We thank the reviewers for their in-depth and constructive reviews. We are happy that idea, presentation, implementation and experimental validation are well perceived and that all reviewers stated that they lean towards acceptance.

Novelty with respect to Tensor Field Networks (TFN) The reviewers state that our work is novel (R3), impactful (R4), valuable (R1), sound (R2), and an important theoretical contribution (R3). R1, R2, & R4 state that the approach could be seen as a straightforward extension of TFNs; however, as R3 states, such extension is nontrivial. We are able to derive invariant weights only through making keys and queries dependent on relative positions. This differs significantly from regular attention and is not an obvious choice. This extends TFNs to the graph setting, with the bonus of permitting edge features. Furthermore, this is one of the first examples of a nonlinear equivariant layer. In Section 3.2, we show our proposed approach relieves the strong angular constraints on the filter compared to TFNs, therefore adding representational capacity. This constraint has been pointed out by several authors in the equivariance literature to limit performance severely (Weiler & Cesa, 2019). We will emphasize this comparison in the introduction (R3).

Computational Cost / Scalability Concerning scalability (R2, R3, R4), wrt the original TFN implementation (R2), we will extend the discussion. The spherical harmonics (SH) have to be computed on the fly for point cloud methods, a bottleneck of TFNs. The TFN authors ameliorated this by restricting the maximum type of feature to type-2, trading expressivity for speed. We built a faster SH library (10x faster on CPU, 100-1000x on GPU than existing libraries) and can handle any SH type. We will include a speed comparison. E.g., for a ScanObjectNN model, we achieve $\sim 22\times$ speed up of the forward pass compared to a network built with SH from the `lielearn` library. **We will release code** for the camera-ready (R2). In the meantime, a recipe for the GPU-based SH generation is already in Appendix C.

(R3): (1) We conducted experiments with up to 2048 points, but with no significant performance improvements. We suspect this is due to the global pooling. Examining cascaded pooling via attention is a future research direction. (2) For the QM9 dataset, we efficiently deal with varying point cloud sizes, leveraging the DGL library for GPU parallelisation.

Experimental Results The experiments validate effectiveness and equivariance of the approach (R3), showing that the SE(3) Transformer consistently outperforms non-equivariant self-attention and TFN (R2) while providing reproducability (R1,R2,R3,R4). It is also noted we do not improve on the previous SOTA on ScanObjectNN and QM9. It is worth noting the TFN baseline we report is significantly scaled up compared to the original implementation (more channels, higher degrees & more layers), enabled by the efficiency improvements described above. The performance difference between TFN & SE(3) Transformer comes on top of those improvements. We want to stress this is the first time an equivariant point cloud network based on irreps reports competitive results on object classification. While these results are not SOTA, we are close. Furthermore, on ScanObjectNN, the baselines are specifically designed for that task, and we introduce a component rather than an architecture, so could in theory combine the benefits of the SE(3) Transformer with, say, PointNet++. From the point of view of the equivariance literature, we have some work to do to catch up with non-equivariant works, but we feel the SE(3) Transformer makes a meaningful step forward in closing that gap. This work brings equivariant methods closer to being a useful tool in the practitioner's toolbox. As correctly noted by R3, we deliberately break SE(3) equivariance for ScanObjectNN. Feeding the z-coord. as a separate, scalar input makes the network SE(2) equivariant. In our opinion, this does not indicate a weakness of the approach of SE(3) invariance in general. Instead it shows object classification datasets are not fully symmetric. We happily include the SE(3) equivariant version in table 2. Why not ModelNet40 (R2)? Due to an internal policy, we were limited to datasets with proper licensing, which ModelNet40 does not provide. ScanObjectNN is a more recent dataset providing a tougher alternative to ModelNet40 based on noisy real world sensor data. **Do the baselines *need* 1024 points (R4)?** Initial experiments show, when training & testing on 128 points only, we do outperform the baselines (PointCNN: $80.3 \pm 0.8\%$, PointGLR: $81.5 \pm 1.0\%$, DGCNN: $82.2 \pm 0.8\%$, **ours**: $85.0 \pm 0.7\%$). A more detailed analysis will be added to the appendix.

On QM9, the only network which beats the SE(3) Transformer on all tasks is LieConv, a concurrently developed approach published after the submission of this work. The equivariance of LieConv networks is based on (left-) regular representations, coming with its own set of dis/advantages (e.g. stochastic forward pass and complicated extension to higher degree input representations). Cormorant is on par but uses expensive Clebsch-Gordan transform nonlinearity - the authors state that training is unstable, an issue we do not have. (R1) Error bars: Table 1 contains error bars, in table 2 it is in the caption. In Table 3 we shall add them. For QM9 stddev is small - e.g. we got ~ 1 meV for the $\varepsilon_{\text{HOMO}}$ task.

Other Questions (R2) L553: quadratic - should say 'square'; (R2,R3) **Related Work** - We thank the reviewers for the pointers and will discuss these references in the paper; (R3) L6 - the claim of **improved sample complexity** is supported by [Fig. 10 Worrall et al. (2017), Fig. 4 Winkels & Cohen (2018), Fig. 2 Bekkers et al. (2018), Fig. 4 Weiler et al., (2018)], we will include a quantitative analysis; (R3) We will add bold for Table 3; (R3) Clarify 'symmetries of the task': we use the definition of symmetry from Mallat, (2016) "Understanding Deep Convolutional Networks" where symmetry is a property of a function/task; (R3) How are input features computed for 3D point cloud? We use relative coordinates as features for the first layer (see appendix D.1); (R3) Equation (13) - c' and c are placeholders for input channels; (R4) We did not augment data for the n-body experiment, but the data is sampled uniformly across all orientations; (R4) L109 - '$i = l$' should be deleted. Finally, we thank the reviewers for all mentioned typos.