Authors' Response for NeurIPS Paper Number 6086. We thank the reviewers for their constructive comments. We first provide a general response common to all the reviewers and then respond to each reviewer's particular comments.

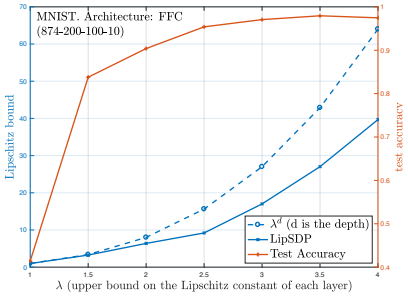
Beyond MNIST and Feed-Forward Fully-Connected (FFC): Since the submission of the paper, we have moved beyond MNIST and FFC neural networks; in particular, we have performed experiments on several CNNs trained on CIFAR-10 and confirmed that (a) our method can scale to larger and more complex networks*, and (b) our bounds are consistently superior to the competing approaches. We will release the code necessary to reproduce these results. In its current status our approach scales to non-trivial tasks such as deep RL for control or robotics [5,6,7]. We can further improve the scalability of our tool by simply improving our hardware (computations were performed on a basic laptop) and our software (exploiting the structure in the SDP, for example).

*We investigated CNNs with two convolutional layers (with three and six filters, no padding, and a stride of one) followed by one linear layer. We remark that unrolling the convolutional operators in a CNN results in a (large) feedforward network. This conversion implicitly handles the padding and stride hyperparameters. Furthermore, since the max function is convex, we can describe the max pooling operation using quadratic constraints without additional assumptions. Therefore, we can directly use `LipSDP` for CNNs. The CNNs we used had approximately 15,000 neurons after unrolling.

Reviewer 1: *Software release:* We have already implemented the software necessary to reproduce our results. All of the code is included in a private GitHub repository that will be made public with the camera-ready version of this paper. Our software links to both interior-point method solvers (Mosek) and first-order method solvers, which are specifically tailored for large-scale cone programs. We tried several large-scale solvers but found the SCS solver, developed by Boyd's group, to be the most stable one.

Reviewer 2: *Lipschitz constant w.r.t. other norms:* In our experiments, we bounded the Lipschitz constant in ℓ_2, then derived the adversarial balls in ℓ_2, and finally converted the ℓ_2 balls to ℓ_∞ balls using the inequality $\|x\|_p \leq n^{\frac{1}{p}-\frac{1}{q}}\|x\|_q$ ($x \in \mathbb{R}^n$). In Remark 1, we discussed how to convert the Lipschitz bound from ℓ_2 norm to other ℓ_p-norms. Apart from this norm conversion, we believe that reformulating the problem (by, e.g., moving to the dual domain or changing the objective function) will enable us to incorporate other ℓ_p norms ($p = 1, 2, \infty$) directly in `LipSDP`. A thorough treatment of this is an important future research direction. *Parallelization:* In principle, we agree with the reviewer that the splitting+parallelization scheme discussed in our paper is not exclusive to our method. However, among the methods that can split the computation, we obtain more accurate bounds as our bounds are more accurate per chunk.

Reviewer 3: *More experiments:* (a) As suggested by the reviewer, we used the training method in [2] to train an FFC NN with architecture 784-200-100-10 on MNIST for various values of λ (a hyper-parameter controlling the Lipschitz constant of each layer). The figure on the right shows that `LipSDP` yields tighter bounds than those guaranteed by [2]. (b) We trained various NNs on MNIST with different initializations. The variance of the Lipschitz bounds found by our method was essentially negligible. For instance, with one hidden layer and 200 neurons, the Lipschitz bounds were around 20 and the standard deviation over 6 trials was <0.01. *Naive lower bound:* In the special case of a purely linear network or under certain positivity assumptions, the "naive lower bound", defined in [4], is equal to the Lipschitz constant. By testing on randomly generated networks, it is our impression that this naive bound, introduced in [4], may be neither a lower bound nor an upper bound on the Lipschitz constant. We will clarify this in the revised manuscript. *Presentation:* We have improved the notation, wording, technical presentation, bibliography, and figures in the revised manuscript. For instance, (a) we clarified that Φ is "the concatenation of activation functions at each layer"; (b) we made the presentation of the assumptions about the activation functions consistent and mentioned that [1] violates the assumption in our paper; (c) we updated the references and cited [1] and [2].



References: [1]: Anil, Cem, James Lucas, and Roger Grosse. "Sorting out Lipschitz function approximation." [2]: Gouk, Henry, et al. "Regularisation of neural networks by enforcing Lipschitz continuity." [3]: Dhillon, Guneet S., et al. "Stochastic activation pruning for robust adversarial defense."[4] Combettes, Patrick L., and J. Pesquet. "Lipschitz Certificates for Neural Network Structures Driven by Averaged Activation Operators." [5]: Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." [6]: Shi, Guanya, et al. "Neural lander: Stable drone landing control using learned dynamics.". [7]: Duan, Yan, et al. "Benchmarking deep reinforcement learning for continuous control."