Common. We thank the reviewers for their helpful feedback which has strengthened the paper. All reviewers noted the strong empirical results, and thorough comparisons to other neural network verification approaches. We also note we are currently open-sourcing our code, similar to the version shared in the supplementary material.

R1 Thank you for the feedback. Regarding comparison with other upper bounding techniques: we compare empirically with the LP-based approach from [Salman, 2019], which subsumes several relevant upper bounding methods such as [Zhang et al., 2018, Weng et al., 2018, Singh et al., 2019, Gowal et al., 2018]. Further, our solver is general purpose and we can directly incorporate additional constraints, e.g. those from [Ehlers, 2017]. We will revise to clarify.

R2 Thanks for your helpful feedback and pointers. We apologize for oversight in omitting citations to relevant optimization work on first-order SDP solvers. We include a paragraph discussing first-order SDP solvers at the bottom of our response.

Clarification of contributions: We do not develop a general-purpose SDP solver or a novel reformulation of semidefinite programming approaches to neural network verification - rather our focus is on *applying* well-known techniques in first-order SDP algorithms to semidefinite relaxations of neural network verification and developing and evaluating a practical implementation that can leverage hardware accelerators (GPUs/TPUs). This will be clarified in our revised paper (Sections 1, 5.1, and 7). Our specific contributions are:

1. Sections 5.1 and 5.2 derive an eigenvalue optimization formulation for neural network verification. While the ideas here are themselves not novel – e.g. Section 3 of Helmberg and Rendl [3] is very similar (we will clarify this in the revision) – we are the first to apply them to neural network verification. Specifically, we show how for neural networks, subgradient computations can be expressed through autodiff of standard network layers, leading to an implementation with linear memory, runtime, and hardware accelerator compatibility.
2. Section 5.3 presents various tricks for initialization, regularization and step-size schedules which enable the strong empirical results demonstrated in Section 6.
3. Section 6 demonstrates that these applications allow us to verify verification-agnostic networks which were intractable for all previous neural network verification methods, as noted by the reviewers. Compared to second-order methods, first-order methods can achieve matching bounds for small networks, while also scaling to verification of mid-size networks where second-order methods become prohibitively expensive.

Comparisons with [33]: We have a direct comparison on 10 samples: see Appendix C.1 (Figure 4). The bounds achieved by SDP-FO and SDP-IP [33] almost exactly coincide, with the SDP-IP bound slightly tighter on each. This makes us confident that the two methods produce identical bounds, and the differences in Table 1 are due to sampling noise. We'll add this note to the caption.

To be added to Section 7 **First-order SDP solvers:** While interior-point methods are theoretically compelling, the demands of large-scale SDPs motivate first-order solvers. Common themes within this literature include smoothing of nonsmooth objectives [7, 4, 2] and spectral bundle or proximal methods [3, 5, 8]. Many primal-dual algorithms [10, 6, 1] exploit computational advantages of operating in the dual – our dual-based approach to verification naturally inherits these advantages. A full survey is beyond scope here, but we refer interested readers to Tu and Wang [9] for an excellent survey.

Our formulation in Section 5.1 closely follows the eigenvalue optimization formulation from Section 3 of Helmberg and Rendl [3]. We show that within within this formulation, subgradients can be computed using autodiff both easily and efficiently – with linear memory, runtime, and efficient GPU/TPU implementations. While in this work, we show that vanilla subgradient methods are sufficient to achieve practical performance, integrating the ideas from the first-order SDP solver literature mentioned above is a promising candidate for future work, and could potentially allow faster convergence in practice.

R3 Thank you for the feedback. We have fixed the typos and added the paragraph below regarding run-times. Note the main advantage relative to [33] is that the SDP-IP approach is simply intractable for our CNN models due to their size.

To be added to Section 6 **Computational resources:** Using a P100 GPU, maximum runtime for our approach is roughly 15 minutes for all MLP instances, and 3 hours for CNN instances, though most instances are verified sooner. For reference, SDP-IP [33] uses 25 minutes on a 4-core CPU for MLP instances, and is intractable for CNN instances due to $O(n^4)$ memory usage.

[1] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. 2016.
[2] Alexandre d'Aspremont and Noureddine El Karoui. A stochastic smoothing algorithm for semidefinite programming. 2014.
[3] Christoph Helmberg and Franz Rendl. A spectral bundle method for semidefinite programming. 2000.
[4] Guanghui Lan, Zhaosong Lu, and Renato DC Monteiro. Primal-dual first-order methods with $O(1/\epsilon)$... 2011.
[5] Claude Lemaréchal and François Oustry. Nonsmooth algorithms to solve semidefinite programs. 2000.
[6] Renato DC Monteiro. First-and second-order methods for semidefinite programming. 2003.
[7] Yurii Nesterov. Smoothing technique and its applications in semidefinite optimization. 2007.
[8] Neal Parikh and Stephen Boyd. Proximal algorithms. 2014.
[9] Stephen Tu and Jingyan Wang. Practical first order methods for large scale semidefinite programming. 2014.
[10] Zaiwen Wen. First-order methods for semidefinite programming. 2009.