0)$ using the usual tower property

$$V(0) = AC(0) + (1-A)W(0) = W(0) - (W(0) - C(0))A.$$

Together these yield

$$\begin{aligned}V(\delta) - V(0) &= (1 - A_\delta)(W(\delta) - W(0)) + A_\delta(C(\delta) - C(0)) + (W(0) - C(0))(A - A_\delta) + o(\delta) \\ &\geqslant (W(0) - C(0))(A - A_\delta) + o(\delta).\end{aligned}$$

Plugging in $C(0) = [V(0) - (1-A)W(0)]/A$ completes the proof. □

B Bounds on Out-of-Sample Performance

The concentration inequality in Fournier and Guillin (2015) is pivotal to derive the out-of-sample performance bounds. It characterizes how likely an empirical measure can deviate from its generating distribution. Recall we denote $\widehat{P}$ as the empirical measure of training set $\mathcal{D}_{tr}$ with size N and $\breve{P}$ as the empirical measure of test set $\mathcal{D}_{tt}$ with size M. We use the same $\widehat{\ }$, $\breve{\ }$ notations to denote quantities computed from $\widehat{P}$ and $\breve{P}$, respectively. We restate the concentration inequality as

$$\mathbb{P}(\mathcal{W}_p(\widehat{P}, P) \geqslant \varepsilon) \leqslant K\exp(-KN\varepsilon^n),$$

where K is a constant only depending on n and p. For convenience, K might change from line to line in this section. Together with Theorem 5.1, we give an out-of-sample clean accuracy guarantee in Corollary 5.3.

Proof of Corollary 5.3. By triangle inequality and concentration inequality, we have

$$\mathbb{P}(\mathcal{W}_p(\breve{P}, \widehat{P}) \geqslant 2\varepsilon) \leqslant \mathbb{P}(\mathcal{W}_p(\breve{P}, P) \geqslant \varepsilon) + \mathbb{P}(\mathcal{W}_p(P, \widehat{P}) \geqslant \varepsilon) \leqslant 2K\exp(-K\varepsilon^n \min\{M, N\}).$$

With Theorem 5.1, it implies that with probability at least $1 - 2K\exp(-K\varepsilon^n \min\{M, N\})$, we have

$$\breve{A} \geqslant \widehat{A}_{2\varepsilon} \geqslant \widehat{A}\widehat{R}^l_{2\varepsilon} + o(\varepsilon).$$

□

The following results provide a guarantee on the out-of-sample adversarial performance.

Proof of Theorem 5.4. Estimates in Fournier and Guillin (2015) imply that with probability at least $1 - K\exp(-KN\varepsilon^n)$, we have $B_\delta(\widehat{P}) \subseteq B_{\delta+\varepsilon}(P)$. Hence, we derive $V(\delta) \leqslant \widehat{V}(\delta+\varepsilon)$. On the other hand, since J_θ is L-Lipschitz, from (Bartl et al., 2021, Appendix Corollary 7.5) we obtain

$$\widehat{V}(\delta+\varepsilon) = \widehat{V}(\delta) + \varepsilon \sup_{Q \in B_\delta^\star(\widehat{P})} \Big(\mathbf{E}_Q \|\nabla_x J_\theta(x,y)\|_*^q \Big)^{1/q} + o(\varepsilon),$$

where $B_\delta^\star(\widehat{P}) = \arg\max_{Q \in B_\delta(\widehat{P})} \mathbf{E}_Q[J_\theta(x,y)]$. Combining above results, we conclude that with probability at least $1 - K\exp(-KN\varepsilon^n)$

$$V(\delta) \leqslant \widehat{V}(\delta) + \varepsilon \sup_{Q \in B_\delta^\star(\widehat{P})} \Big(\mathbf{E}_Q \|\nabla_x J_\theta(x,y)\|_*^q \Big)^{1/q} + o(\varepsilon) \leqslant \widehat{V}(\delta) + L\varepsilon.$$

□

Proof Corollary 5.5. By Theorem 5.1, we have

$$\Delta A_\delta(P) \leqslant \frac{V(\delta) - V(0)}{W(0) - C(0)} + o(\delta).$$

We now bound the numerator and denominator separately. By taking $\varepsilon = \delta$ in Theorem 5.4, we have with probability at least $1 - K\exp(-KN\delta^n)$, $V(\delta) \leqslant \widehat{V}(\delta) + L\delta$. Similarly, we can also show that $V(0) \geqslant \widehat{V}(0) - L\delta$. Hence, we have with probability at least $1 - K\exp(-KN\delta^n)$,

$$V(\delta) - V(0) \leqslant \widehat{V}(\delta) - \widehat{V}(0) + 2L\delta.$$

For the denominator, notice that $\mathbb{P}(\widehat{P} \in B_\delta(P)) \geqslant 1 - K\exp(-KN\exp(\delta^n))$. By Assumption 5.1 (ii), we have with probability at least $1 - K\exp(-KN\delta^n)$,

$$\mathcal{W}_p(\widehat{P}(\cdot|S), P(\cdot|S)) + \mathcal{W}_p(\widehat{P}(\cdot|S^c), P(\cdot|S^c)) = o(\delta).$$

This implies

$$|C(0) - W(0) - \widehat{C}(0) + \widehat{W}(0)| = o(\delta).$$

Combining above results, we conclude that with probability at least $1 - K\exp(-KN\delta^n)$,

$$\Delta A_\delta(P) \leqslant \frac{\widehat{V}(\delta) - \widehat{V}(0)}{\widehat{W}(0) - \widehat{C}(0)} + \frac{2L\delta}{\widehat{W}(0) - \widehat{C}(0)} + o(\delta).$$

□

C W-PGD Algorithm

We give the details of W-PGD algorithm implemented in this paper. Recall in Section 4, we propose

$$x^{t+1} = \operatorname{proj}_\delta\big(x^t + \alpha h(\nabla_x J_\theta(x^t,y)) \|\Upsilon^{-1}\nabla_x J_\theta(x^t,y)\|_s^{q-1}\big), \tag{1}$$

where we take $\|\cdot\| = \|\cdot\|_r$ and hence h is given by $\langle h(x), x\rangle = \|x\|_s$. In particular, $h(x) = \operatorname{sgn}(x)$ for $s = 1$ and $h(x) = x/\|x\|_2$ for $s = 2$. The projection step $\operatorname{proj}_\delta$ is based on any off-the-shelf optimal transport solver $\mathscr{S}$ which pushes the adversarial images back into the Wasserstein ball along the geodesics. The solver $\mathscr{S}$ gives the optimal matching T between the original test data $\mathcal{D}_{tt}$ and the perturbed test data $\mathcal{D}'_{tt}$. Formally, $\operatorname{proj}_\delta$ maps

$$x' \mapsto x + \delta d^{-1}(T(x) - x),$$

where $d = \Big\{\frac{1}{|\mathcal{D}_{tt}|}\sum_{(x,y)\in\mathcal{D}_{tt}} \|T(x) - x\|_r^p\Big\}^{1/p}$ the Wasserstein distance between $\breve{P}$ and $\breve{P}'$. See Algorithm 1 for pseudocodes. In numerical experiments, due to the high computational cost of the OT solver, we always couple each image with its own perturbation.

