Table A: Verified error and running time of different frameworks on CIFAR. The model structure is ConvSmall from [28].

	Regularly trained (ϵ=2/255)			LiRPA trained (ϵ=8/255)			
	[35]	[37]	Ours	[45]	[28]	[50]	Ours
Verified error	63.85%	63.85%	63.85%	74.50%	75.30%	71.59%	**71.57%**
Time (min)	223.37	89.45	**14.37**	248.74	105.31	43.45	**28.11**

Table B: Certified training on Downscaled ImageNet. We use WideResNet with $\epsilon = \frac{1}{255}$.

Dataset	Method	Clean	PGD	Verified
ImageNet (64×64)	IBP [9]	84.04%	90.88%	93.87%
	Ours	**83.77%**	**89.74%**	**91.27%**

We thank all reviewers for their encouraging and helpful comments. We will fix all typos. We answer questions below:

R1. Comparison to other LiRPA implementations on feed-forward NNs. We categorize existing implementations into 2 kinds: (1) for verification only (typically implemented on CPUs, including DeepZ[35], and DeepPoly[37]) (2) for training certified defense (typically using more efficient, yet weaker or approximated bounds: convex outer adversarial polytope[45], DiffAI[28], IBP[9] and CROWN-IBP[50]). For category (1) we compare bound tightness (verified error given a ℓ_∞ norm ϵ) and time to verify the test set; for category (2) we compare verified accuracy *after training* and training time. Results are presented in Table A. Following convex relaxation theory[32], our bound has the same strength as CROWN[49]/DeepPoly[37], but we use GPU acceleration from PyTorch. Our contribution is not to improve tightness of LiRPA bounds, but the first framework that generalizes to general computational graphs in an automatic manner. Results on a large dataset. We conduct additional experiments on downscaled (64×64, 1,000 classes) ImageNet in Table B. With the help of loss fusion, for the first time, we demonstrate LiRPA based certified defense on Downscaled ImageNet and outperform IBP[9], the only method that can scale to this setting previously.

R2. High-order bounds. Admittedly, we only implement the linear relaxations of CROWN and currently do not handle CROWN-quad. In CROWN[50], the quadratic bound is only applied to 2-layer networks and is hard to extend to multiple layers, as when propagating a quadratic bound to the 3rd layer it becomes quadratic (x^4) due to correlations between two quadratic terms ("order explosion"). This makes the concretization problem (in Sec. 3.2) hard to solve. We plan to study high-order bounds on general graphs as our future work. Limitations on linear input constraints We can handle any input constraint $\mathbf{X} \in \mathbb{S}$ as long as the linear "concretization" problem can be efficiently solved (Sec 3.2). When $\mathbb{S}$ is an ℓ_∞ ball, it is linear; but it is non-linear when $\mathbb{S}$ is an ℓ_2 ball (but $\mathbb{S}$ is still convex so easy to solve). We can even handle non-linear, non-convex case. For example, when $\mathbb{S}$ is a sparse perturbation (non-linear and non-convex), e.g, $\mathbb{S} = \{\|\mathbf{X} - \mathbf{X}_0\|_0 \leq k, 0 \leq \mathbf{X} \leq 1\}$, the solution is: $\underline{\mathbf{h}}_{o,j} = \mathbf{A}_{j,:}\mathbf{X} - \sum_{\text{topk}}(\mathbf{A}^+_{j,:} * \mathbf{X}) + \sum_{\text{topk}}(\mathbf{A}^-_{j,:} * (1 - \mathbf{X}))$, $\overline{\mathbf{h}}_{o,j} = \mathbf{A}_{j,:}\mathbf{X} - \sum_{\text{topk}}(\mathbf{A}^-_{j,:} * \mathbf{X}) + \sum_{\text{topk}}(\mathbf{A}^+_{j,:} * (1 - \mathbf{X}))$. where $*$ denotes element-wise multiplication, $_{\text{topk}}$ denotes the indices of largest k elements. We show preliminary results on LiRPA based ℓ_0 norm certified defense in Table D. The input constraints can be even more generalized when it is produced by some parameterized neural network, where we can combine this network with the classifier to verify the whole computation. We will also discuss these extensions. Fairness of comparison to IBP. We compare to IBP in training experiments because (1) IBP is currently the only feasible method for training large-scale certified defense on irregular networks; (2) Even on smaller networks, IBP outperforms many tighter bounds *after training* (see Table 4 in [9]) and IBP based method[50] is currently the state-of-the-art. Performance on NLP benchmarks We discussed this issue in Appendix C.2 (L.545-559). Huang et al. build a convex hull on the input layer, where each instance in the convex hull has only one position perturbed but the perturbation is enlarged to δ times. They use CNN and after the first layer, the convex hull is converted into interval bounds. But this requires the first layer to be an affine layer and different positions have interactions, which is not the case in Transformer/LSTM. E.g., the linear layer before the self-attention in Transformer (to obtain query/key/value) is applied to different positions independently. In this case, Huang et al.'s method gives a $(\delta - 1)$-time over-estimation, compared to assuming all the positions are independently replaced as in Jia et al. [17]. Therefore, we adopt the IBP based method in [17] whose results are not affected by δ. To avoid bugs, we test our code base carefully with continuous integration (Travis CI) and we compare our bounds with references from other libraries when possible (e.g., on feed-forward NNs). Bayesian Neural Networks We greatly appreciate the reviewer on pointing out this potential application and we will discuss it in related works and further study it as our future work.

Table C: Multi-layer NLP models with $\delta_{\text{train}} = 6$.

Model	Method	Verified Test Accuracy (%) $\delta = 0$	$\delta = 1$	$\delta = 3$	$\delta = 6$
2-Layer Transformer	IBP	77.5	75.4	75.4	75.4
	Ours	78.1	77.2	77.2	77.2
4-Layer Transformer	IBP	78.4	76.0	76.0	76.0
	Ours	78.6	77.4	77.4	77.3
2-Layer LSTM	IBP	81.4	78.2	78.2	78.2
	Ours	81.4	78.4	78.4	78.4

Table D: Results of ℓ_0 norm certified defense on a simple MLP model.

Method	Metric	k = 1	k = 4	k = 10
IBP	Verfied err.	5.79%	10.06%	25.15%
	Clean err.	1.57%	2.24%	4.84%
Ours	Verfied err.	5.71%	9.59%	24.67%
	Clean err.	1.62%	2.21%	4.95%

R3. Multi-layer NLP models. Our method natively support multi-layer LSTMs and Transformers. We include additional experiments in Table C. Our framework provides competitive results on these networks. Nature error rates Yes, it is the error rate on clean test set (we will fix the terms used). The accuracy of models is relatively low compared to normally trained models, but this is common in certified defense - our reported error rates are similar to or better than those in state-of-the-art (e.g., [45,9,50] reported *clean test errors* of 71.33%, 50.51% and 54.02% on CIFAR-10 with $\epsilon = \frac{8}{255}$, respectively; ours are around 53%). Currently all LiRPA based certified defenses have this trade-off between robustness and accuracy. We leave further development on improving the clean accuracy of certified training as a future work.