We thank the reviewers for their comments. We address all major comments below (including clarifying some potential misunderstanding from Reviewer #3). We believe that our paper makes an important first step to develop an effective framework to learn good metrics for persistence summaries; and the performance of our current framework on the challenging graph classification problem is already comparable or better than a range of existing state-of-the-arts approaches in the literature, and also outperforms all previous TDA based methods (sometimes by a large margin). We also note that all code/datasets are already made public and results can be reproduced.

Response to Reviewer #1. Thank you for several insightful suggestions (including the generalization of Lemma 3.2). We will incorporate them in the revision. A few clarifications: (1) Indeed: we use 10 times $10 * 10$-fold nested cross validation, and hyperparameter tuning is done in the inner loop using the respective training data from current fold. (2) By "subsampling", we mean the *mini-batch* for stochastic gradient descent during the optimization (for each gradient computation we choose a random subset (mini-batch) of size 50). **All** data from input datasets are used to generate results. For Reddit-12K, as pointed out in Supplement, the EigenPro method by [Ma and Belkin, NIPS 2017] is used to speed up kernel-SVM. But that does not involve subsampling input data. (3) We use extended persistence diagrams for all datasets. (When double-checking results, we noticed that currently what we reported for WKPI-kM for (only) IMDB and Reddit are from using 0-D standard persistence (sub + super levelsets). The results using extended persistence are better: $75.5 \pm 0.1; 51.2 \pm 0.5; 59.5 \pm 0.5; 49.4 \pm 0.6$ for IMDB-Binary, IMDB-Multi, Reddit5K, Reddit12K, respectively.) (4) For heat maps for graph data, we already provide two examples in Figure 3 of the Supplement.

Response to Reviewer #2. Thank you for your comments. Your main comment that we should run other methods with the same setup as ours is a very valid point: (i) All results on topology-based methods are already done in exactly the same setup as ours. (ii) We used 10 times 10*10-fold nested cross validation. Most recent work in graph classification literature use (10 times) 10-fold cross validation, which was partly why we didn't re-run the results. We have now re-run the two best performing approaches *GIN* and *RetGK* using our setup. Results are in Table 1: RetGK stays roughly the same. GIN improves slightly, although our results are still comparable or better than it in general. Note that GIN uses **node attributes**, while our results are obtained without them (based purely on graph structure).

Table 1: Accuracy of GIN, RetGK and Persistence Fisher Kernel (k_{PF}) on graph benchmark datasets

	NCI1	NCI109	PTC	PROTEIN	DD	MUTAG	IMDB-BIN	IMDB-MULTI	Reddit5K	Reddit12K
RetGK	84.5±0.2	84.8±0.2	62.9±1.6	75.4±0.6	81.6±0.4	90.0±1.1	72.3±1.0	47.7±0.4	55.8 ±0.5	48.5± 0.2
GIN	82.4±1.6	86.5±1.5	67.8±6.5	76.7±2.6	81.1±2.5	89.0±7.5	75.6±5.3	52.4±3.1	57.2±1.5	47.9± 2.1
k_{PF}	81.7±0.2	78.5±0.3	62.4±1.2	75.2±0.3	79.4±0.3	85.6±0.5	71.2±0.7	48.6±0.2	56.2±0.4	47.6±0.3

We also provide a few clarifications: (1) For "subsampling technique": please see point (2) in our response to Reviewer #1 above. (2) Our method treats graphs as **non-attributed** (see e.g, lines 314-315 of submission), and persistence summaries are generated using the Ricci curvature and Jaccard index descriptor functions as described in lines 300-303 of submission. Using only graph structures, we can already obtain similar or better results than those previous approaches using attributes, and it will be interesting to explore in the future whether using attributes can further improve performance. (3) Persistent homology can be computed for graphs in a standard way by treating functions defined on it as a piecewise-linear function. We will elaborate on all these points in the revision, add more background on persistent homology, and provide an intuitive example of persistence for neuron trees to explain the ideas.

Response to Reviewer #3. Thank you for your comments. We address / clarify your major comments: (1) *Comparing with other TDA methods on graph datasets:* Indeed, we **already provide** that in Table 3 of Supplement (also see lines 281-284 of submission), and our method outperforms them in all cases. (Note the change of results for WKPI-kM for IMDB+Reddit in our response to Reviewer #1). Results for Persistent Fisher kernel (PF), run in the same setup as ours, are in Table 1 which we will add to the revised paper. The performance of PF on these data is similar to the SW method (which we already compare with) and our method outperforms PF in all cases (sometimes by a large margin).
(2) *Regarding the paper on NCA by Goldberger et al, and metric learning on Mahalanobis distance:* The similarity with NCA is perhaps superficial, mostly in the sense that both intuitively optimize some total "in-class" distance or similarity (which is common in most metric learning approaches). The differences are fundamental. (2.a) NCA utilizes the probability of correct KNN based classification, while idea of our approach comes from spectral clustering, which leads to different precise formulation of the objective function. (2.b) NCA and other metric learning for Mahalanobis distance learns a *linear transformation* of grid points (coordinates) in persistence image (PI), while our approach learns a weight function $\omega : \mathbb{R}^2 \to \mathbb{R}$ defined on the birth-death plane containing PIs. Also, our pseudo-distance on persistence images involve a non-linear kernel. (2.c) NCA is learning a $N \times N$ matrix A, where N is the total number of grid points in one PI. The number of parameter in NCA is roughly N^2 (or dN for dimension reduction to d-D space). However, in our *parametric formulation*, we only learn the parameters of the target weight function: e.g, for ω being a mixture of m isotropic Gaussians, the number of parameters is $O(m)$, which can be several orders of magnitude smaller than N^2.

(3) *Grid size for persistence image (PI):* We don't think, nor have observed that the discretization of PI has significant effect on performance, which is why we didn't set it as a hyperparameter. (Note that the **same size** (see "Setup for persistence images" on Pg 4, Supplement) is used for **all datasets**.) For example, on PROTEIN: the accuracy for grid sizes $s * s$ with $s = 10; 20; 30; 40; 50; 60$ is $76.7; 78.6; 78.5; 78.5; 76.3; 73.2$. We will include these in revision.