Efficiently Computing Local Lipschitz Constants of Neural Networks via Bound Propagation

Zhouxing Shi[1], Yihan Wang[1], Huan Zhang[2], Zico Kolter[2,3], Cho-Jui Hsieh[1]

[1]University of California Los Angeles [2]Carnegie Mellon University [3]Bosch Center for AI

Local Lipschitz Constants

- ℓ_∞ **local Lipschitz constant** for a neural network f, within a local region $\mathcal{X}$:

$$L(f,\mathcal{X}) = \sup_{\mathbf{x}_1,\mathbf{x}_2\in\mathcal{X},\ \mathbf{x}_1\neq\mathbf{x}_2} \frac{\|f(\mathbf{x}_1)-f(\mathbf{x}_2)\|_\infty}{\|\mathbf{x}_1-\mathbf{x}_2\|_\infty} = \sup_{\mathbf{x}\in\mathcal{X},\ \mathbf{J}(\mathbf{x})\in\partial f(\mathbf{x})} \|\mathbf{J}(\mathbf{x})\|_\infty, \qquad \text{where } \mathcal{X}=B_\infty(\mathbf{x}_0,\epsilon).$$

- Connected to many properties of neural networks, such as robustness, fairness, generalization, etc.
- We aim to **efficiently compute sound and tight upper bounds** of $L(f,\mathcal{X})$.

Formulation with a Computational Graph

$$\mathbf{J}_i(\mathbf{x}) \in \frac{\partial f(\mathbf{x})}{\partial h_{i-1}(\mathbf{x})}\Big\{\mathbf{J}_{i+1}(\mathbf{x})\Delta_i(\mathbf{x})\mathbf{W}_i : \mathbf{J}_{i+1}(\mathbf{x}) \in \frac{\partial f(\mathbf{x})}{\partial h_i(\mathbf{x})},\ \Delta_i(\mathbf{x}) \in \partial\sigma(z_i(\mathbf{x}))\Big\}.$$

We formulate the computation for the Clarke Jacobian as a *backward computational graph* augmented to the forward graph for the original neural network computation.

Benefits Our formulation enables:

- Using **linear bound propagation** techniques that originally supports neural network verification on general computational graphs to efficiently and tightly upper bound $\|\mathbf{J}_1(\mathbf{x})\|_\infty$.
- Utilizing recent and future progress in bound propagation-based verification (e.g., branch-and-bound), to further enhance the computation for local Lipschitz constants.

Empirical improvements

- **Scalability and Efficiency**: we support models with 4 convolutional layers on image datasets, compared to only small and shallow MLPs in previous works.
- **Tightness**: on relatively larger models, we obtain up to 20X tighter bounds compared to RecurJac.
- An application for monotonicity analysis.

Linear Bound Propagation

Principles

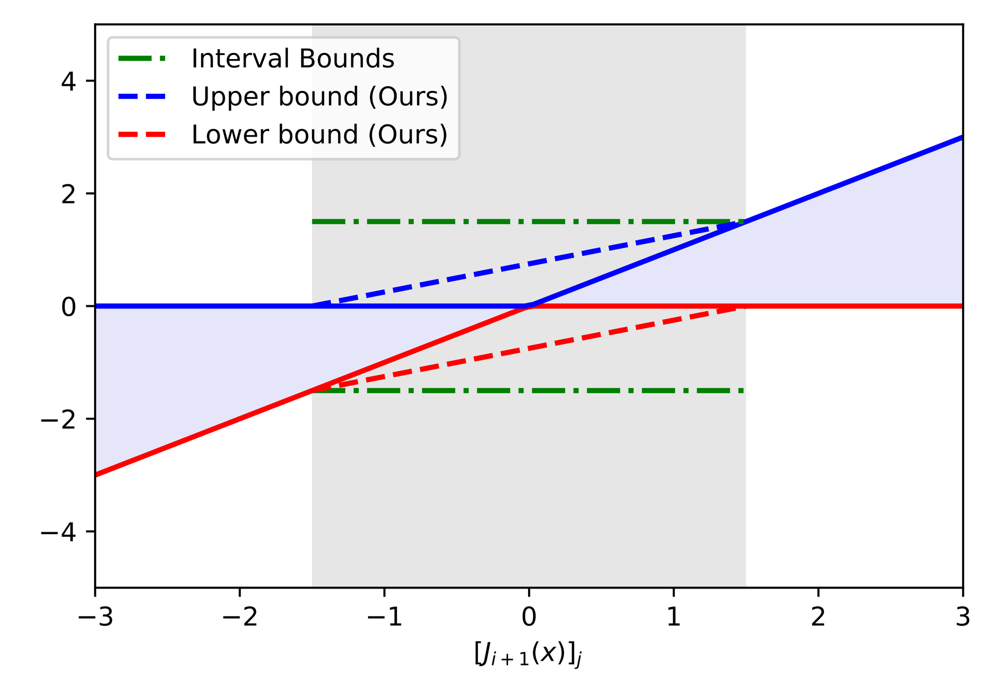
- Given a specified perturbation on the input, linear bound propagation computes the output bounds of a computational graph, by **propagating the linear relationship** between layers, with all the nonlinear operators relaxed by **linear relaxation**.
- A library for general computational graphs: auto_LiRPA (https://github.com/Verified-Intelligence/auto_LiRPA)

