# HOMOPHILY-BASED FILTRATION LEARNING FOR GRAPH NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Graph neural networks (GNNs) are a powerful method of learning representations of graph-structured data. While they excel at learning class-discriminative representations of nodes in homophilous graphs, where connecting nodes tend to belong to the same class, many GNNs struggle with heterophilous graphs whose inter-class connections can muddy the message passing. Inspired by this finding, we propose a topological filtration scheme, treating graphs as 1-dimensional simplicial complexes N with a filter function based on estimated edge heterophily, and introduce two methodologies that use a backbone GNN to learn from the resulting graph filtration. The first trains a GNN on each graph in the filtration sequence consecutively for a portion of the total training time, using embeddings from previous graphs to initialize node embeddings in subsequent graphs. The second approach uses a novel message passing scheme to pass messages jointly within each and between graph levels in the filtration sequence with common nodes. Both methods enhance the influence of early birth adjacent nodes in homophilous subgraphs yet allow for the model to learn from the full range of heterophilous and homophilous connections in the graph. We further extend our approach to learn a graph filtration sequence of graphs through a learnable node filter function. In this work, we provide a framework for multiscale learning of graph data structures for any backbone GNN, along with an approach for learning a geometrically significant sequence of nested subgraphs obtained through a leaned topological filtration of graph data structures. Experiments show that our heterophily-filtered GNNs achieve superior node classification accuracy on both heterophilous and homophilous networks.

## 1 INTRODUCTION

Graphical representations offer an accessible and actionable means to express and expand our understanding of the world, from observed phenomena to abstract notions, by providing explanations or interpretations based on relationships and connections between elements, circumstances, and complex systems. Graph Neural Networks (GNNs) have emerged as a powerful tool for learning from graph structured data by generalizing the convolution operator to unstructured domains by leveraging message passing or neighborhood aggregation schemes to harness structural information. Recently, in graph machine learning, the classical task of node classification has demonstrated strong results when employing GNNs.

Despite their potential, GNNs face several known challenges limiting their effectiveness, one in particular being graphs with high heterophily, whose edges predominantly adjoin nodes of disparate classes. In such cases, it is not clear which parts of the graph structure are most helpful for guiding the message passing and expressivity; of these obstacles, message passing networks schemes probably can not distinguish specific topologies, such as two triangles from a 6-cycle, and a leading known impediment to GNNs are graphs with high heterophily, namely, graphs whose edges predominantly adjoin nodes of disparate classes, which leads to oversmoothing Murphy et al. (2019); Sato et al. (2021); Li et al. (2018). This paper introduces a graph learning methodology that addresses the limitations of conventional GNNs by offering a novel approach employing hierarchical GNNs-leveraging topological insights from persistent homology, namely persistence filtration, to reveal meaningful structure in graphs informed by class connectivity and allowing for a hierarchy of multilevel graph resolutions.

To filter our graphs, we first learn class relationships between nodes by estimating edge-level heterophily labels using the features of their adjacent nodes. We then leverage the probability of homophily assigned to each edge with persistent homology through a persistence filtration function defined over edge values to obtain a multilevel sequence of graphs well suited for hierarchical GNNs. By formulating the learning problem over graph edges, integrating topological summaries, and learning multilevel subgraph hierarchies based on persistent filtration dependent on class connectivity, our approach offers an ordered sequence of graphs that captures both local and global structural information while also addressing critical issues such as over smoothing, reducing computational complexity, and a paradigm for GNNs more resilient to heterophily We argue that by providing a more nuanced understanding of graph topology and leveraging hierarchical learning, our proposed approach can lead to more resilient, computationally efficient, and expressive GNNs. Using either of two hierarchical GNN training strategies we propose, a message-passing GNN can place greater emphasis on edges that are more likely homophilous. At the same time, we retain the ability to learn from heterophilous edges too. Empirically, our approach to topological filtration and hierarchical graph learning compares favorably to previous learning techniques, including standard learning models, standard GNNs, and heterophily-specific models. We also demonstrate competitive results over established benchmark datasets for evaluating GNN performance in low and high heterophily scenarios.

Our contributions are as follows:

- We propose a homophily-based topological filtration method for graphs, where heterophilous edges may be used for learning but given different emphasis than homophilous edges. Our filtration preserves topological structure compared to approaches that drop heterophilous edges.
- We introduce two different methods for improving backbone GNN models, especially on graphs with heterophily, by allowing them to learn from a multi-scale sequence of topological graph filtrations based on class connectivity.
- We introduce an approach for learning a topological filtration of graphs, minimizing the obstacle of heterophily, to obtain a nested sequence of graphs for training backbone GNNs.
- We experimentally demonstrate that our methods lead to improved node classification accuracy relative to previous models.

