We thank all reviewers for their fair and constructive reviews.

General remarks We will revise and restructure the appendix (and the main paper), especially concerning references/theorem numbering and redundancy between the main text.

Scalability We agree that our methods are not ready for industrial-scale graphs. We view our contribution as methodological work that is the first step to make higher-order methods more practical by leveraging *sparsity*, which is perhaps the most significant parameter associated with a class of graphs. Moreover, we want to stress that we empirically verified that our method offers significant benefits (compared to standard GNNs) in the regime of small graphs such as molecules. We will integrate the discussion of Appendix F into the main text (complexity, sampling, ...).

R1 *Computation of isomorphism type* We agree that the method is only of theoretical interest for a large choice of k. Nonetheless, we believe that we provided sufficient empirical evidence showing that for small $k \in \{2, 3\}$, the method provides benefits over standard GNN and other higher-order architectures. Note that strictly speaking, the k-WL does not need to perform an isomorphism test. It merely performs an equality test. Lines 97-98 (in the "appendix.pdf") describe the condition for two tuples to have the same initial labeling: it is only the identity mapping $\mathsf{id}\colon [k] \mapsto [k]$ which we demand to be a (partial) graph isomorphism. Hence, we only need to check "equality" of the two tuples and not their "isomorphism". If we had considered two *sets* of k vertices instead of two *tuples* of k vertices, then we would have needed isomorphism testing (as the reviewer suggests).
Subgraph learning Yes, we agree that results on link and triadic prediction would further highlight the approach's generality. We plan to include them in a revised version/future work and add a discussion to the main paper.
Aggregation scheme We considered all tuples. We will make the complexity, especially the dependence on k, clearer in the main paper, e.g., by including the discussion of Appendix F in the main paper.
Directed graphs Thank you for bringing this to our attention. The applicability of our arguments to the direct case depends on the way we define the k-WL for directed graphs. It is typical to consider both in-neighbours and out-neighbours, but separately. Hence for a tuple T, we have "local-in"-neighbors and "local-out"-neighbors. The local unfolding at the tuple T will have both in-neighbours and out-neighbours at depth-1, and so on. Therefore, in this kind of WL, we are essentially dealing with the underlying undirected graph of the given directed graph. If the graph is weakly connected, we will still see all 1-neighbours of a tuple (local-in,local-out, global) somewhere down the local unfolding tree. In that case, we are fine with weak connectivity. Of course, we could choose to define a WL-variant with only "local-out"-neighbors, where we agree that strong connectivity will be necessary. Since the typical notion of sparsity for directed/undirected graphs is the same, i.e., low edge-density, we would argue that it is natural to stick with the first formalism. In that case, weak connectivity will suffice. We will clarify this in the revised version.

R2 *Performance of δ-2-LWL$^+$* The δ-2-LWL$^+$ is as a middle ground between the purely local δ-2-LWL and the global δ-2-WL. The $+$-Version achieves good generalization performance due to slower convergence while preserving certain global information, which is needed to derive Theorem 2. We will expand on this in the revised version.
Other GNN architectures We agree and will include more recently-proposed GNNs as baselines.
Lower results for GIN We used the implementation available in Pytorch Geometric. The lower numbers are because we did not use an initial one-hot degree feature for datasets that did not provide node labels. We will add a comment and add a row for results using degree features. (Note that all kernels are computed without this information)
Leman vs. Lehman Leman/Lehman personally stated that he prefers the transcription Leman (through private communication with Russian graph theorist Ilia Ponomarenko).[1] Moreover, the spelling Leman is also used throughout the theory community.
Proof of Prop. 1 Thank you for bringing this to our attention. The missing part directly follows from the CFI construction outlined in [15]. However, a formal proof is quite involved and would require repeating the reasoning of the paper above. We will add a proof sketch to the revised version of the paper.

R3 *Scalability/limitations* See general remarks above. We will report absolute running times (<1h for the local architecture on the 10k subset.).
Readability/relevance to a broader audience We will incorporate intuition in the revised paper and emphasize the neural architecture. Moreover, we will try to make the main paper as self-contained as possible.

R4 *Notation* We agree and will make the main text more self-contained.
δ-k-LGNN We note that the δ-k-LGNN is slightly weaker than the δ-k-GNN as it does not use the $\#$ labeling. (It could be included but would require preprocessing.). Moreover, observe that the δ-k-GNN may learn to combine the local and global neighborhood information in a more fine-grained way compared to the kernel (due to its discrete nature). Moreover, we stress here that the δ-k-LGNN still achieves good performance while being much faster than the δ-k-GNN or k-WL-GNN. We will add an expanded discussion in the revised paper.
Explanation on experiment part Thank you for the good suggestion; we will incorporate it into a revised version.

[1] https://www.iti.zcu.cz/wl2018/pdf/leman.pdf