Algorithm 1: W-PGD Attack

Input: Model parameter θ, attack strength δ, ratio r, iteration step I, OT solver $\mathscr{S}$;
Data: Test set $\mathcal{D}_{tt} = \{(x,y)|(x,y) \sim P\}$ with size M;

```
def proj_δ(D_tt, D'_tt):
    T = S(D_tt, D'_tt);                                   // Generate transport map from OT solver
    d = { (1/M) Σ_{(x,y)∈D_tt} ||T(x) − x||_r^p }^{1/p};   // Calculate the Wasserstein distance
    for (x, y) in D_tt do
        x' ← x + δ d^{-1} (T(x) − x);                     // Project back to the Wasserstein ball
        x' ← clamp(x', 0, 1);
    return D'_tt.
def attack(D_tt):
    α ← rδ/I;                                            // Calculate stepsize
    D_tt^adv ← D_tt;
    for 1 ⩽ i ⩽ I do
        Υ = ( (1/M) Σ_{(x,y)∈D_tt^adv} ||∇_x J_θ(x,y)||_s^q )^{1/q};   // Calculate Υ
        for (x, y) ∈ D_tt^adv do
            (x, y) ← (x + α h(∇_x J_θ(x,y)) ||Υ^{-1} ∇_x J_θ(x,y)||_s^{q−1}, y);
        D_tt^adv = proj_δ(D_tt, D_tt^adv);
        return D_tt^adv.
```

D Wasserstein Distributionally Adversarial Training

Theorem 4.1 offers natural computationally tractable approximations to the W-DRO training objective

$$\inf_{\theta \in \Theta} \sup_{Q \in B_\delta(P)} \mathbf{E}_Q[J_\theta(x,y)]$$

and its extension

$$\inf_{\theta \in \Theta} \sup_{\pi \in \Pi_\delta(P,\cdot)} \mathbf{E}_\pi[J_\theta(x,y,x',y')].$$

First, consider a regularized optimization problem:

$$\inf_{\theta \in \Theta} \mathbf{E}_P[J_\theta(x,y) + \delta \Upsilon].$$

The extra regularization term $\delta\Upsilon$ allows us to approximate the W-DRO objective above. A similar approach has been studied in García Trillos and García Trillos (2022) in the context of Neural ODEs, and Sinha et al. (2018) considered Wasserstein distance penalization and used duality.

Training is done by replacing P with $\widehat{P}$. Note that $\widehat{\Upsilon}$ is a statistics over the whole data set. In order to implement stochastic gradient descent method, an asynchronous update of the parameters is needed. We consider $\|\cdot\|_* = \|\cdot\|_s$ and, by a direct calculation, obtain

$$\nabla_\theta \widehat{\Upsilon} = \widehat{\Upsilon}^{1-q} \mathbf{E}_{\widehat{P}}[\langle \nabla_x \nabla_\theta J_\theta(x,y),\, \operatorname{sgn}(\nabla_x J_\theta(x,y)) \rangle \|\nabla_x J_\theta(x,y)\|_s^{q-1}].$$

We calculate the term $\widehat{\Upsilon}^{1-q}$ from the whole training dataset using parameter θ^* from previous epoch; we estimate

$$\mathbf{E}_{\widehat{P}}[\langle \nabla_x \nabla_\theta J_\theta(x,y),\, \operatorname{sgn}(\nabla_x J_\theta(x,y)) \rangle \|\nabla_x J_\theta(x,y)\|_s^{q-1}]$$

on a mini-batch and update current parameter θ after each batch. At the end of an epoch, we update θ^* to θ. See Algorithm 2 for pseudocodes.

Another classical approach consists in clean training the network but on the adversarial perturbed data. To this we employ the Wasserstein FGSM attack described in Section 4 and shift training data (x,y) by

$$(x,y) \mapsto \Big(\operatorname{proj}_\delta\big(x + \delta h(\nabla_x J_\theta(x,y)) \|\widehat{\Upsilon}^{-1} \nabla_x J_\theta(x,y)\|_s^{q-1}\big), y \Big).$$

Similarly to the discussion above, an asynchronous update of parameters is applied, see Algorithm 3 for details. Empirical evaluation of the performance of Algorithms 2 and 3 is left for future research.

Algorithm 2: Loss Regularization

Input: Initial parameter θ_0, hyperparameter δ, learning rate η;
Data: Training set $\mathcal{D}_{tr} = \{(x,y) | (x,y) \sim P\}$ with size N;
$\theta^* \leftarrow \theta_0, \theta \leftarrow \theta_0$;
repeat
- $\Upsilon = \left(\frac{1}{N} \sum_{(x,y)\in\mathcal{D}_{tr}} \|\nabla_x J_{\theta^*}(x,y)\|_s^q \right)^{1/q}$; `// Calculate` Υ `from` θ^*
- **repeat**
  - Generate a mini-batch B with size $|B|$;
  - `// Calculate gradient` $\nabla_\theta J_\theta(x,y)$
  - $\nabla_\theta J_\theta(x,y) = \frac{1}{|B|} \sum_{(x,y)\in B} \nabla_\theta J_\theta(x,y)$;
  - `// Calculate gradient` $\nabla_\theta \Upsilon$
  - $\nabla_\theta \Upsilon = \Upsilon^{1-q} \frac{1}{|B|} \sum_{(x,y)\in B} \langle \nabla_x \nabla_\theta J_\theta(x,y), h(\nabla_x J_\theta(x,y)) \rangle \|\nabla_x J_\theta(x,y)\|_s^{q-1}$;
  - `// Update` θ `by stochastic gradient descent`
  - $\theta \leftarrow \theta - \eta(\nabla_\theta J_\theta(x,y) + \delta \nabla_\theta \Upsilon)$;
- **until** *the end of epoch;*
- $\theta^* \leftarrow \theta$;

