We thank the authors for their careful reading and helpful comments. We have addressed typos, minor mistakes with related work and other comments provided in the "Clarity" section of the reviews. We first summarize the main critiques raised by the reviewers. Reviewer #1 had concerns about the computational cost of our proposed algorithm and the verifiability of our "general position assumption". Reviewer #2 also raised concerns about the scalability of our approach and noted that much of our algorithmic methodology follows previous threads in neural net verification. Reviewer #3 would like us to stress that our proposed approach only holds for a narrow class of norms that excludes the L2 norm, and requested a discussion on scalability. Reviewer #4 commented that our results appear standard and asked a technical question regarding our general position assumption. They also noted the scalability concerns. We will proceed by addressing the shared concerns of our reviewers, and then respond to the individual suggestions/questions.

On scaling to large networks: As we acknowledge in the paper, the primary drawback to our technique is that MIP approaches have worst-case runtime that is exponential in the number of integer variables. In particular, LipMIP will require one integer variable for each ReLU unit that is "unstable" within a region in addition to at most one integer variable per input dimension; this depends on the network, but also the size of the region we wish to verify. As we scale to larger networks or very high-dimensional datasets, this quickly becomes intractable, though there is hope that the regularization techniques of [1] may provide a modest boost in performance by regularizing towards the stability of neurons. On very large networks, the precision of MIP-solvers may also become an issue. We are working on applying our framework in sub-networks to yield efficient bounds for very large models.

On novelty of results: Reviewers #2 and #4 correctly note that MIP-encoding of neural networks for robustness verification is well-studied. However the extension to Lipschitz-computation is novel and technical: it requires novel machinery to encode the backpropagation procedure as well as the careful theoretical analysis of sections 2 and 3 to prove that the optimization problem is well-founded and returns the correct answer. To the best of our knowledge, there is no extant result relating the Lipschitz constant to the generalized Jacobian, though if such a result exists, we will happily discuss and cite it. Further, our inapproximability result significantly extends what was known about the hardness of estimating Lipschitz constants of ReLU networks.

R#1: "The only thing I feel is missing, is a discussion on how to verify if a network is in general position, and the computational complexity of such a method." Good point. We believe that verifying whether or not a given network is in GP is actually computationally hard (See also related proofs on hardness of checking the Restricted Isometry property). There is an easy solution: an arbitrarily small random perturbation of the weights will create a GP network almost surely. This cannot significantly affect accuracy since floating point arithmetic already introduces a small amount of noise during inference, and the classification loss is typically Lipschitz with respect to network parameters. Further, randomly initialized networks that are trained using SGD will be in general position almost surely if infinitesimal noise is added to each gradient step. We will discuss this in the paper.

R#2: Experimental questions/suggestions and "...are there any other properties of networks that could be included to demonstrate the merit of an exact Lipschitz value so as to strengthen the contribution of the paper?" We will present a more complete version of Table 1 in future iterations. To clarify the caption for table 1: an architecture was fixed for each dataset and many initializations/trainings were evaluated to yield the average and variances. All results were statistically significant. LipLP will always do better than FastLip, at the cost of extra computation time. The relative performance of LipSDP largely depends on the input dimension, with smaller input dimension yielding much tighter results. We also note that our technique may be easily extended to other piecewise-linear activation functions, such as LeakyReLU's or Hard Tanh's. These may be used to verify the conjectures made in [2] regarding the effect activation functions have on Lipschitz constants. We will include these results in future iterations.

R #4: "For theorem 2, I am curious about whether the function w.r.t. ReLU network is subdifferential regular under the general position assumption?" Great question. Subdifferential regularity (SR), roughly speaking, allows the Clarke subdifferential to adopt the nice properties of the subdifferential for convex functions, such as the chain rule. SR is different from our General Position (GP) definition and not very useful for ReLU nets. For example, the authors of [3] (sec 5.1) show that even $-|x|$ is not SR at 0. We can consider the 1-layer neural network $f(x) = \sigma(x+1) - \sigma(2x)$, which locally looks like $-|x|$ at 0 and thus is not subdifferentially regular at 0. Observe that f here is a ReLU network in GP. Hence, our GP results seem to be similar to a notion of stratification, where the strata are the ReLU kernels. We would like to argue that this generality of GP (versus SR) is another example of technical novelty of our work.

[1] Kai Y Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. *arXiv preprint arXiv:1809.03008*, 2018.

[2] Mina Basirat and PETER ROTH. L* relu: Piece-wise linear activation functions for deep fine-grained visual categorization. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1218–1227, 2020.

[3] Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. *(FOCS) Foundations of computational mathematics*, 20(1):119–154, 2020.