We would like to thank the reviewers for their thoughtful comments. We are pleased that all four reviewers recommend acceptance, with R1 noting that they *"do not see any weaknesses with the current work"*. Furthermore, we are glad to hear that the reviewers are happy with the correctness of our work, the clarity (both R3 and R4 in particular said our paper was *"very well written"*), and the references to previous work (R2 stating that this was *"quite comprehensive"*). Moreover, we appreciate the reviewers highlighting the importance of our problem (R3: *"The paper aims to tackle a very interesting problem"*, R4: *"It filled an important application gap "*), the benefits of our approach (R2: *"proposed architecture to generate synthesis DAGs is very interesting"*, R4: *"many technical contributions"*), and the *"exciting potential extensions"* (R1). We respond to the individual specific questions from the reviewers below:

Reviewer 1 *[.. caveat on imperfect MT and CASP oracles ..]* This is a good point; we are happy to add a statement to this effect in the paper to make it clearer to those less familiar with this area. An important point here is that our model is agnostic about which particular reaction predictor we use, so one could use a template based approach to achieve higher precision at the expense of lower recall. Being able to use any reaction prediction model also means that our method can take advantage of the future, independent development of these models to obtain improved performance.

[..on formatting of references..] Thank you for spotting this, this has now been fixed!

[.. on reconstruction ..] We are happy to bring this from the Appendix into the main text, thanks for the suggestion. As an alternative to greedy decoding we have also run an alternative where we sample 100 times from the model for each example and then resort on predicted probability. This obtains 66.2% accuracy.

Reviewer 2 *[.. on quality of reaction predictor ..]* We shall add a comment on this issue as you suggest (see also our response to R1 above). Currently, if the reaction predictor suggests no products we select one of the input reactants to act as the product of the reaction – note however that if this happened frequently we would get very poor novelty results.

[.. font size in figures..] We shall increase the font size in the figures as per your suggestion – thanks for this.

[.. dataset size ..] You bring up an interesting point. Our dataset size of 72000 does not appear to be a significant constraint so far in our experiments, as evidenced especially on the Valsartan optimization task: our approach was still able to find good molecules for this optimization task, even though the training set had no good molecules. However, we definitely agree it would be interesting to explore larger DAG datasets in future work! We believe these could either be extracted from large proprietary reaction datasets such as Reaxys or Pistachio or perhaps generated synthetically.

Reviewer 3 *[3 (1)]* To deal with different possible orderings of the DAG serialization we randomly sample an ordering. As described in the caption to Figure 2, we also start at building blocks that are furthest from the final molecule node. This reduces the distance between introducing a building block and first using it to form a product molecule.

[3 (2)] Yes, the model could produce a DAG which is a sub-graph of another DAG by choosing to make a final product instead of an intermediate product for one of the product nodes. Although, perhaps pedantically, we would probably still class these as in-distribution as the model could have been trained on two such DAGs at training time.

Reviewer 4 *[.. Training and inference times ..]* The training set DAGs have an average of 4.6 nodes, with the final molecules containing an average of 20.5 heavy atoms and 21.7 bonds (between heavy atoms). The average number of actions required to construct these DAGs is 11, as each reactant often contributes several atoms and bonds to the product. Using a NVIDIA K80 GPU it takes ≈ 7 mins to run a training epoch for DoG-AE (≈ 0.4 secs per batch of 64). At inference time, where we do not initially have access to the complete sequence, we usually run larger batches, due to the fixed costs and latency of communicating with a Molecular Transformer server. It takes ≈ 29 secs to carry out *per batch of 200 DAGs* (and ≈ 12 secs per batch of 64 DAGs). Our approach could be sped up by using faster reaction predictors, although note that it still compares very favorably to full scale synthesis planning which takes on the order of 10 secs – 1 minute *per molecule*. Thank you for your comment, we shall add these details to the paper.

[.. GNNs versus fingerprints..] In this work we use GNNs due to their improved performance over fingerprints in previous work. GNNs are also popular in reaction prediction (e.g. [45,46]). It is worth pointing out that our GNN implementation is fairly lightweight: it only requires $\approx$ 73k parameters[1], the same weights are used in every message passing step, and the GNN/weights are shared between the encoder and decoder in the DoG-AE model. As we focused on the concept of DAG generation in this paper, and found performance to be sufficient, we did not perform extensive hyperparameter/architecture search. In production use, a fingerprint + MLP (as you reasonably suggest) or a different GNN/message-passing scheme, or perhaps even a concatenation of fingerprint and GNN features (shown to sometimes work well in [Yang et al., 2019, Fig. 16]) may be preferred, depending on performance/compute requirements.

Yang K, Swanson K, Jin W, et al. (2019) Analyzing Learned Molecular Representations for Property Prediction. Journal of chemical information and modeling 59(8): 3370–3388.

[1]Note a learnt linear projection from 2048 dimensional molecular fingerprints to 50 dimensional embeddings would require $\approx$ 102k parameters in comparison.