until *the end condition.*

Algorithm 3: Adversarial Data Perturbation

Input: Initial parameter θ_0, hyperparameter δ, learning rate η;
Data: Training set $\mathcal{D}_{tr} = \{(x,y) | (x,y) \sim P\}$ with size N;
$\theta^* \leftarrow \theta_0, \theta \leftarrow \theta_0$;
repeat
- $\Upsilon = \left(\frac{1}{N} \sum_{(x,y)\in\mathcal{D}_{tr}} \|\nabla_x J_{\theta^*}(x,y)\|_s^q \right)^{1/q}$; `// Calculate` Υ `from` θ^*
- **repeat**
  - Generate a mini-batch B with size $|B|$;
  - `// Do W-FGSM attack on` B
  - $(x,y) \leftarrow \left(\mathrm{proj}_\delta(x + \delta h(\nabla_x J_\theta(x,y)) \|\Upsilon^{-1} \nabla_x J_\theta(x,y)\|_s^{q-1}), y \right)$;
  - `// Update` θ `by stochastic gradient descent`
  - $\theta \leftarrow \theta - \eta \frac{1}{|B|} \sum_{(x,y)\in B} \nabla_\theta J_\theta(x,y)$;
- **until** *the end of epoch;*
- $\theta^* \leftarrow \theta$;

until *the end condition.*

E Robust Performance Bounds

As pointed out in Section 6, with attack budget $\delta = 8/255$ some neural networks have lower bounds $\mathcal{R}^l$ surpassing the reference value $\mathcal{R}$ obtained from AutoAttack. It is because for most of neural networks $\delta = 8/255$ is outside the linear approximation region of the adversarial loss $V(\delta)$, and we underestimate $V(\delta)$ by using first order approximations in Theorem 4.1. In Figure E.1, we plot our proposed bounds $\mathcal{R}^u$ and $\mathcal{R}^l$ against $\mathcal{R}$ for $(\mathcal{W}_\infty, l_\infty)$ threat model with $\delta = 8/255$.

In general, to compute more accurate lower bounds on the adversarial accuracy, as explained in Section 6, we can consider the first lower bound in (13). We thus introduce $\mathcal{R}^l(n)$ given by

$$\mathcal{R}^l(n) = \frac{W(0) - V(\delta, n)}{W(0) - V(0)},$$

where $V(\delta, n)$ is the approximated adversarial loss computed by a W-PDG-(n) attack. In Figures E.2 and E.3, we include plots for different bounds of $\mathcal{R}$ under $\mathcal{W}_2$ threat models which illustrate the changing performance of the lower bound in Theorem 5.1 as $V(\delta)$ is computed to an increasing accuracy. We achieve this performing a W-PDG-(n) attack, where $n = 5, 50$. An $n = 50$ attack takes

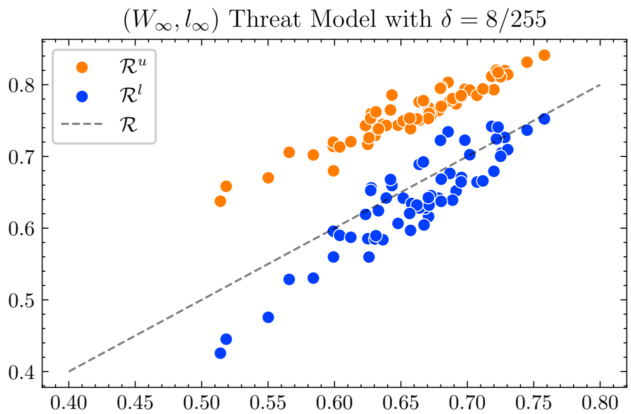

Figure E.1: $\mathcal{R}^u$&$\mathcal{R}^l$ versus $\mathcal{R}$. Bounds are computed based on CE loss across neural networks on RobustBench under a $(\mathcal{W}_\infty, l_\infty)$ threat model with budget $\delta = 8/255$. At this δ linear approximation may be inefficient as seen from the blue dots crossing the diagonal.

10 times more computational time than the $n = 5$ attack and the latter is 5 times more computational costly than the one-step bound $\mathcal{R}^l$. The plots thus illustrate a trade-off between computational time and accuracy of the proposed lower bound. For clarity, we also point out that Figure E.2 has a different scaling from the other plots.

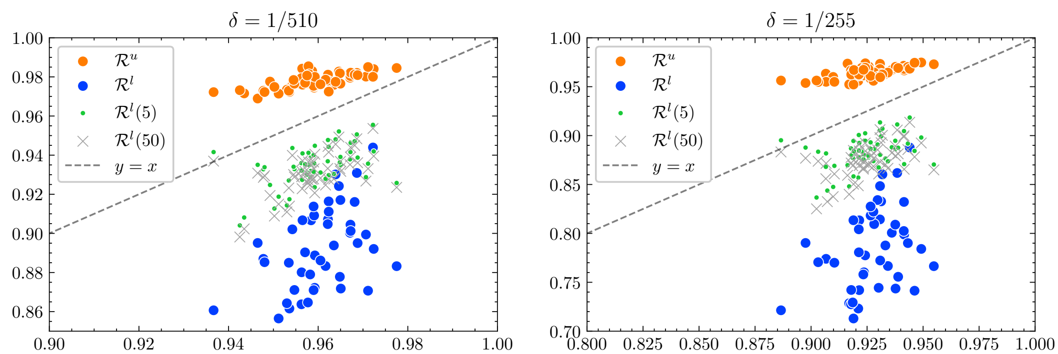

Figure E.2: Comparison of lower bounds computed from different W-PDG-(n) attacks, where $n = 5, 50$. Bounds are computed based on CE loss across neural networks on RobustBench under $(\mathcal{W}_2, l_\infty)$ threat models with budget $\delta = 1/510, 1/255$.