## 2 Backgropund and Related Work

### 2.1 Filtration Functions and Topological Filtration

We now provide the necessary background on topological filtration, how it applies to graphs, and recent advancements in learnable topological filtration schemes.

**Simplicial Complexes and Topological Filtration** A simplicial complex is a collection of simpler topological objects called simplices (such as points, line segments, triangles, etc.) such that every face of a simplex is also included in the simplicial complex, and the intersection of any two simplices is either empty or a shared face. Let $(K^i)_{i=0}^m$ be a sequence of simplicial complexes, or collections of topological objects such that $\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$. Then, $(K^i)_{i=0}^m$. Then, $(K^i)_{i=0}^m$ is called a filtration of $K$. In the filtration of $K$, the sequence of each simplicial complex $K_i$ is subset to the subsequent $K_{j>i}$.

**Filter Functions** Let $\mathbb{E}$ be the domain of simplices and $\mathbb{K}$ the set of possible simplicial complexes over $\mathbb{E}$, the filter function $f : \mathbb{K} \times \mathbb{E} \to \mathbb{R}$ maps each simplicial complex $K$ and simplex $e$ to a real value $f(K,e)$. A learnable filter $f_\varepsilon$ is a filter $f(\varepsilon,K,e)$ is differentiable with respect to an additional parameter $\varepsilon$.It has been shown that if pairwise simplex values are distinct, such a differentiable filter function can be used in the context of $k$-persistent homology as a mapping of simplicial complexes that is differentiable end to end Hofer et al. (2020).

**Construction of Topological Filtration** The filtration is constructed by including simplices in increasing order of their filter values. Simplices are first assigned values through a filter function or

a filter function for the compassion of simplices is derived. Starting with the empty set, simplices are added consecutively in sorted order by increasing the filter function value. Simplices are added such that the resulting simplicial complex contains the previous subset simplicial complexes for lower filter function values. Each step corresponds to subcomplex $K_p$ where $p$ is the threshold value. For filter function $f(e)$ over simplices $e \in \mathbb{E}$ the subcomplex $K_p$ during filtration contains all simplices with filter value less than $p$. The set of simplices in $K_p$ is then $\{e \in K_p | f(e) \leq p\}$. As the filtration progresses, one can track how topological features like connected components, loops, and voids appear and disappear. These features are summarized using tools like persistent homology.

Having now described the relevant concepts in computational topology abstractly, we give some background on graphs and the application of computational topology to graphs.

**Graphs and Homophily** Let $G = (\mathscr{N}, \mathscr{E})$ be a graph consisting of node set $v \in \mathscr{N}$ and edge set $\{e = (u,v) | e \in \mathscr{E} \text{and } u, v \in \mathscr{N}\}$ where edge $e = (u,v)$ is said to adjoin nodes $u$ and $v$. We denote a neighborhood around node $v$ as $N(v) = \{u \in \mathscr{N} | (u,v) \in \mathscr{E}\}$ to consist of all nodes in $\mathscr{N}$ that share an incident edge with $v$ in $\mathscr{E}$. For $k \geq 1$ we denote the $k$-hop neighborhood of node $v$, $N_k(v)$, as the set of all nodes $u \in \mathscr{N}$ such that there is a connected path of $k$ or fewer edges adjoining $u$ and $v$. For nodes $u$ and $v$ with labels $y_u$ and $y_v$ and edge set $\mathscr{E}_i$ for graph $G_i$ in the filtration sequence, the homophily ratio Zhu et al. (2020) is given by $h_i = \frac{1}{|\mathscr{E}_i|} \sum_{v \in \mathscr{E}_i} \frac{|\{u \in \mathscr{N}(v) : y_u = y_v\}|}{|\mathscr{N}(v)|}$.

**Topological Filtration of Graphs** We can interpret a graph $G$ as a 1-dimensional simplicial complex whose vertices (0-simplices) are the graph nodes and whose edges (1-simplices) are the edges. In the topological filtration of graphs, we can think of filtration as a growth process of a weighted graph. As the graph grows, new edges and nodes are introduced, introducing new connected components and neighborhood structures.

