We thank the reviewers for their valuable comments which we will incorporate in our work.

Reviewers #1, #3: *What is the scalability of your method? How scalable it is to larger datasets / networks?*

First, we remark the time to compute the bounds (our contribution) is not the main bottleneck, but the propagation of these bounds through the network with a state-of-the-art verifier. E.g., bound computation for 1 image and 1 split typically takes few seconds while bound propagation through a moderately sized network takes ~ 50 seconds (larger networks increase this time). Switching to other verifiers is unlikely to help as they report similar times [8, 15, 21, 25].

We now elaborate more on the scalability of our method. First, we clarify that *runtime* in Table 3 corresponds to the total time it takes to compute the pixel bounds for 1 image and 1 parameter split, averaged over all splits on the test set. We now also computed the same runtime metric for all experiments in Table 1. The results, in seconds, are:

MNIST: 0.6, 1.8, 11, 36 FashionMNIST: 0.1, 1.4 CIFAR-10: 2.5, 2.5, 1.6, 18

All parameters used for experiments in Table 1 are shown in Table 5. Generally, as also indicated by Table 3, we find that a relatively small number of samples n for which an LP solution is found quickly combined with low ϵ-tolerance so that branch-and-bound terminates quickly (e.g., $n = 100, \epsilon = 0.01$), are sufficient to reach high verified robustness (96.5%) fast (1.2 seconds). Further decrease of ϵ and increase of n brings small benefits in verified robustness (1.7%).

We also ran our method on ImageNet – the method takes ~ 2 minutes per image due to increased number of pixels. The main issue here is that all existing verifiers lose too much precision when propagating constraints through a full blown ImageNet-sized network. Finally, we ran verification of our pixel bounds through a larger network (62K neurons), obtaining similar robustness to the network used in Table 1 (though expectedly, verification time increases).

We note that using IBP for both bound computation and propagation will be more scalable but suffer from very low precision – strictly worse than the Interval baseline of Table 1, which is already much worse than our method.

We will add all updated results and above clarifications to the paper.

Reviewers #1, #3: *Why is the input always assumed to be perturbed with L_∞ noise?*

Our method does not assume L_∞ noise and we do have experiments without it (see Table 1). We did perform some experiments with L_∞ noise following Singh et al. [5] who certified the specific composition of L_∞ noise and rotation.

Reviewer #1: *Why are the verified networks here seemingly robust to these attacks?*

This is because the networks are trained using standard data augmentation (e.g., if we verify rotations, we augment data with rotations). Note that the same training method is also considered by Engstrom et al. [2] and is shown to significantly increase robustness to geometric transformations compared to networks trained without this augmentation.

Reviewers #1, #2: *Do you define a similarity metric under geometric attacks?*

We do not define such a metric in this work but focus on classic transformations (e.g., rotations) which are parameterized. As usual, the user specifies the parameters for which they want to certify the network (e.g., the -30 to 30 degrees for an MNIST image used in Table 1 can be argued to contain images that are indeed visually similar to humans).

Reviewer #2: *How does this relate to vector field based transformations? Is it a subset thereof?*

Our transformations capture the most common instances of vector field transformations, but not all. Note that generally vector field transformations are not guaranteed to preserve image similarity (unless bounded by a norm) which is why we focus on transformations known to produce similar images according to human perception (e.g., rotations).

Reviewer #3: *Can you provide an upper bound on verifiability?*

Yes. We computed an upper bound for the first experiment on CIFAR-10 with rotations $\in [-10, 10]$. We performed “Worst-of-k" attack from [2] which, for every image, randomly samples 100 parameter choices and checks for misclassifications. This gives an upper bound of 73% (verification rate is 51.5% in Table 1). We also ran DeepG with twice as many parameter splits and verified 72% of images (only 1 image remained). In general, DeepG gets close to the upper bound by increasing the number of splits. However, such increase in the number of splits is only possible if the # of parameters is small (otherwise, the cost is prohibitive). We will add these results to the paper.

Reviewer #3: *Can you use a looser offset bound and do branching for parameter refinement at a higher level?*

This is an interesting idea and we considered it earlier. The main problem is that each refinement requires a call to the verifier, which is the main bottleneck as mentioned above. Ideally, there would be a policy (branch and bound or another heuristic) which refines parameters so the number of verifier calls is minimized (so far we did not find a policy which noticeably improves over the uniform refinement used in our work). Importantly, while interesting, this direction is orthogonal to our approach as any split will benefit from more precise linear bounds.