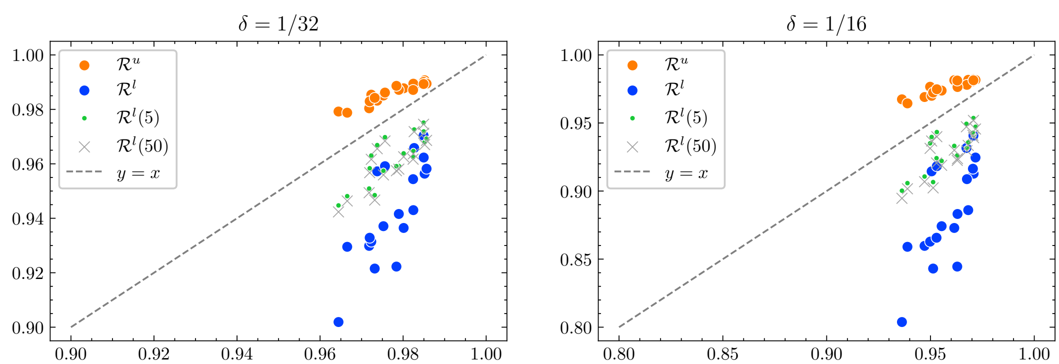

Figure E.3: Comparison of lower bounds computed from different W-PDG-(n) attacks, where $n = 5, 50$. Bounds are computed based on ReDLR loss across neural networks on RobustBench under $(\mathcal{W}_2, l_2)$ threat models with budget $\delta = 1/32, 1/16$.

F Additional Numerical Results

In Tables F.1 and F.2, we report the clean accuracy and the adversarial accuracy under different threat models for all but 4 neural networks available on RobustBench (model zoo).[1] All networks are named after their labels on RobustBench (model zoo). We remove `Standard` l_∞-network and `Standard` l_2-network because they are not robust to any attacks. In addition, we also remove `Kang2021Stable` which contains NeuralODE blocks and `Ding2020MMA` which has a huge gap between adversarial accuracies obtained from PGD and AutoAttack.

In Tables F.3, F.4, F.5 and F.6, we include the complete list of $\mathcal{R}$ and its bounds used in Figure 3. We write $\Delta\mathcal{R}^u = \mathcal{R}^u - \mathcal{R}$ and $\Delta\mathcal{R}^l = \mathcal{R}^l - \mathcal{R}$.

Table F.1: Complete list of adversarial accuracies under $(\mathcal{W}_\infty, l_\infty)$ and $(\mathcal{W}_2, l_\infty)$ threat models.

		δ=1/8		δ=1/4		δ=1/8	
Network	Clean	$\mathcal{W}_\infty$	$\mathcal{W}_2$	$\mathcal{W}_\infty$	$\mathcal{W}_2$	$\mathcal{W}_\infty$	$\mathcal{W}_2$
Augustin2020Adversarial	91.08	87.88	82.03	84.48	73.17	72.91	56.66
Augustin2020Adversarial_34_10	92.23	89.20	83.70	85.80	74.06	76.25	56.33
Augustin2020Adversarial_34_10_extra	93.96	91.63	84.86	88.32	75.04	78.79	59.73
Ding2020MMA	88.02	83.57	77.69	78.44	69.67	66.09	53.47
Engstrom2019Robustness	90.83	86.81	81.99	82.37	73.77	69.24	57.77
Gowal2020Uncovering	90.89	87.62	83.19	84.26	75.83	74.50	60.94
Gowal2020Uncovering_extra	94.73	92.64	88.08	89.52	79.74	80.53	60.18
Rade2021Helper_R18_ddpm	90.57	88.03	83.57	84.58	77.26	76.15	62.74
Rebuffi2021Fixing_28_10_cutmix_ddpm	91.79	89.26	86.09	86.43	81.06	78.80	67.18
Rebuffi2021Fixing_70_16_cutmix_ddpm	92.41	90.36	87.27	87.69	81.67	80.42	67.93
Rebuffi2021Fixing_70_16_cutmix_extra	95.74	93.78	89.73	91.18	81.59	82.32	63.43
Rebuffi2021Fixing_R18_cutmix_ddpm	90.33	87.48	84.24	84.48	78.38	75.86	63.85
Rice2020Overfitting	88.68	85.07	80.62	80.35	73.28	67.68	56.83
Rony2019Decoupling	89.04	84.74	78.47	79.25	69.39	66.44	51.15
Sehwag2021Proxy	90.93	88.42	85.50	85.66	80.50	77.24	67.96
Sehwag2021Proxy_R18	89.76	87.08	83.34	83.87	77.62	74.41	65.02
Wang2023Better_WRN-28-10	95.16	93.44	88.99	90.79	82.58	83.68	68.65
Wang2023Better_WRN-70-16	95.54	93.76	89.94	91.79	83.47	84.97	70.09
Wu2020Adversarial	88.51	85.32	81.33	82.05	75.31	73.66	62.21

[1] https://github.com/RobustBench/robustbench

Table F.2: Complete list of adversarial accuracies under $(\mathcal{W}_\infty, l_\infty)$ and $(\mathcal{W}_2, l_\infty)$ threat models.