Node neighborhoods thus share this property: for each node $v_i \in \mathbf{V}_{p_i} \cap \mathbf{V}_{p_j}$, e.g. $\mathscr{N}_{p_i}(v_i)$ for $v_i \in \mathbf{V}_{p_i}$ and $\mathscr{N}_{p_j}(v_i)$ for $v_i \in \mathbf{V}_{p_j}$, we have that $\mathscr{N}_{p_i}(v_i) \subseteq \mathscr{N}_{p_j}(v_i)$. We give all graphs the full node set found at the lowest threshold level of the filtration sequence, but nodes that would not otherwise exist in the graph at a higher persistence level are disconnected. Early birth nodes, which appear for lower value thresholds, then remain in subsequent graphs in the filtration sequence, with neighborhood structure for each node potentially differing for each node between different graphs in the sequence.

## 3 Methodologies for Filtration Learning

### 3.1 Estimating Edge Homophily Scores

We first filter graphs by the likelihood an edge adjoins nodes of disparate classes. To accomplish this we first frame the task of node classification as an edge classification problem, where the edge classifier serves as a filter function that prescribes filter values to edges representing the inferred likelihood that an edge is homophilous.

Given a graph $G = (\mathscr{N}, \mathscr{E})$ containing labeled nodes $v$ with labels $y_v$ and node features $h_u$ and $h_v$ for a subset or all of node set $\mathscr{N}$, we can derive binary edge labels for edge $e = (u,v)$ as $y_{(u,v)} = \left\{ \begin{array}{l} 1 \text{ if } y_u = y_v \\ 0 \text{ if } y_u \neq y_v \end{array} \right\}$ for graphs with binary or multi-class labels. Our edge filter function is a multilayer perceptron with a single logistic output defined as $\text{MLP}\big(\text{COMBINE}(h_u, h_v)\big)$. In our where $h_u$ and $h_v$ are the feature vectors of adjacent nodes $u$ and $v$ and COMBINE is implemented via concatenation (other choices of functional operator, such as dot product, are possible).

We then train the filter function using the derived homophily edge labels $y_{u,v}$ on a subset of the graph and infer values between 0 and 1 to all remaining edges of the graph, predicting whether an edge is homophilous or not.

### 3.2 Homophily-based Topological Filtration of Graphs

We perform filtration on graphs using a sorted ordering of 1-simplices, or edges, with filter values assigned by $f(\mathscr{E}, e) \forall e \in \mathscr{E}$ of graph $G$, which correspond to the likelihood an edge is homophilous

or not. Beginning at filter threshold $p = 0$ we then add edge $e = (u, v)$ and its incident nodes $u$ and $v$ if $f(e) \leq p$ and such that the expanded graph $G_p$ at level $p$ of the filtration sequence remains a simplicial complex containing all $e = (u, v)$ such that $f(e) \leq p$ once all edges have been assigned a filter function value.

We observe the topological filtration sequence at $P$ levels of filtration thresholds to model multiscale topological information. In contrast to hierarchical graph-level representation learning methods that learn to pool each input graph Ying et al. (2018), this gives us several graph representations of the underlying data to use as input to a graph neural network, which we denote the *multi-scale sequence of graphs* or *graph hierarchy*. A smaller threshold value produces a more highly filtered graph with fewer edges that are more likely to be homophilous. Thus, we obtain a hierarchy of graphs $G_1, \ldots, G_P$ where for $i \in [1, \ldots, P], G_{i-1} \subseteq G_{p_i}$: that is, $G_{i-1}$ is an induced subgraph defined on a subset of the vertices in $G_i$.

### 3.3 Graph Neural Networks with Topological Filtration

We consider two methods for learning node representations for classification using GNNs, which exploit the topological information from the nested sequence of graphs obtained through topological filtration, referred to as the graph hierarchy, in the context of training GNNs. The first method modifies the training of GNN over graph hierarchies, while the second modifies the architecture to pass messages jointly between all hierarchical graph levels.

### 3.4 Topological Filtration for Hierarchical Graph Neural Network

#### 3.4.1 Multi-Scale Successive Training

For our first method, we use a GraphSAGE architecture which learns node features via message passing in the spatial domain Hamilton et al. (2017).

Instead of training a GNN for $N$ epochs on the finest graph $G_P$, we train it for $\frac{N}{P}$ epochs each on graphs $G_1, \ldots, G_P$, using the embeddings from $G_{i-1}$ to initialize the embeddings of corresponding nodes in $G_i \; \forall \; i \in [1 \ldots, P]$. The MLPs and node embedding weight matrices for target and neighboring node embeddings are shared between graph hierarchies. Training begins with the smallest (most highly filtered) graph and iteratively increases in graph size. We also successively share the learned embedding weight matrices and matrices of the MLPs as well.

