Reviewer 1

1. No, they are usually different. The set encoder always maps its $d \times n$ input to a z-space. It is used to encode the ground-truth input set (use as encoder) and encode all the $\hat{\boldsymbol{Y}}^{(t)}$ sets (use in decoder).

2. Yes, this is the loss for the outer optimisation to train the network weights. We will try to make this clearer.

3. They are nested in the sense that differentiating the outer loss requires differentiating through the inner optimisation procedure too. The inner optimisation is nested within the outer optimisation. Only one step of the outer optimisation is shown, so there is an implicit `for` loop around Algorithm 1 for training. We will try to make this clearer too.

4. The MLP baseline uses Eq. 1 and 2, just like in existing work that uses MLPs to predict sets. Algorithm 1 is indeed a core part of our contribution. It is placed near the main model section, while Eq. 1 and 2 are firmly in the background section. We will update the introduction to explicitly mention that Algorithm 1 is part of our core contribution.

Reviewer 2

Correction: our proof is that the gradient of a permutation-invariant function is permutation-*equivariant*, not permutation-*invariant* as you stated in your review. There is a small but important difference.

1. The results of the MLP baseline are obviously so much worse that we didn't feel the need to include this. They are what you could expect from the difference in qualitative results and we can include them in an update. In particular, the test Chamfer loss is 0.00010 for our model and 0.00076 for the MLP on MNIST with mask feature.

2. The MLP baseline, while simple, *is exactly what many existing papers use* for set prediction (see the first sentence in the related works section for some of them). The MLP is the most suitable baseline because it is both simple (no need for much extra explanation for a reader to understand), as well as actually widely-used for set prediction in the literature. We explain why MLPs should struggle with modeling sets, so it shouldn't be a surprise when they do struggle with sets in our experiments. The only other relevant approach is RNN-based, which also has the responsibility problem.

We tried an LSTM (like in [19]) as set decoder, which improves the MNIST baseline to 0.00023 loss and is similar to the MLP on CLEVR (e.g. bbox AP_{95} = 61.2%, state AP_{∞} = 3.3%), so still much worse than our model. Self-attention maps sets to sets, so it can only be used once you already have a set, not for vec-to-set as we are dealing with. Rezatofighi2017 is irrelevant (their title is not as general as it claims to be) because it's more about multi-labeling tasks rather than general set prediction: their sets are sets of discrete labels from a small finite domain, so they can be ordered just like in normal classification or multi-label tasks and is thus "easy". Our work deals with sets where the elements are feature vectors of real numbers, which their method doesn't apply to. In Rezatofighi's follow-up paper (we cite this as [16]) using the same problem set-up as our paper, they use a Hungarian loss and an MLP decoder. So, by comparing against an MLP baseline, we *are* comparing against their (and others') work [1]. We will make this more explicit.

3. The vast majority of sets in the real world are variable size, so the case of fixed size is not particularly relevant. We follow [16] with our MLP baseline, which has this exact mask output. Removing it seems to improve results of the MLP on MNIST to losses similar to our model, but now it is solving an easier task than our model is intended for.

4. FSUnpool is irrelevant because it can only be applied in auto-encoders, not in general set prediction. We state this in Lines 182–183, 223–226. We are evaluating set prediction methods on auto-encoding, not set auto-encoders specifically. Lines 217–218 state that replacing FSPool with other poolings (we tried mean, max, sum) makes the baseline worse.

Reviewer 3

1. Instead of stopping at a fixed 10 iterations as we did, an alternative is to stop when $L_{\text{repr}}(g_{\text{enc}}(\hat{\boldsymbol{Y}}), \boldsymbol{z})$ is below some threshold: this stops when the encoder thinks the representation is of a certain quality corresponding to that threshold. A lower L_{repr} means a higher quality prediction because the representation of prediction and target match better.

2. When computing the outer loss, the initial state $\hat{\boldsymbol{Y}}^{(0)}$ is learned too, which should help with the initialisation. In essence, $\hat{\boldsymbol{Y}}^{(0)}$ is moved to a state that is close to training examples in set space on average, which reduces the distance the inner optimisation has to travel. This is balanced with a $\hat{\boldsymbol{Y}}^{(0)}$ that is close in the representation space; this can be learned because the inner optimisation is being differentiated through.

3. Please refer to our answer to Reviewer 2, point 2, for an additional LSTM baseline.

[1]The other things they developed (permutation head O_2, cardinality head α) are not actually used (they stated this in their openreview rebuttal) or don't help in the experiments in their main body respectively. Thus, their approach is almost exactly the same as our MLP baseline and their experiment in section 4.1 is very similar to how we used the MLP in section 5.2 for bounding boxes.