		δ=2/255		δ=4/255		δ=8/255	
Network	Clean	$\mathcal{W}_\infty$	$\mathcal{W}_2$	$\mathcal{W}_\infty$	$\mathcal{W}_2$	$\mathcal{W}_\infty$	$\mathcal{W}_2$
Addepalli2021Towards_RN18	80.23	73.85	68.54	66.70	58.76	51.06	42.91
Addepalli2021Towards_WRN34	85.32	79.87	73.41	73.18	64.60	58.04	48.12
Addepalli2022Efficient_RN18	85.71	79.16	71.75	71.37	59.59	52.48	37.84
Addepalli2022Efficient_WRN_34_10	88.70	82.95	74.75	75.52	63.35	57.81	42.03
Andriushchenko2020Understanding	79.85	72.30	66.16	64.45	55.83	43.93	37.86
Carmon2019Unlabeled	89.69	84.59	77.02	77.87	66.59	59.53	46.32
Chen2020Adversarial	86.04	79.68	67.45	71.61	51.23	51.56	29.66
Chen2020Efficient	85.34	78.48	70.91	70.64	59.24	51.12	39.32
Chen2021LTD_WRN34_10	85.21	79.78	72.74	72.97	61.86	56.94	41.48
Chen2021LTD_WRN34_20	86.03	80.71	75.31	74.19	64.95	57.71	44.87
Cui2020Learnable_34_10	88.22	81.95	71.25	74.20	57.88	52.86	40.14
Cui2020Learnable_34_20	88.70	82.20	72.16	74.46	57.74	53.57	37.73
Dai2021Parameterizing	87.02	82.39	77.35	76.96	69.28	61.55	52.43
Debenedetti2022Light_XCiT-L12	91.73	86.59	78.09	79.42	66.77	57.58	44.90
Debenedetti2022Light_XCiT-M12	91.30	86.14	77.83	78.89	66.86	57.27	43.90
Debenedetti2022Light_XCiT-S12	90.06	84.91	77.11	77.50	65.35	56.14	44.28
Engstrom2019Robustness	87.03	80.34	74.30	72.22	63.43	49.25	41.85
Gowal2020Uncovering_28_10_extra	89.48	84.78	78.08	78.77	67.93	62.80	48.51
Gowal2020Uncovering_34_20	85.64	79.84	73.22	73.47	62.78	56.86	42.87
Gowal2020Uncovering_70_16	85.29	79.81	72.89	73.47	62.25	57.20	43.29
Gowal2020Uncovering_70_16_extra	91.10	86.86	79.81	81.06	70.01	65.88	50.55
Gowal2021Improving_28_10_ddpm_100m	87.50	83.37	78.46	78.30	70.50	63.44	54.95
Gowal2021Improving_70_16_ddpm_100m	88.74	84.76	80.48	80.08	73.27	66.11	59.34
Gowal2021Improving_R18_ddpm_100m	87.35	82.08	76.83	76.22	68.16	58.63	51.17
Hendrycks2019Using	87.11	81.36	74.48	74.21	63.45	54.92	44.19
Huang2020Self	83.48	77.59	69.33	70.55	58.46	53.34	38.96
Huang2021Exploring	90.56	85.77	77.78	79.50	67.52	61.56	47.30
Huang2021Exploring_ema	91.23	86.84	79.28	80.79	69.02	62.54	48.46
Huang2022Revisiting_WRN-A4	91.59	87.35	79.76	81.69	69.62	65.79	50.22
Jia2022LAS-AT_34_10	84.98	79.26	73.00	72.44	62.88	56.26	43.92
Jia2022LAS-AT_70_16	85.66	80.29	73.90	73.52	63.76	57.61	44.19
Pang2020Boosting	85.14	79.34	71.93	72.60	61.17	53.74	39.40
Pang2022Robustness_WRN28_10	88.61	83.73	78.46	77.16	69.51	61.04	51.67
Pang2022Robustness_WRN70_16	89.01	84.77	79.57	78.58	70.61	63.35	52.93
Rade2021Helper_ddpm	88.16	83.26	77.54	76.83	67.49	60.97	47.37
Rade2021Helper_extra	91.47	86.72	80.47	80.29	69.48	62.83	47.57
Rade2021Helper_R18_ddpm	86.86	81.12	75.67	74.39	64.92	57.09	43.99
Rade2021Helper_R18_extra	89.02	83.21	77.16	76.36	66.10	57.67	43.38
Rebuffi2021Fixing_106_16_cutmix_ddpm	88.50	84.32	78.88	78.83	70.59	64.64	52.31
Rebuffi2021Fixing_28_10_cutmix_ddpm	87.33	82.36	76.33	76.08	66.35	60.75	46.73
Rebuffi2021Fixing_70_16_cutmix_ddpm	88.54	84.31	78.64	78.79	68.95	64.25	49.75
Rebuffi2021Fixing_70_16_cutmix_extra	92.23	88.02	81.21	82.76	70.39	66.58	47.97
Rebuffi2021Fixing_R18_ddpm	83.53	77.99	71.46	71.68	61.83	56.66	44.26
Rice2020Overfitting	85.34	79.60	74.06	72.82	64.81	53.42	43.66
Sehwag2020Hydra	88.98	83.49	76.02	76.18	65.38	57.14	44.37
Sehwag2021Proxy	86.68	81.72	76.91	76.39	68.84	60.27	53.29
Sehwag2021Proxy_R18	84.59	79.25	73.87	72.74	64.73	55.54	46.58
Sehwag2021Proxy_ResNest152	87.21	82.55	78.08	77.30	70.78	62.79	56.67
Sitawarin2020Improving	86.84	80.21	74.16	72.47	63.54	50.72	43.25
Sridhar2021Robust	89.46	84.34	77.42	78.03	66.42	59.66	46.45
Sridhar2021Robust_34_15	86.53	81.45	73.95	75.63	64.18	60.41	45.77
Wang2020Improving	87.51	82.15	74.59	75.47	63.17	56.29	42.57
Wang2023Better_WRN-28-10	92.44	88.40	81.51	83.10	71.42	67.31	52.03
Wang2023Better_WRN-70-16	93.26	89.72	82.88	84.90	73.95	70.69	55.17
Wong2020Fast	83.34	75.77	69.60	67.23	58.38	43.21	36.96
Wu2020Adversarial	85.36	79.69	73.33	72.90	62.64	56.17	42.28
Wu2020Adversarial_extra	88.25	82.98	76.83	76.61	66.45	60.04	46.70
Zhang2019Theoretically	84.92	78.96	71.68	71.15	60.30	53.08	40.05
Zhang2019You	87.20	79.42	72.40	70.29	60.61	44.83	36.65
Zhang2020Attacks	84.52	78.48	71.10	71.32	59.45	53.51	40.76
Zhang2020Geometry	89.36	84.01	81.57	77.10	73.45	59.64	53.50

Table F.3: Complete list of $\mathcal{R}$ and its bounds under $(\mathcal{W}_\infty, l_2)$ threat model based on CE loss.

