We thank the reviewers for their time and valuable feedback that will strengthen our paper. Overall, we are glad the reviewers agree that our paper identifies general subgraph prediction as a sizable gap in the GNN literature, introduces a sound and interesting initial solution to fill this gap, and provides 7 baselines and 8 new datasets to jump-start this area of research. We think the following clarifications and new analyses resolve all key issues raised.

(1) Novelty & general advancement. **R1** and **R4** questioned our theoretical or general advancement. We respectfully disagree and would like to clarify our paper's five key advancements. **1)** SUB-GNN *"proposes a new field of study: subgraph classification"*, *"is the first method that works with varying subgraph sizes"*, and *"can easily deal with large subgraphs"* (**R1**). **2)** In our initial solution for subgraph classification, the taxonomization of subgraph topology (Table 1) is grounded in network science theory [Yang & Leskovec, ICDM 2012], and our design of disentangled channels is directly rooted in these theoretical properties. **3)** We propose 7 strong baselines, each carefully designed to test SUB-GNN and its components. **4)** We construct 8 new and original datasets. **5)** SUB-GNN is a flexible framework for learning subgraph embeddings—it can operate with any sampling scheme, patch embedder, and similarity function.

(2) Related work & baselines. **(2.1)** **R1** pointed out a missing reference to Meng et al, 2018, which, unlike SUB-GNN, performs prediction on small (3-4 nodes), fixed-size subgraphs. We extensively studied this interesting paper, but left out the citation in error. **(2.2)** **R4** requested a baseline in which a GNN is run on each subgraph. We agree this baseline is important, and it already appears in our paper (L275, Baseline 7). We believe the confusion occurred because we incorrectly stated that the model uses pretrained node embeddings (L275). In fact, the GNN model (GIN) trains node embeddings *from scratch* on each task. We will correct this typo in the final version.

(3) Run-time. **R3** raised an important question about the efficiency of SUB-GNN and the performance gains from pre-computation. We find that pre-computation results in an 18X training speedup. With pre-computation, training SUB-GNN on PPI-BP takes 30s per epoch on a single GPU, of which 25%, 43%, and 32% are spent on neighborhood, position, and structure channels, respectively. We will include a full run-time analysis in the final version. We note that training time can be further improved via locality sensitive hashing or k-hop shortest paths for similarity calculations.

(4) Reproducibility and code. We thank **R2** and **R3** for pointing out that *"the source code is provided to make the work reproducible."* **R4** and **R1** questioned whether our descriptions are sufficient to implement the model. To address this, we included in our initial submission a link to the complete SUB-GNN implementation, together with examples of usage, baseline implementations, and hyperparameters. Furthermore, we appreciate the reviewers' suggestions for improving the clarity in our writing, and will incorporate them in the final version to improve reproducibility.

(5) Datasets. **(5.1)** **R1** raised an important point on the possibility of information leakage in case train and test subgraphs overlapped. While we disagree that this constitutes information leakage, as subgraphs with overlapping nodes can have different labels, we strongly agree that assessing performance as a function of overlap in nodes between train and test subgraphs is important. We have two answers to this point. First, test subgraphs in the COMPONENT and CORENESS datasets have zero nodes in common with any train or validation subgraphs. Our strong performance on these datasets (Table 2) indicates that node overlap between subgraphs does not explain SUB-GNN generalization power. Second, we generated an extra COMPONENT dataset with varying degrees of overlap between the test and train/val subgraphs. We find that SUB-GNN performs strongly on both test subgraphs with low overlap (<10%) and test subgraphs with high overlap (>50%) (92.9 vs 96.1 F1-score). **(5.2)** **R2** questioned whether the HPO tasks are *"actually done in practice."* HPO tasks are of incredible practical relevance. For example, the Undiagnosed Diseases Network, a major program backed by NIH, seeks to diagnose patients from HPO codes [Splinter et al. NEJM 2018]. We designed the HPO datasets to directly mimic this task of rare disease diagnosis [Bradley, Nature Reviews 2020; Austin & Dawkins, Nature 2017].

(6) Further considerations. **(6.1)** **R1** questioned the effect of different node representations and similarity functions. In a new experiment, we used GraphSAINT [Zeng et al. ICLR 2020] to learn node representations. SUB-GNN outperforms the strongest baseline by 10.5% on average on synthetic datasets. We will experiment with graph edit distance as the similarity function for the structure channel and include in the final version. **(6.2)** **R1** raised an important point about whether SUB-GNN can be inductive. SUB-GNN is inductive because it can generalize to new subgraphs on unseen portions of the graph, a notion similar to that defined in Meng et al. 2018. **(6.3)** **R1** asked several questions about our anchor patch sampling procedure. We do not average over all possible anchor patches. For the structure and border position channels, anchor patches are sampled independently of the subgraphs and shared across subgraphs; this is broadly inspired by the shared anchor-set sampling in P-GNN [You et al, ICML 2019], and is inductive to new subgraphs. For the neighborhood channel, sampling is a standard k-hop neighborhood sampling procedure, performed locally within and around each subgraph. **(6.4)** **R1** rightly pointed out that permutation invariance of $h_{\mathbf{X},c}$ does not remove all position information. We agree and will correct this in the final version. The order invariant representations are needed for layer-to-layer message passing. **(6.5)** **R1** noted that SUB-GNN has many hyperparameters and raised the important question of how to set them without high family-wise error rates. We used random search (see Appendix and submitted code). Importantly, we performed the same number of hyperparameter searches on all baselines and SUB-GNN. Of the hyperparameters we optimized, most are standard in modern neural networks. Only 7 (border sizes, number of anchor patches, etc.) were peculiar to our model, some of them indicating flexibility of SUB-GNN, e.g., to turn on/off a channel. We observed similar performance on validation and test sets, indicating no overfitting.