We thank the reviewers for their constructive feedback. We will incorporate the suggestions and improve the overall presentation. We first answer common questions followed by individual ones.

Q. *How does k-ReLU and kPoly compare against other methods in terms of precision and timing?*
R. k-ReLU allows for tighter relaxations than [1-5] which are based on 1-ReLU. As Rev. 2 points out, k-ReLU is generic and can be combined with [1-5] to improve their precision. For example, eq. (7) in [2] can be solved using more precise relaxations from k-ReLU. In the paper, *kPoly* combines k-ReLU with *DeepPoly*. Experimentally: (i) we show that *kPoly* is more precise than *DeepPoly*, which in turn is as precise or more precise (Sec. 4.2 in [8] and 3.3.4 in [5]) than [1-5] in our experiments, (ii) we compare the timing of *kPoly* with *DeepPoly* (*DeepPoly* has similar timing to [1-3, 5], [4] is much less precise than all of these). Pure MILP based approaches [6-7] do not scale on our benchmarks and time out. Overall, *kPoly* verifies more robustness properties that other verifiers [1-8].
Q. *On ConvSmall and ConvBig network, kPoly proves less than state-of-the-art [3,4,7]?*
R. The lower provability is due to the networks we used – note that [1-8] on these obtain *lower provability* than *kPoly*. This is because [1-5] are less precise than [8] ([6,7] time out) and [8] is less precise than *kPoly*. We now also ran *kPoly* on more networks from [2] and DiffAI (for 100 images). *kPoly* obtains state-of-the-art provability on both: MNIST ConvBig from DiffAI (93% for $\epsilon = 0.3$) and CIFAR10 ConvSmall from [2] (35% for $\epsilon = 0.03$). We will clarify this point better in the paper.
Q. *What is the complexity of kPoly? Can kPoly handle larger networks and ResNets?*
R. The complexity of *kPoly* depends on the # of refined neurons (which can be tuned) and the complexity of solving the resulting LP formulations. For larger networks, we refine neurons in the fully connected layer which are much fewer compared to the network size. For networks trained to be robust, the LP formulations are easier to solve, see Table 3: *kPoly* runs slower on the smaller ConvSmall (not trained to be robust) than on the (robustly trained) ConvBig which is $\approx$ 10x larger. *kPoly* scales to ResNets: we now ran *kPoly* on the 100K CIFAR10 ResNet from [2] obtaining 26% provability (100 images) for $\epsilon = 0.03$ ([2] obtains $\approx$22% on full test set). We will include this result.
Reviewer 1:
Q. *Where are the results on CIFAR10 in the paper?*
R. The last network in Tables 2 and 3 is a CIFAR10 network (we will fix this typo).
Q. *The authors only report the time limit used for the LP and MILP solvers in their framework.*
R. The timings reported in Table 3 are end-to-end timings for the overall verification procedure.
Reviewer 2:
Q. *Why does* DeepPoly *help?*
R. *DeepPoly* helps in two ways: (i) it computes precise initial interval bounds which help the LP solver refine faster, and (ii) it computes precise octagonal bounds that speed-up *kPoly*.
Q. *In Equation 8, you need to use $\cap$ instead of $\cup$.*
R. No, we first compute convex hulls for each set of k-neurons which are then intersected.
Q. *On line 151, it is not "2^n convex hulls", but one convex hull that requires 2^n inequalities.*
R. No, there are 2^n polyhedra corresponding to either +ve ($x_i \geq 0$) or -ve ($x_i \leq 0$) value of x_i.
Q. *On line 147, each $\mathcal{S}$ is actually a subset of $\mathbb{R}^{hxh}$ and not (as is stated) $\mathbb{R}^h$, right?*
R. No, each $\mathcal{S}$ is defined over the h neurons in the layer so it is a subset of $\mathbb{R}^h$.
Q. *Is the heuristic for selecting the partitioning $\mathcal{I}$ more effective than random partitioning?*
R. We ran *kPoly* with 2-ReLU on the MNIST 6×100 network with $\epsilon = 0.026$. The results show that in most cases, the computed widths of the correct label are smaller with our heuristic.
Reviewer 3:
Q. *What does precision mean? I think it means the % of images certified to be robust.*
R. This is correct, we will fix this.
Q. Can you elaborate on *[6], which looks related?*
R. Yes, while pure MILP formulations [6-7] do not scale on our benchmarks, we believe *kPoly* may benefit from tighter MILP formulations from [6]. We believe this is an interesting future direction.
Q. *Could you explain how the expression in sec. 3.3 leads to the constraints 2-ReLU uses in Fig. 2?*
R. Yes, we will explain this and add a detailed derivation in the appendix.
Q. *Line 107 - Is 1-ReLU equivalent to the LP-based formulation of FastLin [5] and [3]?*
R. No, both Fast-Lin and [3] use a relaxation of 1-ReLU and are thus less precise.
Q. *Figure 3 - Why are there two figures? What are (a) and (b)?*
R. *DeepPoly* uses either of the 2 relaxations of 1-ReLU in Fig. (a) and (b) based on the input bounds.

References 1. Zhang et al. NeurIPS 2018, 2. Wong et al. NeurIPS 2018, 3. Wong et al. ICML 2018, 4. Gowal et al. arxiv 2018, 5. Weng et al. ICML 2018, 6. Anderson et al. IPCO 2019, 7. Tjeng et al. ICLR 2019, 8. Singh et al. ICLR 2019.