	δ=1/8			δ=1/4		
Network	$\mathcal{R}$	$\Delta\mathcal{R}^u$	$\Delta\mathcal{R}^l$	$\mathcal{R}$	$\Delta\mathcal{R}^u$	$\Delta\mathcal{R}^l$
Augustin2020Adversarial	0.9649	0.0026	-0.0153	0.9275	0.0083	-0.0324
Augustin2020Adversarial_34_10	0.9669	0.0035	-0.0097	0.9303	0.0115	-0.0201
Augustin2020Adversarial_34_10_extra	0.9752	0.0031	-0.0150	0.9400	0.0084	-0.0239
Ding2020MMA	0.9496	0.0016	-0.0176	0.8912	0.0081	-0.0434
Engstrom2019Robustness	0.9557	0.0019	-0.0087	0.9069	0.0057	-0.0256
Gowal2020Uncovering	0.9640	0.0022	-0.0063	0.9271	0.0045	-0.0191
Gowal2020Uncovering_extra	0.9779	0.0013	-0.0121	0.9451	0.0050	-0.0209
Rade2021Helper_R18_ddpm	0.9720	0.0017	-0.0124	0.9339	0.0061	-0.0209
Rebuffi2021Fixing_28_10_cutmix_ddpm	0.9724	0.0034	-0.0107	0.9416	0.0060	-0.0232
Rebuffi2021Fixing_70_16_cutmix_ddpm	0.9778	0.0021	-0.0137	0.9489	0.0052	-0.0259
Rebuffi2021Fixing_70_16_cutmix_extra	0.9795	0.0018	-0.0091	0.9524	0.0040	-0.0184
Rebuffi2021Fixing_R18_cutmix_ddpm	0.9686	0.0032	-0.0150	0.9352	0.0061	-0.0339
Rice2020Overfitting	0.9593	0.0030	-0.0178	0.9061	0.0090	-0.0342
Rony2019Decoupling	0.9517	0.0015	-0.0144	0.8899	0.0077	-0.0316
Sehwag2021Proxy	0.9724	0.0027	-0.0099	0.9420	0.0065	-0.0235
Sehwag2021Proxy_R18	0.9703	0.0028	-0.0133	0.9344	0.0057	-0.0271
Wang2023Better_WRN-28-10	0.9819	0.0013	-0.0098	0.9541	0.0047	-0.0147
Wang2023Better_WRN-70-16	0.9814	0.0015	-0.0069	0.9609	0.0029	-0.0163
Wu2020Adversarial	0.9640	0.0024	-0.0088	0.9270	0.0051	-0.0221

Table F.4: Complete list of $\mathcal{R}$ and its bounds under $(\mathcal{W}_2, l_2)$ threat model based on ReDLR loss.

	δ=1/32			δ=1/16		
Network	$\mathcal{R}$	$\Delta\mathcal{R}^u$	$\Delta\mathcal{R}^l$	$\mathcal{R}$	$\Delta\mathcal{R}^u$	$\Delta\mathcal{R}^l$
Augustin2020Adversarial	0.9844	0.0058	-0.0358	0.9723	0.0131	-0.0505
Augustin2020Adversarial_34_10	0.9868	0.0026	-0.0518	0.9738	0.0094	-0.0743
Augustin2020Adversarial_34_10_extra	0.9879	0.0049	-0.0452	0.9756	0.0105	-0.0652
Ding2020MMA	0.9788	0.0087	-0.0694	0.9644	0.0148	-0.1079
Engstrom2019Robustness	0.9834	0.0036	-0.0798	0.9719	0.0085	-0.1250
Gowal2020Uncovering	0.9837	0.0048	-0.0581	0.9719	0.0110	-0.0918
Gowal2020Uncovering_extra	0.9915	0.0026	-0.0574	0.9855	0.0039	-0.0898
Rade2021Helper_R18_ddpm	0.9879	0.0055	-0.0530	0.9802	0.0075	-0.0871
Rebuffi2021Fixing_28_10_cutmix_ddpm	0.9902	0.0034	-0.0572	0.9826	0.0069	-0.0912
Rebuffi2021Fixing_70_16_cutmix_ddpm	0.9920	0.0023	-0.0610	0.9852	0.0055	-0.0966
Rebuffi2021Fixing_70_16_cutmix_extra	0.9922	0.0020	-0.0539	0.9851	0.0050	-0.0852
Rebuffi2021Fixing_R18_cutmix_ddpm	0.9894	0.0034	-0.0711	0.9791	0.0081	-0.1104
Rice2020Overfitting	0.9859	0.0046	-0.0825	0.9738	0.0104	-0.1263
Rony2019Decoupling	0.9829	0.0045	-0.0855	0.9672	0.0116	-0.1267
Sehwag2021Proxy	0.9918	0.0018	-0.0767	0.9824	0.0047	-0.1165
Sehwag2021Proxy_R18	0.9890	0.0039	-0.0558	0.9789	0.0097	-0.0868
Wang2023Better_WRN-28-10	0.9902	0.0023	-0.0492	0.9832	0.0054	-0.0775
Wang2023Better_WRN-70-16	0.9919	0.0020	-0.0494	0.9850	0.0043	-0.0780
Wu2020Adversarial	0.9867	0.0037	-0.0704	0.9759	0.0092	-0.1114

Table F.5: Complete list of $\mathcal{R}$ and its bounds under $(\mathcal{W}_\infty, l_\infty)$ threat model based on CE loss.