#### 3.4.2 Multi-Scale Joint Training

**Message**

Message passing occurs during a single layer and iteration $k$ where each node's feature $h_v^k$ is updated to incorporate the features associated with nodes within its neighborhood and the layer's learnable parameters denoted $\varepsilon$ and $W$. Message passing in this way can be viewed as a representational signal moving through adjacencies within a neighborhood. With the output of each layer serving as input to the subsequent layer, 'farther' serves to incorporate information from more distant nodes diffusing throughout the network. The result is a learned target node representation based neighborhood $r$, denoted $\mathcal{N}_p^r(v)$ for node $v$. Messages are further aggregated, similar to pooling in standard convolutional networks, in that they are functionally propagated and interlaced. In 1 node $v$ is highlighted in yellow within each graph of the nested filtration sequence and the dotted blue arrows denote the messages passed to node $v$ in each neighborhood $N_p^r(v)$ where $p$ denotes the graph level in the sequence at filtration value $p \in \{0, \cdots, P\}$ that the neighborhood being considered corresponds to. A single node that exists within multiple graphs of the filtration sequence can have neighborhood structures that differ but are subsets to subsequent neighborhoods later in the sequence of graphs resulting from filtration.

$$h_m^r(u, v) = \text{Message}(h_u^r, h_v^r | v \in \mathcal{N}_p^r(v)))$$

where $h_u$ and $h_v$ are feature vectors of nodes $u$ and nodes $v$.

**Aggregation Within Neighborhoods** Messages are then aggregated for each neighborhood $N_p^r(v)$ across nodes in the target node's neighborhood. In Figure 1 this is illustrated with the regions highlighted in blue where, for each graph, messages from each neighborhood $r$ of $v$ in that graph are

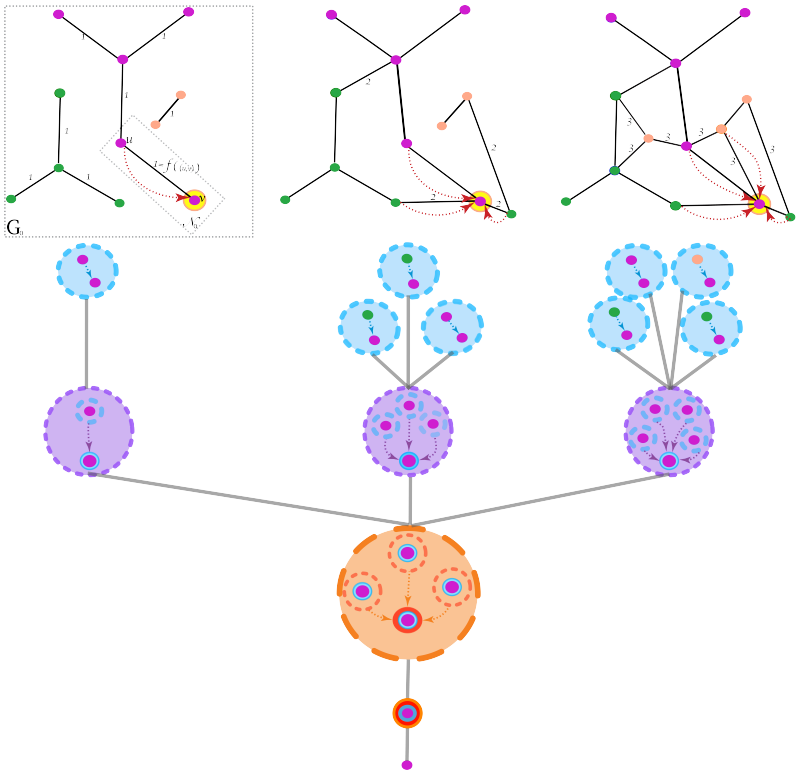

Figure 1: A sequence of three nested graphs from topological filtration (top row) offering three levels of graph resolution. Messages are shown being passed (dotted arrow) to the bottom right node that, having survived the filtration, is common to all graphs in the multi-scale sequence, differing in neighborhood connectivity. For each subgraph $G_p$ of the total $P$ graphs in the graph hierarchy $\{G_0, \cdots, G_p, \cdots, G_P\}$, messages are passed from each neighborhood $N^r(v)$ to the target node. Messages at each graph level are then aggregated within neighborhoods (highlighted in blue). Messages from all neighborhoods within each graph are then aggregated between one another (highlighted in purple). We introduce (highlighted in orange) across neighborhood aggregation, where messages are passed across neighborhoods in the multi-scale sequence of graphs. The target node's feature representation is then updated with the aggregated embeddings.

