We thank the reviewers for the valuable feedback. We will reflect minor errors instantly and try suggestions in the future. We want to address concerns and to clarify misconceptions. In this rebuttal, CAP denotes Wong et al. **[R2 R3] Under ℓ_2-norm, IBP does not work well enough.** In IBP, the authors didn't mention the certifiable training for ℓ_2-norm. In CROWN-IBP, however, the authors explained that IBP can be applied to ℓ_2-norm. We run the code for IBP provided by the authors of CROWN-IBP on CIFAR-10 under ℓ_2-setting with ℓ_2-norm $36/255$, where we considered a wide range of parameters including those suggested by CROWN-IBP and our range of settings described in Section C. IBP achieved the verification (standard) accuracy of 22.6-23.0% (31.3-33.5%) which was inferior to CAP = 50.29% (60.14%) and to LMT = 37.20% (56.49%). It also implies LMT-bound is much tighter than IBP-bound under ℓ_2-norm. Thus, we compared our method to LMT and CAP rather than IBP under ℓ_2-case in the main text. **[R2 R3] On the novelty.** BCP was carefully designed in a layer-wise manner (Fig1) to obtain the tighter outer bound. One may simply use IBP to get the additional box constraint in (11), but this box constraint is redundant because it is much looser than the ℓ_2-constraint in (11). BCP is designed to provide a nonredundant box constraint in (11) to tighten the bound. We observed the tightness in Fig2 and Fig3. As a result, BCP outperforms both LMT and IBP in a large margin (>12-28%p) under ℓ_2-norm. For further intuition behind the layerwise design of BCP, refer to Section B.1. **[R2 R3] BCP outperforms the others with a meaningful margin.** In Tab 1, the evaluation results at a single ϵ_{eval} seem to be a marginal improvement compared to CAP. However, when considering a wide range of ϵ_{eval} in Fig4 (and in FigS3), BCP outperforms CAP by 3.7-5.6%p in standard accuracy on CIFAR-10. For $\epsilon_{eval} > 36/255$, BCP outperforms CAP in a large margin. For example, when evaluating at $72/255$, BCP (34.2%) defeats CAP (23.9%) by 10.3%p. Moreover, only BCP can achieve a meaningful verification accuracy on Tiny ImageNet, while others cannot. **[R1 R2] Fig2 illustrates how BCP can tighten the outer region by introducing the box constraint.** In Fig2 (a)-(c), we can easily visualize the high-dimensional ellipsoid $h_K(\mathbb{B}_2^{(K-1)}) \subset \mathbb{R}^c$ in 2D plane with ζ_y- and $\zeta_{m'}$-axes (line 211-212) by projection. However, a high-dimensional parallelogram $h_K(\mathbb{B}_\infty^{(K-1)})$ for $c > 2$ is hard to visualize in the 2D plane. Thus, we use the red lines in Fig2 to indicate that the projection of the outer region $h_K(\mathbb{B}_2^{(K-1)} \cap \mathbb{B}_\infty^{(K-1)})$ must lie above the red line and inside the ellipsoid. Based on (9), we used the verification boundary ($\zeta_y - \zeta_{m'} \geq \zeta_y^* - \zeta_{m'}^*$), where the red line is obtained from the solution $\zeta_y^*, \zeta_{m'}^*$ of (9) for $\hat{z}(\mathbb{B}(\boldsymbol{x})) = h_K(\mathbb{B}_2^{(K-1)} \cap \mathbb{B}_\infty^{(K-1)})$, and the blue line is for $\hat{z}(\mathbb{B}(\boldsymbol{x})) = h_K(\mathbb{B}_2^{(K-1)})$. Fig2 (d) explicitly illustrates the ellipsoid and the parallelogram with the verification boundary for a toy binary classification ($c = 2$). Fig2 shows typical examples for which the verification succeeds with BCP but fails without BCP (R2-8-3). On comparing the gap between BCP and other baselines, the visualization of the logit space can be inappropriate because it is not possible to directly compare the logits from the models obtained by different certifiable training methods. As verification methods, comparing only verification phase using one trained model can be unfair because the performance highly depends on which method the model is trained with. **[R2 R3] We computed the radius $\rho^{(k)}$ with the layerwise Lipschitz constants $L^{(i)}$'s.