	δ=2/255			δ=4/255		
Network	$\mathcal{R}$	$\Delta\mathcal{R}^u$	$\Delta\mathcal{R}^l$	$\mathcal{R}$	$\Delta\mathcal{R}^u$	$\Delta\mathcal{R}^l$
Addepalli2021Towards_RN18	0.9205	0.0172	-0.0234	0.8314	0.0403	-0.0390
Addepalli2021Towards_WRN34	0.9361	0.0144	-0.0266	0.8577	0.0374	-0.0399
Addepalli2022Efficient_RN18	0.9236	0.0126	-0.0194	0.8327	0.0329	-0.0312
Addepalli2022Efficient_WRN_34_10	0.9352	0.0097	-0.0167	0.8514	0.0275	-0.0218
Andriushchenko2020Understanding	0.9054	0.0125	-0.0200	0.8071	0.0296	-0.0496
Carmon2019Unlabeled	0.9431	0.0056	-0.0122	0.8682	0.0236	-0.0144
Chen2020Adversarial	0.9261	0.0074	-0.0319	0.8323	0.0206	-0.0495
Chen2020Efficient	0.9199	0.0111	-0.0107	0.8273	0.0293	-0.0177
Chen2021LTD_WRN34_10	0.9363	0.0099	-0.0224	0.8564	0.0275	-0.0347
Chen2021LTD_WRN34_20	0.9382	0.0105	-0.0280	0.8624	0.0322	-0.0476
Cui2020Learnable_34_10	0.9290	0.0070	-0.0178	0.8410	0.0196	-0.0324
Cui2020Learnable_34_20	0.9267	0.0079	-0.0150	0.8395	0.0224	-0.0296
Dai2021Parameterizing	0.9468	0.0069	-0.0173	0.8844	0.0139	-0.0362
Debenedetti2022Light_XCiT-L12	0.9440	0.0041	-0.0154	0.8658	0.0162	-0.0233
Debenedetti2022Light_XCiT-M12	0.9435	0.0049	-0.0164	0.8641	0.0194	-0.0233
Debenedetti2022Light_XCiT-S12	0.9428	0.0048	-0.0239	0.8605	0.0198	-0.0357
Engstrom2019Robustness	0.9231	0.0101	-0.0203	0.8298	0.0263	-0.0406
Gowal2020Uncovering_28_10_extra	0.9476	0.0072	-0.0149	0.8803	0.0221	-0.0214
Gowal2020Uncovering_34_20	0.9323	0.0081	-0.0156	0.8579	0.0206	-0.0331
Gowal2020Uncovering_70_16	0.9357	0.0055	-0.0144	0.8614	0.0155	-0.0279
Gowal2020Uncovering_70_16_extra	0.9535	0.0050	-0.0129	0.8898	0.0196	-0.0137
Gowal2021Improving_28_10_ddpm_100m	0.9528	0.0073	-0.0192	0.8949	0.0177	-0.0347
Gowal2021Improving_70_16_ddpm_100m	0.9551	0.0061	-0.0147	0.9024	0.0169	-0.0265
Gowal2021Improving_R18_ddpm_100m	0.9397	0.0065	-0.0166	0.8726	0.0117	-0.0387
Hendrycks2019Using	0.9340	0.0057	-0.0250	0.8519	0.0215	-0.0447
Huang2020Self	0.9294	0.0075	-0.0139	0.8451	0.0270	-0.0201
Huang2021Exploring	0.9471	0.0051	-0.0091	0.8779	0.0227	-0.0091
Huang2021Exploring_ema	0.9519	0.0054	-0.0103	0.8856	0.0226	-0.0103
Huang2022Revisiting_WRN-A4	0.9537	0.0052	-0.0101	0.8919	0.0167	-0.0127
Jia2022LAS-AT_34_10	0.9327	0.0102	-0.0173	0.8524	0.0262	-0.0277
Jia2022LAS-AT_70_16	0.9372	0.0070	-0.0178	0.8585	0.0229	-0.0262
Pang2020Boosting	0.9319	0.0181	-0.0349	0.8526	0.0430	-0.0611
Pang2022Robustness_WRN28_10	0.9449	0.0095	-0.0199	0.8708	0.0229	-0.0325
Pang2022Robustness_WRN70_16	0.9524	0.0056	-0.0220	0.8828	0.0225	-0.0331
Rade2021Helper_ddpm	0.9444	0.0068	-0.0195	0.8715	0.0188	-0.0317
Rade2021Helper_extra	0.9481	0.0057	-0.0158	0.8778	0.0186	-0.0247
Rade2021Helper_R18_ddpm	0.9339	0.0098	-0.0203	0.8564	0.0220	-0.0407
Rade2021Helper_R18_extra	0.9347	0.0095	-0.0183	0.8578	0.0220	-0.0370
Rebuffi2021Fixing_106_16_cutmix_ddpm	0.9528	0.0063	-0.0196	0.8907	0.0226	-0.0295
Rebuffi2021Fixing_28_10_cutmix_ddpm	0.9432	0.0088	-0.0193	0.8713	0.0262	-0.0286
Rebuffi2021Fixing_70_16_cutmix_ddpm	0.9522	0.0071	-0.0204	0.8899	0.0208	-0.0311
Rebuffi2021Fixing_70_16_cutmix_extra	0.9544	0.0051	-0.0169	0.8973	0.0191	-0.0282
Rebuffi2021Fixing_R18_ddpm	0.9337	0.0078	-0.0140	0.8584	0.0242	-0.0270
Rice2020Overfitting	0.9327	0.0091	-0.0273	0.8532	0.0199	-0.0548
Sehwag2020Hydra	0.9383	0.0055	-0.0123	0.8561	0.0262	-0.0126
Sehwag2021Proxy	0.9428	0.0073	-0.0150	0.8813	0.0166	-0.0348
Sehwag2021Proxy_R18	0.9369	0.0091	-0.0209	0.8598	0.0240	-0.0364
Sehwag2021Proxy_ResNest152	0.9466	0.0062	-0.0138	0.8864	0.0151	-0.0308
Sitawarin2020Improving	0.9237	0.0084	-0.0190	0.8345	0.0243	-0.0428
Sridhar2021Robust	0.9428	0.0063	-0.0112	0.8722	0.0221	-0.0171
Sridhar2021Robust_34_15	0.9413	0.0067	-0.0065	0.8740	0.0208	-0.0089
Wang2020Improving	0.9387	0.0119	-0.0216	0.8624	0.0362	-0.0303
Wang2023Better_WRN-28-10	0.9563	0.0067	-0.0149	0.8990	0.0166	-0.0248
Wang2023Better_WRN-70-16	0.9620	0.0049	-0.0149	0.9104	0.0165	-0.0239
Wong2020Fast	0.9093	0.0128	-0.0282	0.8068	0.0301	-0.0608
Wu2020Adversarial	0.9336	0.0068	-0.0161	0.8540	0.0225	-0.0267
Wu2020Adversarial_extra	0.9403	0.0073	-0.0139	0.8681	0.0228	-0.0230
Zhang2019Theoretically	0.9298	0.0061	-0.0190	0.8381	0.0238	-0.0289
Zhang2019You	0.9107	0.0073	-0.0196	0.8061	0.0203	-0.0506
Zhang2020Attacks	0.9285	0.0082	-0.0167	0.8438	0.0257	-0.0267
Zhang2020Geometry	0.9402	0.0187	-0.0307	0.8629	0.0404	-0.0514

Table F.6: Complete list of $\mathcal{R}$ and its bounds under $(\mathcal{W}_2, l_\infty)$ threat model based on ReDLR loss.