aggregated.

$$h^r_{v,p} = \text{AGG}_{u \in \mathcal{N}^r_p(v)}(h^r_m(u,v))$$

**Aggregation Between Neighborhoods** Messages are then aggregated between all neighborhood $N^r_p(v) \in N_p(v)$ of nodes in target node $v$'s neighborhood in subspace $p$

$$h^{N_p}_{v,p} = \text{AGG}_{\mathcal{N}^r_p \in N_p}(h^r_{v,p})$$

In Figure 1 this is illustrated with the regions highlighted in purple.

**Multi-Neighborhood Aggregation Across Graph Neighborhoods** This work introduces the step highlighted in orange in Figure 1. Namely, once messages from each neighborhood have been aggregated, messages are then aggregated across all target node neighborhoods in all graphs within the sequence of graphs to which target node $v$ belongs.

$$h^N_v = \text{AGG}(h^{N_p}_{v,p})$$

In Figure 1 this is illustrated with the regions highlighted in orange.

**Update** Finally, the feature representation of the target node is updated with aggregated embeddings

$$h_v = \text{Update}(h^N_{v_P}, h^t_a)$$

For node $v_i$ belonging to node set $\mathbf{V}_{p_i}$ of subgraph $\mathbf{G}_{p_i}$, the feature representation $h^k_{v_i}$ at GNN layer $k \in \{1, \cdots, K\}$ first combines self-representations from the previous level of aggregation with aggregated neighbor information:

$$h^k_{v_i} = COMBINE\big(W^{k-1}_s h^{k-1}_v, \\ AGGR\big(\{W^{k-1}_n h^{k-1}_u : u \in \mathscr{N}_{p_i}(v_i)\}\big)\big)$$

Once this is done, the resulting embedding is combined with the embedding representation from other persistence subgraphs. As an example for two persistence levels $p_i$ and $p_j$ where $v_i \in \mathbf{V}_{p_i} \subset \mathbf{G}_{p_i}$ and $v_j \in \mathbf{V}_{p_j} \subset \mathbf{G}_{p_j}$, we have $h^k_v = \sigma\big(COMBINE(h^k_{v_i}, h^k_{v_j})\big)$.

**Learned Node Filtrations of Graphs**

This work introduces a second, learnable node filter function for learning a filtration of nodes in graphs within the graph hierarchy during training. Filtration of nodes in the graph hierarchy aims to learn which graphs (and their respective nodes) in the nested graph sequence obtained from topological filtration are most informative for multi-scale or hierarchical graph learning. Graph Isomorphism Networks Xu et al. (2018), (GIN-$\varepsilon$) not only have theoretical expressivity benefits for distinguishing graph substructures but have also been shown to be a viable candidate as a learnable filter function for persistent homology when also combined with a multilayer perceptron Hofer et al. (2020). We then use a learnable node filter function during Multi-Scale Joint Training defined as

$$f_p(\mathscr{N}_p, u) := \text{MLP}(\text{GIN}^p_\varepsilon(h^p_u)) \to \mathbb{R}$$

.

where $h^p_u$ is the embedding representation of node $u$ in $\mathscr{N}_p$ in graph $G_p = (\mathscr{N}_p, \mathscr{E}_p)$ for graph level $p$ of the graph hierarchy. $\text{GIN}^[_\varepsilon$ is a GIN network which is learnable and differentiable in $\varepsilon$ Xu et al. (2018), and MLP is a multilayer perceptron which takes the resulting embedding from $\text{GIN}^p_\varepsilon$.The resulting real-valued output of $f_p$ is the learned filter value, which is a scalar $q_p(u)$ for node $u$ in graph level $G_p$.

This work introduces a methodology for using the learned filter values $q_p(u)$ to filter graphs within the graph hierarchy. To learn the importance and reduce or increase the contribution of graphs in the graph hierarchy during multi-scale joint aggregation, we use the $q_p$ filter value for attention type based aggregation by weighing the aggregated embedding obtained from graph $G_p$ in the hierarchy before multi-scale aggregation across all graphs $G_i$ in the multi-scale filtration sequence of $P$ graphs $G_0, G_1, \cdots, G_P$. Before the combination step of multi-scale aggregation, we then factor each node's resulting embedding from each graph level by its respective filter value $q_p$ In doing so, the learned filter value can learn to eliminate or accentuate the contribution of embeddings from specific graphs in the multi-scale graph sequence.

