We appreciate insightful comments from all reviewers to our paper 'Graph cross networks with vertex infomax pooling'. First, we address three common concerns about the proposed vertex infomax pooling (VIPool) and GXN.

• **VIPool vs. Other graph pooling. 1.** VIPool is a novel method for vertex selection, which is also critical to network science, graph theory and graph signal processing. Recent graph pooling methods mainly have two approaches: the vertex-grouping-based approach (DiffPool [47] and StrucPool [48]), which groups vertices to some clusters; and the vertex-selection-based approach (gPool [20], SAGPool [29], AttPool [25]), which selects representative vertices and then coarsens the graph based on the selected vertices. **2.** VIPool provides an *explicit optimization* (Eq.1) for vertex selection, which can be trained via self-supervision. Most vertex-selection methods, including gPool, SAGPool and AttPool, purely rely on a subsequent task to select vertices, lacking generalization and interpretation. For example, only VIPool can be used in active sampling for semi-supervised learning; see Appendix E. **3.** VIPool *resolves the clustering issue* in many vertex-selection-based approaches, including gPool and SAGPool; see Appendix F. The clustering issue is: most selected vertices come from a small subgraph. In VIPool, the vertex-selection criterion explicitly punishes those selected vertices that share similar neighborhoods. **4.** Compared to recent vertex-grouping-based approaches, VIPool has a *lower computational cost* than StrucPool ($O(N)$ vs. $O(N^3)$); DiffPool requires a subsequent task to supervise vertex clustering, while VIPool has an explicit optimization to select vertices.

• **VIPool vs. Deep graph infomax (DGI).** Both leverage mutual information neural estimation (MINE) [2]. Two major differences are: **1. Aim.** VIPool aims to obtain an optimization for vertex selection whose objective function is obtained through MINE; while DGI aims to learn a *graph embedding*, which is a trainable mapping updated through MINE. **2. Formulation.** Since VIPool selects vertices in a given graph, VIPool trains on *a single graph* and its training samples are positive/negative *pairs of vertices and neighborhoods* in the same graph; since DGI maps each graph to an embedding, DGI trains on *multiple graphs* and its training samples are positive/negative *pairs of vertices and graphs*.

• **GXN vs. Graph U-Nets.** Two major differences are: **1. Intermediate fusion vs. late fusion.** GXN fuses features at multiple scales in each network layer while graph U-net fuses features at the end of each scale. **2. Deep vs. shallow learning in each scale.** GXN extracts features multiple times in each scale while graph U-net extracts single-scale features only once in each scale and then uses a skip-connection to fuse features across scales.

Next, we address the specific questions from each reviewer. **Please zoom in to see the figures precisely.**

• **Reviewer 1:** Q1: Compare VIPool to previous methods. A1: see VIPool vs. Other graph pooling.
Q2: VIPool is similar to DGI. GXN is a straightforward extension of graph U-net. A2: We researchers all build our works on the shoulders of giants and we pursue simple, yet nontrivial designs. VIPool and GXN make distinct contributions to self-supervised trainable vertex selection and multiscale architecture design, respectively; see VIPool vs. DGI and GXN vs. Graph U-Nets. Compared to previous works, both designs are nontrivial, effective and intuitive.
Q3: How many runs for vertex classification with different model initialization? Try other dataset splits. We run 100 times with different initializations. We test 5 random splits and compare GXN to GCN and GAT. The results of the semi-supervised vertex classification on Cora are GCN/GAT/GXN: $78.4 \pm 0.7/79.7 \pm 1.3/\mathbf{81.4 \pm 0.8}$

• **Reviewer 2:** Q1: VIPool builds on known techniques. A1. see VIPool vs. Other graph pooling and VIPool vs. DGI.
Q2: Effects of the numbers of selected vertices $|\Omega|$. A2. Fig. 1 (a) shows the appropriate and effective $|\Omega|$ is important.
Q3: Why do we modify $C(\Omega)$ to solve (1)? A3. The modification makes the method faster without sacrificing too much performance. On IMDB-B: before modification: $77.7 \pm 0.5\%$ accuracy, 3.7×10^{-2} *second* test time cost per graph; after modification: $77.3 \pm 0.8\%$ accuracy, 4.3×10^{-5} *second* test time cost per graph, which is much faster.

• **Reviewer 3:** Q1: Not enough comparisions of VIPool to other methods. A1. see VIPool vs. Other graph pooling.
Q2: Combine StructPool to GXN. A2. Table 1 shows VIPool outperforms StructPool on graph and vertex classification.

	IMDB-B	IMDB-M	COLLAB	DD	PROTEINS	ENZYMES	Cora	Citeseer	Pubmed
GXN-StructPool	76.40	54.02	79.35	83.77	80.03	**60.17**	84.4	74.2	79.8
GXN-VIPool	**77.30**	**54.57**	**80.62**	**84.26**	**80.38**	59.59	**85.1**	**74.8**	**80.2**

Table 1: Based on the same GXN architecture, we compare StructPool and VIPool on graph and vertex classification.

• **Reviewer 4:** Q1: Effects of neighborhood radius R for vertex classification. A1. Fig. 1 (b) shows that various Rs are stale and lead to minor effects for vertex classification. We choose $R = 3$ in our model.
Q2: Compare the training time, show the training process. A2. Fig. 1 (c) shows both task and pooling losses converge stably; the overall loss descends with α; and GXN converges faster than StructPool and graph U-net.
Q3: Effects of α and mutual information in training. A3. Fig. 1 (d) shows the training loss converges stably with various α. We initialize $\alpha = 2$ to balance task objective minimization and mutual information maximization.
Q4: VIPool on other architectures. A4. On IMDB-B: Encoder-decoder+VIPool: $74.0 \pm 1.0\%$; Readout+VIPool: $76.3 \pm 0.9\%$; Graph U-net+VIPool: $76.7 \pm 0.5\%$; GXN+VIPool: $\mathbf{77.3 \pm 0.8}\%$. GXN outperforms the others.
Q5: Show how difficult to train feature-crossing and effects of number of layers. A5. Fig. 1 (e) shows although fewer feature-crossing layers converge faster, training feature-crossing layers is not hard.

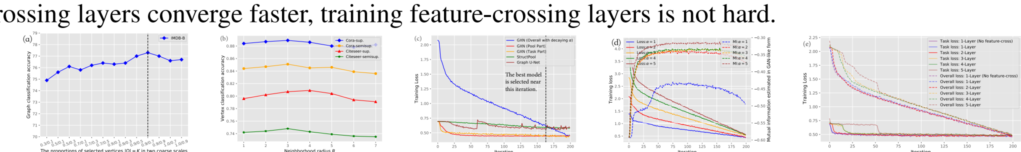

Figure 1: (a) Effect of $|\Omega|$; (b) Effect of R; (c) Training loss; (d) Effect of α; (e) Effect of feature crossing and hidden layers.