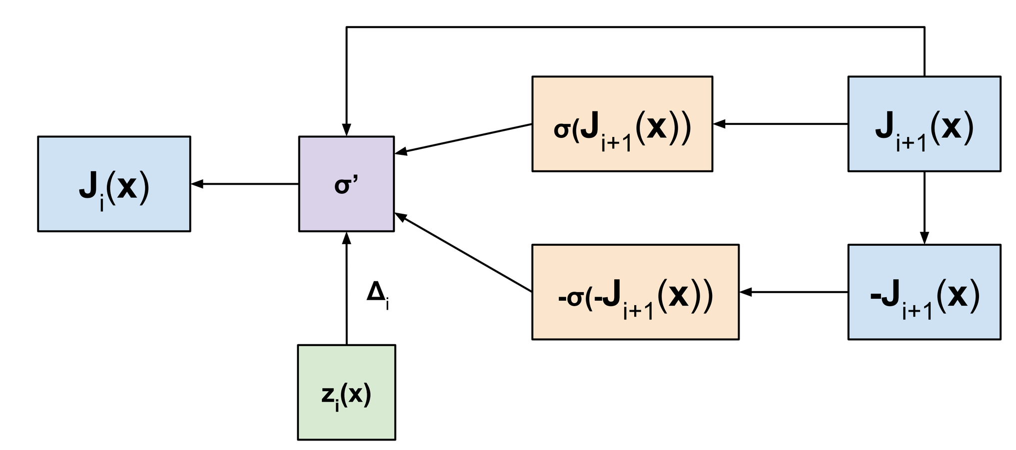
Proposed Linear Relaxation for Clarke Gradients

- We propose a tight linear relaxation for $[\mathbf{J}_{i+1}(\mathbf{x})\Delta_i(\mathrm{v})]_j$ required in bounding $\|\mathbf{J}(\mathbf{x})\|_\infty$.

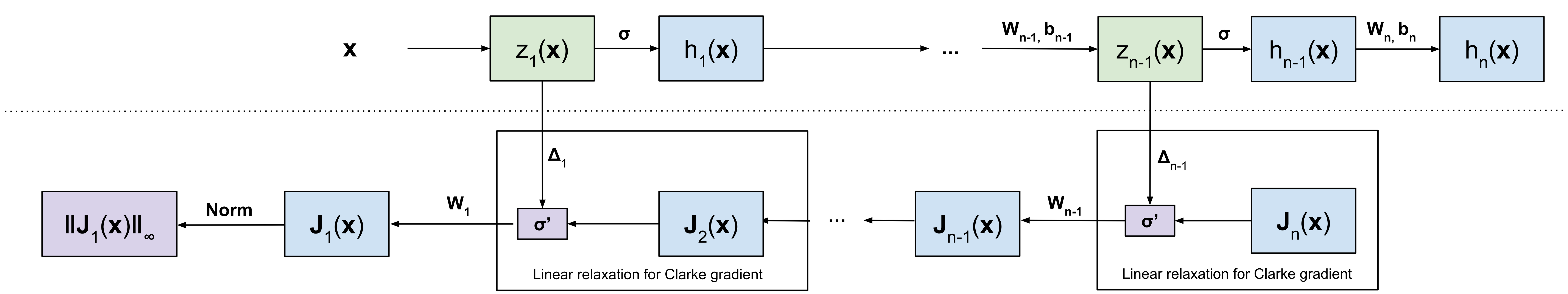
$$\underline{s}[\mathbf{J}_{i+1}(\mathbf{x})]_j + \underline{t} \le [\mathbf{J}_{i+1}(\mathbf{x})]_j \cdot [\Delta_i(\mathbf{x})]_{jj} \le \overline{s}[\mathbf{J}_{i+1}(\mathbf{x})]_j + \overline{t} \quad \text{for } [\mathbf{J}_{i+1}(\mathbf{x})]_j \in \left[[\mathbf{L}_{i+1}]_j, [\mathbf{U}_{i+1}]_j\right], [\Delta_i(\mathbf{x})]_{jj} \in [0,1].$$