The learned filtration of graphs in the graph hierarchy during multi-scale joint aggregation can then be expressed as the combined embedding from each subgraph, where each subgraph's embedding is obtained with a backbone GNN and a filter function $f_p(\mathscr{N}_p, u)$, unique to each graph level, for a final embedding that is weighted by a filter value $q_p$ before global combination. For node $v_P$ in the highest resolution graph $G_P$ in the filtration sequence, we then have:

$$h^k_{v_P} = COMBINE\big(f_{0,\varepsilon_0}(h^{k-1}_{v_0}) \cdot W^{k-1}_{s,0} h^{k-1}_{v_0} \\ , f_{i,\varepsilon_i}(h^{k-1}_{v_1}) \cdot W_{s,i} h^{k-1}_{v_1} \\ f_{P,\varepsilon_P}(h^{k-1}_{v_P}) \cdot W_{s,P} h^{k-1}_{v_P}\big)$$

From the original precomputed filtration sequence of graphs, we then learn a subset filtration sequence of graphs through the learnable node filter function $f_p(\mathscr{N}_p, u)$, which learns a weighted attention on each node in each graph level to obtain a learned filtration sequence of graphs.

# 4 EXPERIMENTS

**Implementation** We implement all models with PyTorch and all GNNs with Pytorch Geometric Paszke et al. (2019); Fey & Lenssen (2019). Topological filtration is performed with Dionysus Morozov (2007).

**Hyperparameters and Computing Environment** We train for 321 epochs and perform validation every 8. We employ early stopping based on validation scores for a deviation of $1e-3$ with patience of 4 rounds of validation accuracy scores for the edge filter function's training and 10 for the multi-scale model used. For dropout, the edge filter function uses a value of 0.3, and multi-scale models have a value of 0.65. As the size of the datasets tested varied, hidden dimensions varied to account for the limitations of the machine being used and model overfitting. We perform a parameter sweep for each of the following hyperparameters over the values provided here. Filtration thresholds and number of graphs in graph hierarchy with early stopping:(0.5,0.7,0.9), (0.8,0.9), (0.6,0.9), (0.6,0.9), (0.8,0.9), (0.7), (0.8), (0.9), (0.1). For all filtration sequences of graphs, the entire graph with no thresholding was also included. Learning Rate of Multi-Scale methodology (MsST or MJT): 3e-4, 3e-3, 0.03, 1e-4, 1e-3, 1e-2 Learning rate of edge filter function: 1e-3,3e-3 ,1e-2 ,3e-2 Weight decay: 5e-8 ,5e-6 ,5e-5 ,5e-4 ,0

All experiments were run on a laptop with 3 GB GeForce GTX 970M with 1280 CUDA Cores GPU and 3.5GHz i7-6700HQ processor running Ubuntu, Linux.

## 4.1 DATASETS

We use several different standard node classification benchmarks covering a range of homophily levels.

Planetoid (CiteSeer, Cora, and PubMed): These are widely used citation networks, where nodes represent documents, and edges represent citation links between them. The task is typically node classification, where the goal is to predict the category of each document Yang et al. (2016). These datasets are well known to have high homophily.

Wikipedia Network (Chameleon): is a graph derived from the page-to-page link network of Wikipedia articles. Nodes represent Wikipedia pages, and edges represent hyperlinks between them. The nodes are classified based on the traffic levels of the web pages. This dataset is used to study the performance of graph learning methods designed for heterophilous networks. Rozemberczki et al. (2021).

WebKB (Cornell, Wisconsin, and Texas): The WebKB datasets, including Cornell, Wisconsin, and Texas, consist of webpages collected from computer science departments of various universities. Nodes represent webpages, and edges represent hyperlinks between them. The task is to classify the type of webpage (e.g., course, faculty, student) and these datasets are also known to have high levels of heterophily.Peixoto & Emmert-Streib (2019).

| Dataset | Subdataset | Nodes | Edges | Classes |
|---|---|---|---|---|
| Planetoid | Cora | 2,708 | 5,429 | 7 |
| | CiteSeer | 3,327 | 4,732 | 6 |
| | PubMed | 19,717 | 44,338 | 3 |
| WikipediaNetwork | Chameleon | 2,277 | 31,421 | 5 |
| WebKB | Cornell | 183 | 295 | 5 |
| | Wisconsin | 251 | 499 | 5 |
| | Texas | 183 | 309 | 5 |

Table 1: Summary of datasets and their characteristics.