** We computed $L^{(i)}$ for each i-th operation (BCP.py line 254,286), and then computed $\rho^{(k)} = \epsilon \Pi_{i=1}^{k} L^{(i)}$, multiplying the expansion rate (=Lipshitz constant) through each layer (line 114-118,137-139, Section B.2). We are sorry to make R2 confused. Lipschitz constant usually refer to the global Lipschitz constant (gL) rather than a local one (lL) when not specified. We used the gL for the efficient computation (line 127-128). For the layerwise Lipschitz constant, when the layerwise operation h_k is linear, the maximum eigen-value of weight matrix corresponds to both gL and lL (gL = lL). **[R3] On the stabilization of $\rho^{(K-1)}$.** In FigS1 (top), the Lipschitz constant keeps increasing through standard training. To stabilize the Lipschitz constant, we applied a commonly-used scheduling scheme on ϵ and λ during the BCP training (line 195-200). In FigS1 (bottom), $L^{(-1)} = \Pi_{i=1}^{K-1} L^{(i)}$ is about 10 after training with BCP, and it is multiplied by ϵ to provide $\rho^{(K-1)} = \epsilon L^{(-1)}$. Thus, $\rho^{(K-1)}$ is about 10ϵ. In the code, we initialize $\rho^{(0)} = \epsilon$ (BCP.py line 214) and update the radius by $\rho^{(i+1)} = \rho^{(i)} * L^{(i+1)}$ (`r = r*p`) for each layer as implemented in BCP.py line 236-243,270,300. We also tried BN, but it is not effective to improve robustness. Moreover, there is a paper named "Batch Normalization is a Cause of Adversarial Vulnerability". **[R3] On the implementation of BCP.** Our main focus is on ℓ_2-certifiable training. In the main text, we describe the BCP algorithm and provide the results for ℓ_2-case, so, in the code, we set `args.linfty=False` as default. However, BCP can be applied to ℓ_p-norm for any $p \in (0, \infty]$ (line 154-155). In appendix, we presented the results for ℓ_∞-norm in Tab S2, and the implementation is available by setting `args.linfty=True` in the code (BCP.py line 216-217). Therefore, the description of the algorithm, including the ℓ_∞-case (line 154-159), is consistent with the implementation. Moreover, for reproducibility, we provided how to run the code in one-line command in README.md and explained details of the hyper-parameters in Section C, including the scheduling on ϵ and λ. For reference, in the ℓ_∞-case, BCP is not IBP but a generalized version of IBP because BCP still uses the ℓ_2-bound propagation. **[R3] $\mathbf{c}$ and $\mathbf{z}^{(K-1)}$ have the same shape.** The notation $\mathbf{W}_i$ is for the i-th row vector of the matrix $\mathbf{W}$ (line 144). Let $\mathbf{W}^{(K)} \in \mathbb{R}^{m \times n}$, then $\mathbf{W}_i^{(K)}$ is an n-dimensional row vector, and $\mathbf{z}^{(K-1)}$ is an n-dimensional column vector. Therefore, since $\mathbf{c}^T = \mathbf{W}_1^{(K)} - \mathbf{W}_0^{(K)}$, $\mathbf{c}$ has the same dimension to $\mathbf{z}^{(K-1)}$. **[R1] Trade-off between efficiency and accuracy.** BCP is much faster than CAP (Tab1) as well as achieving the better performance in a wide range of ϵ_{val} (Fig4). BCP can outperform LMT and IBP in a large margin with affordable computational overhead.