	δ=1/510			δ=1/255		
Network	$\mathcal{R}$	$\Delta\mathcal{R}^u$	$\Delta\mathcal{R}^l$	$\mathcal{R}$	$\Delta\mathcal{R}^u$	$\Delta\mathcal{R}^l$
Addepalli2021Towards_RN18	0.9762	0.0080	-0.1301	0.9533	0.0202	-0.1902
Addepalli2021Towards_WRN34	0.9774	0.0102	-0.0849	0.9590	0.0197	-0.1257
Addepalli2022Efficient_RN18	0.9740	0.0088	-0.1012	0.9534	0.0195	-0.1480
Addepalli2022Efficient_WRN_34_10	0.9749	0.0124	-0.0645	0.9563	0.0231	-0.0942
Andriushchenko2020Understanding	0.9679	0.0140	-0.0969	0.9438	0.0294	-0.1357
Carmon2019Unlabeled	0.9775	0.0079	-0.0803	0.9584	0.0178	-0.1162
Chen2020Adversarial	0.9639	0.0182	-0.0351	0.9371	0.0351	-0.0485
Chen2020Efficient	0.9713	0.0147	-0.0742	0.9495	0.0281	-0.1073
Chen2021LTD_WRN34_10	0.9758	0.0140	-0.0622	0.9578	0.0236	-0.0960
Chen2021LTD_WRN34_20	0.9780	0.0078	-0.1077	0.9635	0.0124	-0.1615
Cui2020Learnable_34_10	0.9694	0.0121	-0.0663	0.9462	0.0228	-0.0838
Cui2020Learnable_34_20	0.9710	0.0132	-0.0722	0.9492	0.0231	-0.0982
Dai2021Parameterizing	0.9814	0.0082	-0.0825	0.9654	0.0164	-0.1206
Debenedetti2022Light_XCiT-L12	0.9760	0.0076	-0.0730	0.9579	0.0181	-0.1063
Debenedetti2022Light_XCiT-M12	0.9781	0.0127	-0.0477	0.9574	0.0280	-0.0682
Debenedetti2022Light_XCiT-S12	0.9795	0.0078	-0.0911	0.9620	0.0161	-0.1370
Engstrom2019Robustness	0.9746	0.0086	-0.1117	0.9550	0.0174	-0.1600
Gowal2020Uncovering_28_10_extra	0.9791	0.0075	-0.0813	0.9623	0.0153	-0.1213
Gowal2020Uncovering_34_20	0.9748	0.0110	-0.0786	0.9546	0.0220	-0.1166
Gowal2020Uncovering_70_16	0.9722	0.0162	-0.0583	0.9552	0.0232	-0.0906
Gowal2020Uncovering_70_16_extra	0.9801	0.0070	-0.0786	0.9639	0.0136	-0.1145
Gowal2021Improving_28_10_ddpm_100m	0.9814	0.0070	-0.0852	0.9690	0.0119	-0.1281
Gowal2021Improving_70_16_ddpm_100m	0.9838	0.0072	-0.0778	0.9726	0.0114	-0.1150
Gowal2021Improving_R18_ddpm_100m	0.9797	0.0069	-0.1049	0.9626	0.0160	-0.1520
Hendrycks2019Using	0.9778	0.0090	-0.0963	0.9582	0.0183	-0.1398
Huang2020Self	0.9714	0.0117	-0.0764	0.9478	0.0252	-0.1082
Huang2021Exploring	0.9770	0.0109	-0.0556	0.9593	0.0208	-0.0803
Huang2021Exploring_ema	0.9792	0.0091	-0.0625	0.9623	0.0189	-0.0923
Huang2022Revisiting_WRN-A4	0.9810	0.0088	-0.0521	0.9646	0.0181	-0.0766
Jia2022LAS-AT_34_10	0.9762	0.0139	-0.0743	0.9575	0.0232	-0.1118
Jia2022LAS-AT_70_16	0.9751	0.0138	-0.0707	0.9583	0.0219	-0.1091
Pang2020Boosting	0.9745	0.0093	-0.0244	0.9528	0.0195	-0.0330
Pang2022Robustness_WRN28_10	0.9823	0.0055	-0.1311	0.9707	0.0095	-0.1988
Pang2022Robustness_WRN70_16	0.9836	0.0078	-0.0882	0.9711	0.0139	-0.1378
Rade2021Helper_ddpm	0.9808	0.0068	-0.0936	0.9672	0.0124	-0.1430
Rade2021Helper_extra	0.9809	0.0066	-0.0846	0.9674	0.0130	-0.1298
Rade2021Helper_R18_ddpm	0.9778	0.0056	-0.1266	0.9627	0.0098	-0.1904
Rade2021Helper_R18_extra	0.9784	0.0127	-0.0720	0.9587	0.0244	-0.1073
Rebuffi2021Fixing_106_16_cutmix_ddpm	0.9829	0.0046	-0.0787	0.9681	0.0130	-0.1163
Rebuffi2021Fixing_28_10_cutmix_ddpm	0.9816	0.0050	-0.1001	0.9651	0.0115	-0.1436
Rebuffi2021Fixing_70_16_cutmix_ddpm	0.9824	0.0072	-0.0694	0.9680	0.0141	-0.1035
Rebuffi2021Fixing_70_16_cutmix_extra	0.9809	0.0056	-0.0718	0.9661	0.0126	-0.1058
Rebuffi2021Fixing_R18_ddpm	0.9758	0.0104	-0.0855	0.9564	0.0194	-0.1276
Rice2020Overfitting	0.9773	0.0082	-0.1061	0.9593	0.0159	-0.1547
Sehwag2020Hydra	0.9758	0.0096	-0.0805	0.9577	0.0192	-0.1201
Sehwag2021Proxy	0.9818	0.0073	-0.0831	0.9657	0.0162	-0.1202
Sehwag2021Proxy_R18	0.9795	0.0071	-0.0964	0.9609	0.0154	-0.1370
Sehwag2021Proxy_ResNest152	0.9805	0.0144	-0.0361	0.9667	0.0243	-0.0573
Sitawarin2020Improving	0.9747	0.0088	-0.1054	0.9511	0.0204	-0.1505
Sridhar2021Robust	0.9766	0.0087	-0.0832	0.9594	0.0160	-0.1240
Sridhar2021Robust_34_15	0.9749	0.0105	-0.0602	0.9537	0.0221	-0.0858
Wang2020Improving	0.9738	0.0195	-0.0134	0.9552	0.0341	-0.0185
Wang2023Better_WRN-28-10	0.9840	0.0054	-0.0767	0.9692	0.0129	-0.1120
Wang2023Better_WRN-70-16	0.9835	0.0049	-0.0747	0.9727	0.0074	-0.1121
Wong2020Fast	0.9674	0.0163	-0.0901	0.9448	0.0268	-0.1281
Wu2020Adversarial	0.9776	0.0089	-0.0837	0.9577	0.0221	-0.1215
Wu2020Adversarial_extra	0.9778	0.0077	-0.0963	0.9588	0.0158	-0.1419
Zhang2019Theoretically	0.9774	0.0132	-0.0670	0.9565	0.0277	-0.1003
Zhang2019You	0.9720	0.0135	-0.1073	0.9502	0.0247	-0.1580
Zhang2020Attacks	0.9724	0.0208	-0.0225	0.9504	0.0386	-0.0312
Zhang2020Geometry	0.9875	0.0035	-0.1267	0.9777	0.0068	-0.1838