**Node Classification Results** In Table 2, we report the test accuracy results of MsST and MsJT in the context of the reported accuracies of other state-of-the-art models with equivalently proportioned train, validation and test splits. Our methodologies excel in all except one dataset, achieving the highest and second highest accuracies.

| Model | Texas | Wisconsin | Chameleon | Cornell | CiteSeer | Pubmed | Cora |
|---|---|---|---|---|---|---|---|
| Homophily Ratio | 0.06 | 0.16 | 0.25 | 0.11 | 0.74 | 0.80 | 0.83 |
| MsST | 96.72% | 95.62% | 90.12% | 95.63% | 88.73% | 71.04% | 93.17% |
| MsJT | 89.27% | 93.57% | 91.25% | 91.71% | 93.49% | 91.78% | 95.39% |
| Geom-GCN-X | 75.67% | 76.47% | 89.32% | 66.66% | 79.12% | 90.11% | 90.57% |
| Dropedge-X | 83.52% | N/A | 64.94% | N/A | N/A | N/A | 87.36% |
| H_2GCN-X | 84.86% | 86.67% | 59.39% | 82.16% | 77.07% | 89.59% | 87.81% |
| HMPNN | N/A | N/A | N/A | N/A | N/A | N/A | 92.16% |
| X-NSD | 85.95% | 89.41% | 68.68% | 86.49% | 77.14% | 89.49% | 87.30% |
| DCM | 85.71% | 89.33% | 53.76% | N/A | 78.60% | 88.61% | 86.63% |

Table 2: Test accuracy results for node classification over datasets, sorted by homophily level, for our methodologies (MsST and MsJT), heterophily oriented model designs, and topological deep learning (TDL) models (sorted by row). Accuracy results are reported for test sets with train, validation, and test split of 40%, 20%, and 20% in comparison to the reported optimal accuracy for similar works with equivalently proportioned splits. The highest accuracy scores are highlighted in green, with the second highest outlined in blue Feng et al. (2019); Huang et al. (2022); Pei et al. (2020); Zhu et al. (2020); Bodnar et al. (2022)

# 5 Related Work

Here, we review related works on graph neural networks, whether using similar techniques based on filtration learning, or focused on the problem we consider of learning from graphs with heterophily.

## 5.1 Graph Neural Networks for Heterophilous Graphs

For a recent overview of graph neural networks, see Zhu et al. (2023); Luan et al. (2024). Methods designed to handle graphs with heterophily usually focus on one of two challenges posed by heterophily. The first is that in heterophilous graphs, nodes may not have relevant neighbors (e.g. neighbors of the same class) to attend to, and one solution is to expand the receptive field of graph neural networks to allow them to learn non-local dependencies. Such approaches include higher-order neighborhood aggregation, proposed by Zhu et al. (2020), rewiring the graph based on other forms of node similarity Pei et al. (2020), or introducing non-local attention Zhu et al. (2020). Such approaches may incur a higher computational cost.

Our work falls into a second category of approaches, which recognize that heterophily also may muddy the message passing of graph neural networks by causing messages to be aggregated from irrelevant neighbors. Such approaches modify the message passing to better suit heterophilous graphs, for example by performing it over signed Yan et al. (2022) or directed **?** edges, or by learning a class compatibility matrix that may be used to control message passing **?**. Other approaches include rewiring the graph to be better suited to heterophily, by connecting nodes based on some other measure of similarity or distance Pei et al. (2020); Bi et al. (2024); Li et al. (2023).

Most related to our work, this rewiring has been performed with a edge classifier that predicts if an edge is heterophilous or homophilous Huang et al. (2022); Xue et al. (2024). A backbone graph neural network is trained on a graph that has been preprocessed using the classifier to improve its homophily properties. The edge classifier may be used to filter the graph only by pruning heterophilous edges Huang et al. (2022), or new edges may also be added that are predicted to be homophilous Xue et al. (2024). Both methods result in a single output graph which loses any information that the heterophilous edges may contain. In contrast, our approach gives a model access to the input graph at a variety of filtration levels, giving us the best of both worlds: an ability to use all the information in the original graph, while also being able to emphasize the high-homophily connections in a more filtered graph. Moreover, by using persistent homology to perform our filtration rather than arbitrarily thresholding edges, we maintain more topological structure in the filtered graphs.