- The exact upper bound and lower bound is a ReLU and inverted ReLU respectively.
- Thus we propose a closed-form linear relaxation implemented as relaxing ReLU.
- This relaxation is the provably **optimal linear relaxation** and much tighter than the interval relaxation in prior works (such as RecurJac).





Forward graph (original neural network computation)



Backward graph (norm of Clarke Jacobian)

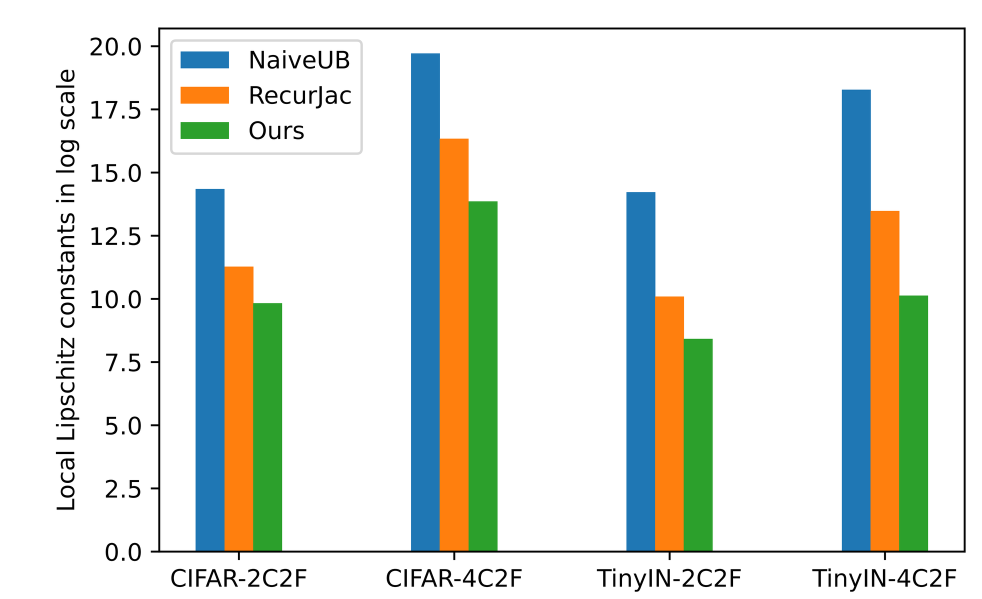
Experiments

Results on MNIST

Method	3-layer MLP Value	3-layer MLP Runtime	CNN-2C2F Value	CNN-2C2F Runtime
NaiveUB	3,257.16	0.00	80,239.62	0.00
LipMIP	14,218.99*	120.51	-	-
LipBaB	947.69	62.77	-	-
RecurJac	1,091.31	0.22	12,514.55	115.43
Ours (w/o BaB)	**688.15**	**4.95**	**5,473.03**	**8.21**
Ours	**397.25**	52.23	**5,458.84**	60.04

- Our method is **more efficient** than LipMIP and LipBaB, while computing **tighter** results than RecurJac.
- LipMIP (solving MIP) and LipBaB (interval bounds + BaB) are costly and only feasible on small models; RecurJac (a recursive algorithm with relaxation) computes looser bounds.

Results for CNNs on CIFAR and Tiny-ImageNet



(Results are in log scale.)

- Up to **20X tighter bounds** than RecurJac.
- LipMIP and LipBaB cannot handle these larger models.
- The original RecurJac is limited to MLP. Here we re-implement RecurJac's relaxation in our more general and flexible framework to obtain the results for CNNs.

Paper Code auto_LiRPA

NeurIPS 2022, New Orleans, LA

Contact: {zshi, yihanwang, chohsieh}@cs.ucla.edu, huan@huan-zhang.com, zkolter@cs.cmu.edu