We sincerely appreciate the reviewers for their careful reading, constructive questions and suggestions. We would very much like further exchanges to improve our work, but the following is our best effort within the current limits.

First, we address questions appeared at least twice. We write **P1, P2** for paragraph reference, and **Rx** for reviewers.

P1: Motivation of using similarity. We discuss two main motivations here: lack of graph loss, and empirical failure of distinguishing power. **First**, in tasks where the target is a fixed graph, e.g. graph autoencoding, we need an efficient differentiable measure of graph similarity to guide the model toward the target. Existing works either use GNN without principle or use costly graph matching. Being a graph pseudo-distance, WLS is a good candidate for graph loss; thus, we successfully applied it to graph generation, although preliminary yet. **Second**, most of GNN theory, including GIN, focuses on the distinguishing power. That is, they only consider when the representations are precisely equal. However, higher distinguishing power (arXiv:1905.12560, 1905.11136) does not translate to empirical advantage. A reason we suspect is that, deep learning works on continuous representations where distance can be more important than equality. Optimization, generalization, and as **R1** suggested, adversarial robustness analysis of GNNs require the continuity perspective. To the best of our knowledge, our work is the first to incorporate continuous similarity into designing GNN.

P2: Comparison to GIN. We believe WLS has sufficient advantages over GIN and other GNN models. **First, difference in detail.** Both equation 4.1 in the GIN paper and official GIN implementation uses sum$\rightarrow$weighted sum$\rightarrow$transform. The central node's (v_c) representation is distinguished from its neighbors' (v_N) only by scale. In contrast, the WL test, and hence WLS, separate v_c from v_N. Even with the simple WLS-GNN, the concatenation makes the next MLP act differently to the pair. **Second, WLS makes different inductive biases easier to apply.** WLS interprets the transformation as a feature map, and we experimented with two extreme biases: RBF kernel without parameter learning, and MLP without significant inductive bias. Both cases show that WLS is capable of generating useful graph representations, as shown from the empirical improvements. **Third, performance.** With no other component than aggregation, WLS-GNN already outperforms popular models. Techniques from other GNN models, e.g. pooling-over-layers from GIN or JKNet, may likely improve the predictive performance further. Moreover, as **R1** kindly commented, we provided well-motivated reasoning for each step via the WL test and kernel distance.

The following are responses to the individual reviewers.

Reviewer #1. **Q1: Motivation of using similarity.** Please refer to **P1** for motivation. We appreciate very much the suggestion of further subjects to study similarity-based models on. **Q2: Clarity.** We agree that a flow diagram can significantly help the readers and will add one. We will move the GNN graph classification table to the main paper. **Q3: Related work.** We will expand our review of GNN literature. Deep graph kernel is definitely in our scope and will be mentioned. **Q4: Typos.** The comments are absolutely right and will be corrected. **Q5: Broader impact.** Thank you so much for providing concrete examples. We will carefully discuss with our peers to consider further hazards.

Lastly, we are afraid of a slight difference in understanding that may affect the evaluation negatively. The mentioned transform scheme for GNN is absolutely an interesting direction; however, we replaced the Taylor approximation and random projection with an MLP, which results in a simple GNN similar to other models, yet empirically strong.

Reviewer #2. First, we apologize for including too much material in the appendix. **Q1: Compare to GIN and similar aggregations.** Please see **P2** and additionally **P1**. **Q2: Graph classification with GNN.** The table is in the appendix, but will be moved to the main paper. **Q3: ZINC.** Thank you for pointing this out. Only GatedGCN used edge features, and we will rename it to GatedGCN-E. We also newly ran the WLS-GNN graph classification model without edge features on ZINC. It obtained MAE 0.332$\pm$0.007, outperforming all baselines including GatedGCN-E significantly, but not WLS-E. We will include it in the table. **Q4: Regarding Figure 2(b).** We appreciate the suggestion. We will move the statements regarding Figure 2(b) to the main paper. **Q5: WLS-GNN readout.** We used the same readout used for other models. A 3-layer MLP with specified dimensions is applied to the average of final layer node representations.

Reviewer #3. **Q1: Compare to GIN and explain motivation.** Please see **P1** and **P2**. **Q2: Why R-GCN collapses?** WLS as a graph loss and a discriminator are our next subjects. Thorough investigation is required, but here is our yet untested hypothesis. The MolGAN discriminator has several components, and it is unclear which graphs have similar representations. In contrast, WLS creates similar representations for similar graphs, and possibly, chemically valid graphs can be grouped together in the representation space, which helps stabilization. **Q3: Stacking random projection may accumulate errors.** In downstream tasks, there are many other factors, but the accuracy degrades after 5-8 layers depending on the datasets. If we stack 10 projections with dimension 200, 95% of the norms are within 30% error. For dimension 1000 we have 14% error. Considering many applications of shallow GCNs and the characteristics of real-world graph datasets, we believe many tasks exist where this is sufficient. Details will be added to the appendix.

Reviewer #4. **Q1: Linear kernel.** Thank you for informing us of the issue. We tested linear SVM which was similar to or marginally worse than the main paper on 5 out of 6 datasets, and significantly worse ($> 25\%$p difference) on ENZYMES. The relative performance remains the same and will be mentioned. **Q2: About hyperparameters tuned to each split.** We completely agree that one set of hyperparameters for all splits is the right way. Fortunately, despite the variances across splits, the average accuracy is within 1%p between two tuning schemes, and we keep the conclusion.