### 5.2 Filtration Learning for Graph Neural Networks

Graph filtration learning using persistent homology has been used for graph-level representation learning Hofer et al. (2020). For node classification, persistent homology has also been used to guide message passing in graph neural networks Zhao et al. (2020). More related to our work, Leventhal et al. (2023) has used graph neural networks to classify topological objects in image data, learning successively or jointly from hierarchical levels of a topological filtration. Here, we show how to derive such a hierarchical filtration for graphs specifically based on homophily, and we develop a node-level filtration that allows us to attend to different levels of filtered graphs.

## 6 Conclusion

We propose graph neural network methods that learn from hierarchies of graphs representing a sequence of nested graphs obtained from topological filtration. Treating graphs from the perspective of 1-D simplicial complexes, our filtration is defined as being over 1-cells (graph edges). We obtain a sequence of simplicial complexes, where each complex in the series is subset to the subsequent, from a sorted ordering of 1-cells, thresholded by filtration values prescribed to simplexes through a filter function defined as a learning model trained to infer if edges are homophilous, akin to a class similarity measure between 0-cells (graph nodes). The proposed approach is generalizable to any edge based classification task or graph with weighted edges that afford an ordering. We then present two methods for hierarchical graph learning. The former, achievable for any standard GNN model, performs message passing across subsequent graphs in the graph hierarchy, aggregating node embeddings sequentially moving up the graph filtration sequence. the ladder introduces a novel multi-scale message passing scheme with aggregation performed jointly across graphs in the filtration sequence, with learned attention based aggregation or learned topological filtration of subset graph nodes, effectively learning the degree of contribution graphs within the hierarchy contribute to learned aggregated node embeddings. Our results demonstrate promise for increased node classification accuracy on heterophilous networks compared to conventional GNNs.

## REFERENCES


Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Make heterophilic graphs better fit gnn: A graph rewiring approach. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Cristian Bodnar, Alexandru T Vasilache, Michael Rüttgers, Mohsen Afshar, Michael Bronstein, Markus Böhm, and Federico Monti. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *arXiv preprint arXiv:2202.04579*, 2022.

Yifan Feng, Haoxuan You, Zeyang Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3558–3565, 2019.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL `https://arxiv.org/abs/1903.02428`.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

Christoph Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *International Conference on Machine Learning*, pp. 4314–4323. PMLR, 2020.

Minhao Huang, Chen Cai, Yuyu Liang, Jun Wang, and Le Song. Revisiting the role of heterophily in graph representation learning: An edge classification perspective. In *Advances in Neural Information Processing Systems*, 2022.

Samuel Leventhal, Attila Gyulassy, Valerio Pascucci, and Mark Heimann. Modeling hierarchical topological structure in scientific images with graph neural networks. In *2023 IEEE International Conference on Image Processing (ICIP)*, pp. 2995–2999. IEEE, 2023.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Shouheng Li, Dongwoo Kim, and Qing Wang. Restructuring graph for higher homophily via adaptive spectral clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8622–8630, 2023.

Sitao Luan, Chenqing Hua, Qincheng Lu, Liheng Ma, Lirong Wu, Xinyu Wang, Minkai Xu, Xiao-Wen Chang, Doina Precup, Rex Ying, et al. The heterophilic graph learning handbook: Benchmarks, models, theoretical analysis, applications and challenges. *arXiv preprint arXiv:2407.09618*, 2024.

Dmitriy Morozov. Dionysus - a library for computing persistent homology. `http://www.mrzv.org/software/dionysus/`, 2007. Accessed: 2024-08-10.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning*, pp. 4663–4673. PMLR, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8024–8035, 2019.

Hongwei Pei, Bingzhe Wei, Kevin Chang, Yuxiao Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Tiago P Peixoto and Frank Emmert-Streib. Network cartography of university websites: A comparative analysis. In *Proceedings of the 30th International Conference on Computer Networks*, pp. 1–11, 2019.

Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Multi-scale attributed node embedding. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1235–1243, 2021.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pp. 333–341. SIAM, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yanfeng Xue, Zhen Jin, and Wenlian Gao. A data-centric graph neural network for node classification of heterophilic networks. *International Journal of Machine Learning and Cybernetics*, pp. 1–11, 2024.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 1287–1292. IEEE, 2022.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48. PMLR, 2016.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In *International Conference on Artificial Intelligence and Statistics*, pp. 2896–2906. PMLR, 2020.

Jiong Zhu, Yujun Yan, Lingfei Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

Jiong Zhu, Yujun Yan, Mark Heimann, Lingxiao Zhao, Leman Akoglu, and Danai Koutra. Heterophily and graph neural networks: Past, present and future. *IEEE Data Engineering Bulletin*, 2023.