We thank the reviewers for their detailed feedback. We are encouraged by the fact that the reviewers appreciated the practical runtime improvements (**R1**, **R2**, **R4**), the theoretical contributions showing the soundness of our method (**R1**, **R3**, **R4**) and the appropriateness of our method (**R1** "very suitable for the problem", **R2** "a more natural form"). Most reviewers (**R1**, **R2**, **R4**) found the paper to be well-written. We provide clarification of specific concerns raised, which we will incorporate into the final version, along with other suggestions made in the reviews.

**R1 - Justification of the claims of worst case behaviour for PDHG/Toy example** Yes, [15] only provides lower bounds for unconstrained optimization but these techniques readily extend to constrained optimization. This exactly the idea that is used in [16, Section 3.1]. Replicating the argument presented in Section 3.1 for (2) establishes that (2) takes $n-1$ iterations for any saddle point algorithm. We admit that this might not be straightforward for readers not familiar with saddle point algorithm and lower bounds for convex optimization. Consequently, we will add a brief explanation on why [16, Section 3.1] can be used to establish that (2) takes $n-1$ iterations in the Appendix to the final version.

**R1 - It is not clear if it will be better than PDHG in theory, is this correct?** Yes. Although the constants on the two bounds are not directly comparable because they involve different quantities. It would be interesting in the future to see if better worst-case bounds are achievable, maybe with slightly stronger assumptions. As Remark 3 mentions PDHG provably performs poorly on (2) whereas our method solves the problem in one iteration.

**R1 - What is the reason for the much better practical permformance? Is it because the degenerate case does not arise in practice?** Indeed the one reason is that degenerate points do not appear in practice, see line 244-260 and Figure 4b. Also see lines 168-173 for some brief intuition for why does the method perform well in practice. The practical performance of this approach is not something we fully understand yet and we will add it as an open problem.

**R1 - Practical comparison to PDHG** We ran a test with both internal PDHG and mirror-prox code on a few problems and neither got close to optimal after 100,000 iterations. For the final version we will report results with (`https://odlgroup.github.io/odl/math/solvers/nonsmooth/pdhg.html`). Also, note in the paper we compare with SCS, which can be viewed as preconditioned PDHG (`https://arxiv.org/pdf/1811.08937.pdf`).

**R1 - Can the analysis be extended to the variant with backtracking and momentum?** Backtracking line search automatically works as it satisfies line 10 of Algorithm 1. The theory also applies to safeguarded momentum schemes. We will elaborate further on these points in the final version.

**R2 - I don't understand why in appendix C.3,** $(f(s^k,z^k)-f_*)^2 \leq \zeta_2(f(s^k,z^k)-f(s^{k+1},z^{k+1}))$**.** We will update the reasoning in line 505-506 as follows. Define $\zeta_1 = 1$, $\zeta_2 = L\left(D_s\sqrt{2}+2\frac{c+D_s\|W\|_2+D_z\|V\|_2}{\gamma}\right)^2$, and $\zeta_3 = \frac{2\Delta(s^1,\theta^1)}{L}$. Lemma 1 shows that if $\delta_L(s^k,\theta^k) \leq \Delta(s^1,\theta^1)/\zeta_3$ then $\Delta(s^k,\theta^k)^2 \leq \zeta_2\delta_L(s^k,\theta^k)$. Furthermore, $\delta_L(s^k,\theta^k) \leq \psi_0(s^k,\theta^k)-\psi_0(s^{k+1},\theta^{k+1}) = f(s^k,z^k)-f(s^{k+1},z^{k+1})$ by line 10 of Algorithm 1 and the definition of $\psi_0$ respectively. Combining these two inequalities yields $(f(s^k,z^k)-f_*)^2 \leq \zeta_2(f(s^k,z^k)-f(s^{k+1},z^{k+1}))$. Applying Lemma 1 to the latter inequality yields the result.

**R3 - It is difficult to evaluate how realistic the assumptions are and how some of the conditions are to be verified** Assumption 1 and 2 are standard assumptions (smoothness and bounded feasible region) that are usually easy to verify. In Appendix A, we verify assumption 3 for all the examples presented.

**R3 - Escaping local minimizers for nonconvex problem is in general difficult ... maybe the authors want to eludicate more on what makes it possible for this class of problems?** Section 3.1 proves that we can escape local minimizers in the exact case and therefore is a good source of intuition for the inexact case (Section 3.2 and Appendix D). Also, see line 109-124 and Figure 2. The final version will expand on these.

**R3 - Would there be convex reformulations that could similarly harness the structure but not introduce the complications of non-convexity?** Despite this problem being well studied, we are not aware of any such convex reformulations. We sacrifice convexity to replace 'hard' convex constraints with easy bound constraints. We suspect there is no convex reformulation that achieves this property. Similarly, nonconvex reformulations of SDPs sacrifice convexity to eliminate the constraint that $X$ is positive semidefinite. There are also no known convex reformulations of SDPs that eliminate the 'hard' SDP constraint.

**R4 - Elaborate more on the related work on Deep Net verification.** We will add the following text to the end of the introduction, before subsection 1.1. *The convex relaxation was proposed by Ehlers [Ehlers, ATVA 2017] but the high computational cost of solving it with off-the-shelf LP solvers was prohibitive for large instances. A large number of papers such as IBP [Gowal et al., ICCV 2018], DeepPoly[Singh et al, POPL 2019], Crown [Zhang et al., NeurIPS 2018], Neurify [Wang et al., NeurIPS 2018], LP-relaxed-dual [Wong & Kolter, ICML 2018] focused on looser relaxations to allow for fast, closed form solutions of the bound computation problem scaling to larger networks. In parallel, work was done to reformulate the optimization problem to allow the use of better algorithms: DeepVerify [Dvijotham et al., UAI 2018] introduced an unconstrained dual reformulation of the non-convex problem and showed equivalence with the convex relaxation. Proximal [Bunel et al., UAI 2020] performed lagrangian decomposition and used proximal methods to solve the problem faster. The application of our method to network certification follows this research direction of speeding up the computation of network bounds without compromising on tightness.*